From Accepting Computation to Satisfying Kripke Model.

A detailed presentation of parts of the EXPTIME-hardness proof for PDL.

Author: Chris Ausema
First Supervisor: prof. dr. H.H. Hansen
Second Supervisor: D.R.S. Ramanayake, PHD

A thesis presented for the degree of
Bachelor of Computing Science

university of groningen

faculty of science and engineering

Faculty of Science and Engineering
University of Groningen
The Netherlands
22-07-2021

Abstract

A method that can be used to find the time complexity of a problem is to reduce it to another problem, or vice versa. In the book Dynamic logic by David Harel, Dexter Kozen and Jerzy Tiuryn, it is shown that you can find the lower bound complexity of the satisfiability problem of PDL by using reduction from the membership problem for polynomial space bound single tape alternating Turing machines (PSST ATM). The proof shown was an informal high level proof, which only showed the intuition behind their proof and could therefore benefit from a more rigorous presentation. We will use the definitions of PDL and PSST ATMs to show a more detailed presentation of half their proof. In this thesis we will show that when a PSST ATM has an accepting computation on some input, then a Kripke model exists that encodes the workings of the PSST ATM.

CONTENTS

1 Introduction **3**

2 Computational Complexity **3**
2.1 Computational problems 4
2.1.1 Search Problems 4
2.1.2 Decision Problems 4
2.2 complexity classes 4

3 Propositional Dynamic Logic **5**
3.1 Syntax 5
3.2 Semantics 6
3.3 Satisfiability problem 8

4 Polynomial Space Bounded Single Tape Alternating Turing Machines **8**

5 From Accepting Computation To Satisfying Kripke Model **11**
5.0.1 Preliminaries 12
5.0.2 START 13
5.0.3 CONFIG 14
5.0.4 MOVE 22
5.0.5 ACCEPTANCE 27
5.0.6 The Final Step 29

6 Conclusion **31**

References **31**

1 Introduction

When we talk about the complexity of an algorithm, we normally talk about how long it will take to finish, or how much space the algorithm will use, this is called the computational time or space complexity (or simply, the complexity) of an algorithm.
The term complexity is also used when talking about different problems, like the traveling salesman problem, to indicate the complexity of all possible algorithms that could solve said problem. For these problems there could be many (possibly infinitely many) algorithms with different complexities that could solve them, therefore we make a distinction between the lower bound complexity and upper bound complexity of a problem.
One of the problems of which we know the lower and upper bound complexity is the satisfiability problem of propositional dynamic logic (PDL). PDL is a dynamic logic of which the syntax consists of propositions and programs. The formulas of PDL are formed by combining atomic propositions and atomic programs which are then interpreted over Kripke models. The satisfiability problem of PDL is stated as follows: "Given a PDL formula φ, is there a Kripke model $\mathcal{K}$ that satisfies φ".
In section 8.2 of the book *Dynamic Logic* by Harel et al [4] it is shown that the lower bound complexity of the satisfiability problem of PDL is EXPTIME, by reducing the membership problem of polynomial space bounded single tape alternating Turing machines (PSST ATMs) to the satisfiability problem of PDL. The membership problem is stated as: "Given a PSST ATM $\mathcal{M}$ and an input x, will $\mathcal{M}$ accept x?"
In this project we will give a detailed mathematical presentation of part of the high-level proof by Harel et al. in [4].
In their proof Harel et al. constructed a formula $ACCEPTS_{\mathcal{M},x}$, which has the property that a Kripke model that satisfies $ACCEPTS_{\mathcal{M},x}$ will encode the working of a PSST ATM $\mathcal{M}$ on input x. They gave the definition of this formula, but never showed that an accepting computation of $\mathcal{M}$ on input x results in a Kripke model that satisfies $ACCEPTS_{\mathcal{M},x}$. They also never showed that if there was a Kripke model that satisfies $ACCEPTS_{\mathcal{M},x}$ then there is an accepting computation on a PSST ATM $\mathcal{M}$ with input x.

We started this project with the goal of finding the lower bound complexity of the satisfiability problem of another logic, called Game Logic [7]. This is a modal logic which is syntactically and semantically similar to PDL, Therefore the first step will be to understand the proof of the lower bound of SAT-PDL. This however proved to be more challenging than first expected, and therefore we will focus in this thesis on the following theorem:

For all PSST ATMs $\mathcal{M}$ and all inputs x there exists a PDL formula $ACCEPTS_{\mathcal{M},x}$ such that if there is an accepting computation of M on input x then there is a satisfying Kripke model of $ACCEPTS_{\mathcal{M},x}$.

The following chapters of the thesis will focus on the following:
In section 2 we will give short background on computational complexity. In section 3, we will describe the syntax, semantics and axiomatizations of propositional dynamic logic, as well as describe what it means for a formula to be satisfiable. In section 4, we will present the PSST ATMs. We will present the proof of the theorem in section 5 . We will then conclude the thesis in section 6 ,in which we will discuss possible future work.

2 Computational Complexity

A question that is often heard when someone is designing an algorithm, or deciding how difficult a certain task is, is "how efficiently can this be done". This efficiency is then related to some resource, for example how much time will it take to solve this problem, or how much space will this algorithm take up. The field concerned with how much resources a problem or an algorithm uses is called computational complexity theory. In this section we present computational complexity by first describing what a computational problem is. We will then go over what complexity classes are, including characteristics.

In this section we encode most objects with their binary string representation. This will allow us to talk about complexity and computational problems in the general sense without worrying about the type of objects that are used. We will follow the conventions as described in [5, p. 18], where they define the set of all string of length n, with n being a natural number, as $\{0,1\}^n$ and the set of all strings as $\{0,1\}^*$. We also use the symbols $\bot$ and $\emptyset$ to denote false and the empty set respectively.

2.1 Computational problems

There are many different types of computational problems. In this section we will explain two of those: search problems and decision problems, together with the definition of solving those problems, but first we will present the general notion of a computational problem. As stated in [2, p. 1054], a computational problem is defined as a binary relation R over a set of problem instances I, and a set of problem solutions S. An example would be: "For an integer x, find the closest prime number y for which $x < y$". In this example the set I of problem instances would be the set of all integers $\mathbb{Z}$, the set S of problem solutions would be the set of all prime numbers, and the binary relation R would be the set of pairs (x, y), for which $x \in I, y \in S, x < y$ and for all i, such that $x < i < y, i \notin S$. This is also an example of a search problem.

2.1.1 Search Problems

A search problem is a computational problem where, given an instance x, we are tasked to find the, possibly multiple, corresponding solution(s) from S, or conclude that no such solution exists [5]. These problems could range from finding the greatest common divisor of two numbers to finding the shortest path (for example the traveling salesman problem). Solving a search problem requires a solving strategy $g : \{0,1\}^* \rightarrow \{0,1\}^* \cup \{\perp\}$, which maps the input instance $x \in \{0,1\}^*$ to a solution $y \in \{0,1\}^*$ if one exists or it maps to $\perp$ if no solution exists. This leads to the definition of solving search problems as given in [5, p.19]:

Definition 2.1. *Let the binary relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, $x \in \{0,1\}^*$ be a problem instance and $S = \{0,1\}^* \cup \{\perp\}$. We will define $R(x)$ as the set of all problem solutions $y \in S$ corresponding to x, so $R(X) \stackrel{def}{=} \{y \mid (X, y) \in R\}$. Then $g : \{0,1\}^* \rightarrow \{0,1\}^* \cup \{\perp\}$ solves the search problem of R, iff $g(x) \in R(x)$, if $R(x) \neq \emptyset$, else $g(x) = \perp$.*

By this definition it should hold that $g(x)$ gives a solution for x if one exists, that is if $R(x)$ is not empty.

2.1.2 Decision Problems

Decision problems are problems that can be answered with either yes or no, which we will represent in binary with 1 and 0. A decision problem is defined as follows: "Given an instance x, is $x \in S$" [5]. A simple example of a decision problem is whether a certain number is odd or even. Just like with search problems, solving a decision problem requires a solving strategy $h : \{0,1\}^* \rightarrow \{0,1\}$, which maps the input instance to either a 0 (no,reject) or 1 (yes,accept). Solving decision problems has been defined in [5, p.20] as follows:

Definition 2.2. *Let S be a set with $S \subseteq \{0,1\}^*$ and $x \in \{0,1\}^*$ be a problem instance, then $g(x) = 1$ iff $x \in S$ and $g(x) = 0$ iff $x \notin S$.*

In complexity theory we mainly consider decision problems, for their simplicity of only having two possible answers [6]. We will only consider decision problems for the remainder of this section.

2.2 complexity classes

The complexity of a problem is defined by the amount of a specific resource it is using. When we restrict the amount of resources that a machine can use to solve a problem, then we get what we call a complexity class [4, p. 38]. A complexity class is specified by: [6]:

1. The model of computation (a multi-string Turing machine is often chosen).
2. The mode of computation (for example deterministic or nondeterministic).
3. The resources used (we choose to use time and space).
4. A bound function $f : \mathbb{N} \rightarrow \mathbb{N}$.

We often choose to use a multi-string Turing machine (TM) as our model of computation for decision problems. We often see the set of problem solutions S as a language X, and we say that a TM $\mathcal{M}$ decides the language $X \subseteq \{0,1\}^*$ iff for all $x \in \{0,1\}^*$ $\mathcal{M}$ halts at an accepting state if $x \in X$, and halts at an rejecting state if $x \notin X$ [3]. We say that a multi-string TM $\mathcal{M}$ accepts a problem instance x iff $\mathcal{M}$ takes x as input, and halts at an accepting state.

The mode of computation is most often chosen to be deterministic or nondeterministic [6, p.139]. The result of a deterministic TM is always fixed, while the result of a nondeterministic TM might differ.
The bound function f is a function that maps non-negative integers to non-negative integers and, as the name suggest, it defines how much resources a multi-string TM uses to determine whether a problem instance x is in its language. We say a multi-string TM runs in time $f(n)$ if it takes at most $f(n)$ steps to halt, and it runs in space $f(n)$ if from the start state all reachable configurations us at most $f(n)$ work tape cells [4].

We define a complexity class as follows [6]:

Definition 2.3. *A complexity class is the set of all languages decided by some multi-string TM $\mathcal{M}$, operating in some mode, such that, for any input x, $\mathcal{M}$ expends at most f(|x|) units of the chosen resource.*

We define the complexity classes $\mathbf{DTIME}(f(n))$ and $\mathbf{NTIME}(f(n))$ as the set of all languages (problems) accepted by a deterministic multi-string TM, and a nondeterministic TM respectively, that runs in time $f(n)$.
We define the complexity classes $\mathbf{DSPACE}(f(n))$ and $\mathbf{NSPACE}(f(n))$ as the set of all languages (problems) accepted by a deterministic multi-string TM, and a nondeterministic TM respectively, that runs in space $f(n)$ [4, p.39].
Using this we can define some popular complexity classes:

$$\begin{aligned}\mathbf{P} &\stackrel{def}{=} \mathbf{DTIME}(n^{\mathbf{k}}),\\ \mathbf{NP} &\stackrel{def}{=} \mathbf{NTIME}(n^{\mathbf{k}}),\\ \mathbf{EXPTIME} &\stackrel{def}{=} \mathbf{DTIME}(n^{\mathbf{ck}}).\end{aligned}$$

We say that a problem P is in a certain time complexity class C if the language of P is in C. We say a problem P is hard for a certain time complexity class C if we can reduce (transform one problem in to another) all problems in C to P.

3 Propositional Dynamic Logic

In this section, we present the modal logic propositional dynamic logic (PDL). PDL is a dynamic logic that is used to reason about nondeterministic programs, which are programs of which the outcome could be different between executions on the same input.
The models over which PDL formulas are normally interpreted are called Kripke models, named after Saul Kripke. These models consist of a set of states, a set of binary relations and a valuation function. When PDL is interpreted over a Kripke model, the programs of PDL language are seen as binary relations over the states of the model. In the next subsections we define the syntax and semantics of PDL, and lastly the satisfiability problem of PDL.

3.1 Syntax

The syntax of PDL is defined in terms of programs and propositions. The set Π_0 is the set of atomic programs, which we denote by $a, b, c\ldots$. Φ_0 denotes the set of atomic propositions denoted by $p, q, r\ldots$. In the following, we assume the sets Π_0 and Φ_0 to be given. PDL is not one language, it is a collection of infinitely many languages. A language of PDL is a language parameterized by the pair (Π_0, Φ_0). Using the atomic programs and atomic propositions we are able to form complex programs and propositions. Complex programs are formed by using program operators and complex propositions are formed by using propositional operators.

Definition 3.1. *Given a finite non-empty set Π_0 of atomic programs and a finite non-empty set Φ_0 of atomic propositions, we define the language PDL(Π_0, Φ_0) as the pair (Π, Φ), where Π is the set of all programs and ϕ the set of all propositions. the programs α and the propositions φ are defined inductively as follows:*

$$\begin{aligned}\alpha &:= a \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \varphi?\\ \varphi &:= 0 \mid 1 \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \varphi \leftrightarrow \varphi \mid [\alpha]\varphi\end{aligned}$$

with $a \in \Pi_0$ and $p \in \Phi_0$.

In definition 3.1 the ; denotes sequential composition, meaning: the program $\alpha;\beta$ is the program that first executes α and then immediately executes β. The nondeterministic choice $\alpha \cup \beta$ means to choose either α or β nondeterministically and execute it. α^* means to execute α a nondeterministically chosen finite number (zero or more) of times. The operator $*$ is also called the iteration operator. The program $\varphi?$ is the test program, this program will test φ, if true then proceed, fail if false.
The proposition 0 represents propositional falsity, the proposition $\neg\varphi$ is the negation of proposition φ, which means that when φ is true, $\neg\varphi$ is false and vice versa. The proposition $[\alpha]\varphi$ is the proposition that states that after executing program α it is necessary that φ holds. The [ ] operator is called the box operator.

Note that the definitions of Π and Φ are intertwined, due to the box and test operator. The box operator is a propositional construct, but requires a program to execute. And the test operator is a program construct, but it needs a proposition to test.
The operator $\langle\rangle$ is the dual of [] and defined as:

$$\langle\alpha\rangle\varphi = \neg[\alpha]\neg\varphi$$

The formula $\langle\alpha\rangle\varphi$ is true at states where the execution of program α leads to at least one state where φ is true.

3.2 Semantics

PDL formulas are interpreted over Kripke models. In a Kripke model atomic programs are interpreted as binary relations over the set of states.

Definition 3.2. *A Kripke model for the language PDL(Π_0, Φ_0) is a 3-tuple:*

$$\mathcal{K} = (S, R, V),$$

with S being a set of states, R a mapping from atomic programs to binary relations on S, with $R : \Pi_0 \to \mathcal{P}(S \times S)$, and V being a valuation function, $V : \Phi_0 \to \mathcal{P}(S)$, giving the set of states in which atomic propositions hold.
We write R_a for the relation $R(a)$ and uR_aw when $(u, w) \in R_a$. We also define the set $R_a(u) = \{v \in S \mid (u, v) \in R_a\}$ as the set of a-successors of u.
For non-atomic programs and propositions the definition of R and V are given by mutual induction on the structure of the program or proposition.

Definition 3.3. *The definitions of V and R for non-atomic propositions and programs are as follows:*
$(K1)$ $V(0) = \emptyset$
$(K2)$ $V(\neg\varphi) = S - V(\varphi)$
$(K3)$ $V(\varphi \wedge \psi) = V(\varphi) \cap V(\psi)$
$(K4)$ $V(\varphi \vee \psi) = V(\varphi) \cup V(\psi)$
$(K5)$ $V(\varphi \to \psi) = V(\neg\varphi) \cup V(\psi)$
$(K6)$ $V(\varphi \leftrightarrow \psi) = V(\varphi \to \psi) \cap V(\psi \to \varphi)$
$(K7)$ $V([\alpha]\varphi) = \{u \mid R_\alpha(u) \subseteq V(\varphi)\}$
$(K8)$ $R_{\alpha;\beta} = \{(u, v) \mid \exists w \in S$ *such that* $w \in R_\alpha(u)$ *and* $v \in R_\beta(w)\}$
$(K9)$ $R_{\alpha\cup\beta} = R_\alpha \cup R_\beta$
$(K10)$ $R_{\alpha^*} = \bigcup_{n\geq 0} (R_\alpha)^n$
$(K11)$ $R_{\varphi?} = \{(u, u) \mid u \in V(\varphi)\}$.

We write $\mathcal{K}, u \models \varphi$ when $u \in V(\varphi)$, meaning that φ is true at state u. Similarly we write $\mathcal{K}, u \not\models \varphi$ when $u \notin V(\varphi)$, which says that φ is not true at state u.

Example 3.4. In figure 1 you can see the graph of the Kripke model $\mathcal{K} = (S, R, V)$ with:
$\Pi_0 = \{a, b\}$
$\Phi_0 = \{X, Y, Z\}$
$S = \{S1, S2, S3, S4\}$
$R_a = \{(S1, S2), (S2, S2), (S1, S3)\}$

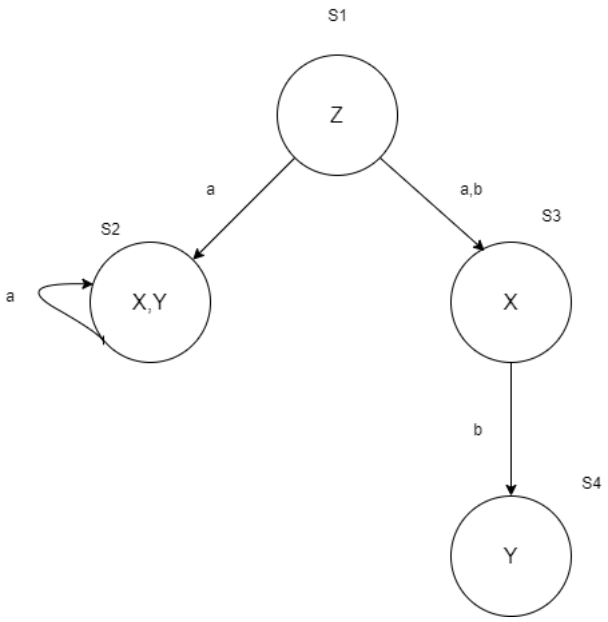

Figure 1: The graph of a Kripke model

$R_b = \{(S1, S3), (S3, S4)$
$V(X) = \{S2, S3\}$
$V(Y) = \{S2, S4\}$
$V(Z) = \{S1\}$

In this model we can see that:
$\mathcal{K}, S1 \models [a]X$
$\mathcal{K}, S2 \models X \wedge Y$
$\mathcal{K}, S1 \models \langle a; b\rangle Y$
$\mathcal{K}, S1 \models \langle a^*\rangle X$
$\mathcal{K}, S4 \models [a]X$
$\mathcal{K}, S4 \not\models \langle a\rangle X.$

We will show for some of the formulas that they are true by using the statements of definition 3.3:

$\mathcal{K}, S1 \models [a]X$
iff $S1 \in V([a]X)$ (definition $\models$)
iff $S1 \in \{u \mid R_a(u) \subseteq V(X)\}$ (K7)
iff $R_a(S1) \subseteq V(X)$
iff $\{S2, S3\} \subseteq V(X)$ (definition of $R_a(S1)$
iff $\{S2, S3\} \subseteq \{S2, S3\}$ (definition V(X))
iff $true$ (definition $\subseteq$)

$\mathcal{K}, S4 \models [a]X$
iff $S4 \in V([a]X)$ (definition $\models$)
iff $S4 \in \{u \mid R_a(u) \subseteq V(X)\}$ (K7)
iff $R_a(S4) \subseteq V(X)$

iff $\emptyset \subseteq V(X)$ (value of $R_a(S4)$)
iff $true$ ($\emptyset$ is a subset of every set)

$\mathcal{K}, S1 \models \langle a^* \rangle X$
iff $S1 \in V(\langle a^* \rangle X)$ (definition $\models$)
iff $S1 \in V(\neg[a^*]\neg X)$ (definition $\langle a \rangle$)
iff $S1 \in \{u \mid R_{a^*}(u) \not\subseteq V(\neg X)\}$ (K7)
iff $S1 \in \{u \mid \{v \in S \mid (u,v) \in R_{a^*}\} \not\subseteq V(\neg X)\}$ (definition $R_{a^*}(u)$)
iff $S1 \in \{u \mid \{v \in S \mid (u,v) \in \bigcup_{n\geq 0}(R_a)^n\} \not\subseteq V(\neg X)\}$ (definition R_{a^*})
iff $S1 \in \{u \mid \{v \in S \mid (u,v) \in \bigcup_{n\geq 0}(R_a)^n\} \not\subseteq (S - V(X))\}$ (K2)
iff $\{v \in S \mid (S1,v) \in \bigcup_{n\geq 0}(R_a)^n\} \not\subseteq (S - V(X))$
iff $\{S2, S3\} \not\subseteq (S - V(X))$ (definition $\{v \in S \mid (u,v) \in \bigcup_{n\geq 0}(R_a)^n\}$)
iff $\{S2, S3\} \not\subseteq \{S1, S4\}$ (definition S-V(X))
iff $true$ (definition $\not\subseteq$)

$\mathcal{K}, S4 \not\models \langle a \rangle X$
iff $S4 \notin V(\langle a \rangle X)$ (definition $\models$)
iff $S4 \notin V(\neg[a]\neg X)$ (definition $\langle a \rangle$)
iff $S4 \notin \{u \mid R_a(u) \not\subseteq V(\neg X)\}$ (definition $V(\neg[a]\neg X)$)
iff $R_a(S4) \not\subseteq V(\neg X)$
iff $\emptyset \not\subseteq (S - V(X))$ (definition $R_a(S4)$)
iff $false$ $\emptyset$ is a subset of every set

3.3 Satisfiability problem

A PDL formula ψ of some language PDL(Π_0, Φ_0) is satisfiable *iff* there exists a Kripke model $\mathcal{K}$, for which $V(\psi) \neq \emptyset$. The problem stating: "For some PDL proposition ψ of some PDL language PDL(Π_0, Φ_0), is there a Kripke model $\mathcal{K}$ that satisfies ψ?" is called the satisfiability problem of PDL.

Definition 3.5. *The satisfiability problem of PDL is defined as:*
Given a PDL formula ψ, does a Kripke model $\mathcal{K} = (S, R, V)$ exist for which $V(\psi) \neq \emptyset$? In other words, is there a state u in some Kripke model $\mathcal{K} = (S, R, V)$ for which $\mathcal{K}, u \models \psi$.

4 Polynomial Space Bounded Single Tape Alternating Turing Machines

In this section, we will define polynomial space bounded single tape alternating Turing machines (PSST ATM) [1]. Alternating Turing machines (ATM) are an alternating variation of the standard Turing machines. This means that the states of ATMs can be either universal, existential or negating, on top of being accepting or rejecting. A computation from configuration α, which is in an existential state q and which can reach configurations β, ω, γ is said to lead to acceptance iff at least one of the configurations β, ω, γ leads to acceptance. If for the same configuration α, state q was universal, then a computation from α will lead to acceptance iff all of the configurations β, ω, γ lead to acceptance. If the state q of α was negating, however, then α would only be able to reach one configuration, say β, and a computation

in α would lead to acceptance iff β would lead to rejection.

Two special cases of ATMs are the single tape version, for which there is only one work tape, and the polynomial space bounded ATM, for which the length of the work tape is finite, and the input x, with length n, is directly written on the work tape, enclosed with special end markers. In this section, we will restrict our discussion of ATMs to the combination of these two special cases.

Definition 4.1. *A PSST ATM is a 7-tuple:*

$$\mathcal{M} = (z, Q, \Sigma, \Gamma, \delta, q_0, g).$$

where:
z is a fixed constant,
Q is the set of states,
Σ is the input alphabet (with $[,] \in \Sigma$ being the begin and end markers),
Γ is the work tape alphabet (including the blank symbol # and $\Sigma \subseteq \Gamma$),
$\delta : (Q \times \Gamma) \rightarrow \mathcal{P}\,(Q \times (\Gamma - \{\#\}) \times \{-1,0,+1\})$ is the next move function,
q_0 is the start state,
g is the assignment function: $Q \rightarrow \{universal, existential\}$.

As can be seen from this definition, a PSST ATM uses the symbols [and] as end markers, these will specify the start and the end of the work tape. These markers make sure that the tape is of finite length (in contrast to standard Turing machines, for which an infinite tape is assumed). The end markers impose the following restrictions on the PSST ATM:

1. The tape will always contain the end marker [at tape position 0, and the end marker] at tape position $n^z + 1$.
2. The PSST ATM cannot overwrite the end markers.
3. The PSST can not move to the left of the left end marker, nor to the right of the right end marker.

It has been proven by Chandra et al. [1, Theorem 2.5 on p. 120] that for every ATM $\mathcal{G}$ with an accepting computation on input x, in time t or space s, with $t, s \in \mathbb{N}$, there exists an ATM $\mathcal{M}$ without negating states, that also has an accepting computation on input x, in time t or space s. Therefore for the remainder of this section we will focus on PSST ATMs without negating states whenever we talk about PSST ATMs.

The PSST ATM $\mathcal{M}$ will always be in exactly one state q, reading exactly one tape cell at some position i and will have exactly one symbol on each cell of its work tape, with $q \in Q$ and $0 \leq i \leq n^z + 1$. This is called a configuration of $\mathcal{M}$.

Definition 4.2. *Given a PSST ATM $\mathcal{M} = (z, Q, \Sigma, \Gamma, \delta, q_0, g)$ and an input string of size n, the set of all possible configurations of $\mathcal{M}$, $C_{\mathcal{M}}$, is defined as:*

$$C_{\mathcal{M}} = Q \times (\Gamma - \{\#\})^m \times \{\#\}^w \times \overline{N}(n^z + 1)$$

with $m < n^z + 1$, $w = n^z + 1 - m$ and $\overline{N}$ being the function that takes an integer n as input and returns the set: $\{i \mid i \in \mathbb{N}$ and $0 \leq i \leq n^z + 1\}$. A configuration α, with $\alpha \in C_{\mathcal{M}}$ is of the form:

$$(q, [, a_1, ..., a_m, \#^w,], i),$$

which represents the configuration that is in state q, has a work tape containing the symbol [on position 0, m times a symbol a, with $a \in \Gamma$, and blanks until the end of the tape where it is closed by the symbol].

At the start of a computation of $\mathcal{M}$, the machine will be in the start state, with input x written on the work tape followed by blanks, and it will be scanning cell 0.

Definition 4.3. *The start configuration of $\mathcal{M}$ on input x is denoted as:*

$$\sigma_{\mathcal{M}}(x) = (q_0, [, x_1, ..., x_n, \#^w,], 0),$$

with $x = x_1, ..., x_n$, $w = n^z - n$ and q_0 being the start state of $\mathcal{M}$.

The next move function $\delta : (Q \times \Gamma) \rightarrow \mathcal{P}(Q \times (\Gamma - \{\#\}) \times \{-1,0,+1\})$ is a function mapping the pairs consisting of states q from Q and a symbol a from the work tape alphabet, to the set of triples (p, b, d) where p is a state from Q, b a symbol from the work tape alphabet, and d a direction in which the tape head moves.
The way ATMs move along the function δ is as follows:
Assume we have some configuration α of $\mathcal{M}$ in state q, which is scanning a symbol a at tape position i. Let us also assume that there is a triple $(p, b, +1)$ which is in $\delta(q, a)$, and let us denote configuration β as the configuration reachable from α by the action $(p, b, +1)$. When $\mathcal{M}$ moves from α to β it will change its state from q to p, overwrite the currently read symbol a with symbol b and move the tape head by 1 position to the right.
The function δ does not always return one triple, it can return multiple, or even none. This makes it possible for every configuration to have 0 or more successor configurations.

A move of $\mathcal{M}$ consists of $\mathcal{M}$ going from one configuration to another, this move is represented by the next configuration relation $\vdash$.

Definition 4.4. *Given a PSST ATM $\mathcal{M}$ on input x, we call a configuration $\beta = (p, [, a_1, ..., a_{i-1}, b, a_{i+1}, ..., i_{n-i}, \#^w,], i+d)$ a successor of the configuration $\alpha = (q, [, a_1, ..., a_n, \#^w,], i)$ iff $(p, b, d) \in \delta(q, a)$. This is denoted as $\alpha \vdash \beta$ and we refer to $\vdash$ as the next configuration relation.*

A computation path of $\mathcal{M}$ is a sequence of steps starting from the start configuration, this is written as: $\sigma_{\mathcal{M}}(x) \vdash \alpha_1 \vdash ... \vdash \alpha_n$. The relation $\vdash$ is a set containing pairs of configurations (α, β). We call a configuration β reachable from some configuration α, if there is a computation path from α to β in 0 or more steps.

We define the computation tree T of a PSST ATM $\mathcal{M}$ on input x as follows:
The root node of the tree is the start configuration $\sigma_{\mathcal{M}}(x)$, from here the tree will create m different branches, one for each triple in the set returned by $\delta(q_0, [)$, which are all the successor configurations of $\sigma_{\mathcal{M}}(x)$. Remember from definition 4.3 that q_0 is the start state, and $\mathcal{M}$ is reading the symbol [on tape position 0. These branches will then branch out to their successor configurations. This process can result in a tree of infinite depth, however the set of configurations $C_{\mathcal{M}}$ does not contain an infinite amount of elements due to the work tape being of finite length. We define the number of all possible configurations c as: $c = |C_{\mathcal{M}}|$. We can be sure that the computation tree of $\mathcal{M}$ can reach all configurations of $C_{\mathcal{M}}$ that *can* be reached in c steps, therefore we can truncate the tree at depth c. Not all configurations of $C_{\mathcal{M}}$ might be reachable from $\sigma_{\mathcal{M}}(x)$, which means not all configurations of $C_{\mathcal{M}}$ need be in T.

Definition 4.5. *For all PSST ATMs $\mathcal{M}$ and all inputs x of length n, with c being the number of configurations of $\mathcal{M}$, we define the configurations of the computation tree of $\mathcal{M}$ on input x as follows:*

$$C_T = \{\beta \mid \sigma_{\mathcal{M}}(x) \vdash^* \beta, \textit{ in c or fewer steps}\}.$$

We will now define acceptance on the computation tree.
Whether a configuration leads to acceptance depends on the state of the configuration and the successor configurations. A state can be either universal or existential. For an existential configuration (a configuration with an existential state) to be accepting means that at least one of its successors must lead to acceptance. For a universal configuration (a configuration with a universal state) to be accepting means that all its successors must lead to acceptance. We denote a universal configuration with $\wedge$ and an existential configuration with $\vee$. We also define an accepting configuration as a universal configuration with no successors and a rejecting configuration as an existential configuration with no successors.
After truncating we might have some branches which do not return accepting or rejecting, but instead had successors which were truncated. We call these branches unfinished, which we denote by $\perp$. However we do want an existential configuration with at least one accepting successor to return accepting even if one or more of its branches did not finish their computation, and the same goes for the universal states that should return rejecting if at least one of its successors return rejecting. This is why we extend the domain of $\wedge$ and $\vee$ to the domain $\{0, \perp, 1\}$, as defined in [1, 118]:

$\wedge$:	1	$\perp$	0
1	1	$\perp$	0
$\perp$	$\perp$	$\perp$	0
0	0	0	0

$\vee$:	1	$\perp$	0
1	1	1	1
$\perp$	1	$\perp$	$\perp$
0	1	$\perp$	0

Now a universal state will return reject if one of its successors return reject, even when another successor has not returned yet, since $0 \wedge \bot = 0$. An existential state returns accepting if one of its successors return 1, since $\bot \vee 1 = 1$. If the computation returns $\bot$ for configuration $\sigma_{\mathcal{M}}(x)$, then the computation will be considered unfinished.

Proposition 4.6. *An accepting configuration is a universal configuration with no successor configurations and a rejecting configuration is an existential configuration with no successor configurations.*

We define acceptance recursively using the labeling function $l : C_T \to \{0, \bot, 1\}$ as follows:

$$l(\alpha, P) = \begin{cases} \text{iff } \alpha \in P: & \bot \\ \text{else:} & \begin{cases} \bigwedge\limits_{\alpha \vdash \beta} l(\beta, P \cup \{\alpha\}), & \text{if } \alpha \text{ is universal} \\ \bigvee\limits_{\alpha \vdash \beta} l(\beta, P \cup \{\alpha\}), & \text{if } \alpha \text{ is existential .} \\ 1 & \text{if } \alpha \text{ is accepting} \\ 0 & \text{if } \alpha \text{is rejecting} \end{cases} \end{cases}$$

The labeling function l is recursive, with P denoting the set of configurations that l has already seen in a previous recursion step. These are the configurations higher up in the computation tree. Intuitively speaking, l goes down along the branches of the computation tree without assigning any labels until it reaches a configuration with no successors, or a configuration that is already in P. A configuration with no successors is either universal or existential, and therefore it is labeled with 1 (accepting) or 0 (rejecting).

Whether a branch loops can be discovered in less than c steps, as is implemented in l. This is because there is a maximum of c configurations, meaning that the only way for a PSST ATM to loop is by having a branch in T which is not solely constructed with unique configurations. If in T there are two (or more) configurations that are the same, let us name this configuration α, then these two configurations α will also have the same successors β, due to δ, and therefore these successors β will have the same successors γ, until eventually it reaches configuration α again. Therefore by adding all configurations to a set P as they are labelled by l, we can give the label $\bot$ to configurations that were already labelled before, and therefore avoiding loops. We can now define:

$$l^*(\alpha) \stackrel{def}{=} l(\alpha, \emptyset).$$

Definition 4.7. *A PSST ATM $\mathcal{M}$ accepts an input x iff $l^*(\sigma_{\mathcal{M}}(x)) = 1$, rejects x iff $l^*(\sigma_{\mathcal{M}}(x)) = 0$ and is unfinished for x iff $l^*(\sigma_{\mathcal{M}}(x)) = 0$.*

The question of whether an input string x will be accepted by a PSST ATM $\mathcal{M}$ is called the membership problem for PSST ATMs.

Definition 4.8. *The membership problem for PSST ATMs is defined as follows:*
Given a PSST ATM $\mathcal{M}$ and an input string x, does $\mathcal{M}$ accept x?

It has been proven by Chandra et al. that the complexity of the membership problem for PSST ATMs is **EXPTIME**-hard [1, corollary 3.6 p. 122].

5 From Accepting Computation To Satisfying Kripke Model

A proof of the lower bound complexity of SAT-PDL being **EXPTIME** has been given by Harel et al. in the book Dynamic Logic [4, p. 216]. This proof was done by reducing the membership problem for PSST ATMs to SAT-PDL. They do this by creating a PDL formula called $ACCEPTS_{\mathcal{M},x}$, for which they state that all satisfying Kripke models encode the workings of a PSST ATM, we will give the definition of $ACCEPTS_{\mathcal{M},x}$ later. The reduction results in a proof of the following theorem:

Theorem 5.1. *For all PSST ATMs $\mathcal{M}$ and all inputs x there exists a PDL formula $ACCEPTS_{\mathcal{M},x}$ such that there is an accepting computation of $\mathcal{M}$ on input x iff there is a satisfying Kripke model of $ACCEPTS_{\mathcal{M},x}$.*

This theorem can be split into two separate theorems:

Theorem 5.2. *For all PSST ATMs $\mathcal{M}$ and all inputs x there exists a PDL formula $ACCEPTS_{\mathcal{M},x}$ such that if there is an accepting computation of $\mathcal{M}$ on input x then there is a satisfying Kripke model for $ACCEPTS_{\mathcal{M},x}$.*

Theorem 5.3. *For all PSST ATMs $\mathcal{M}$ and all inputs x there exists a PDL formula $ACCEPTS_{\mathcal{M},x}$ such that if there is a satisfying Kripke model for $ACCEPTS_{\mathcal{M},x}$, then there is an accepting computation of $\mathcal{M}$ on input x.*

The proof as given by Harel et al. is an informal proof since they only described, but did not prove, the formulas of which $ACCEPTS_{\mathcal{M},x}$ is build. It could therefore benefit from a more detailed presentation. In this section we present a detailed proof of theorem 5.2, and we will give an intuition of how to proof theorem 5.3 in the section 6.

5.0.1 Preliminaries

In this proof it will be necessary to describe PSST ATMs as PDL formulas. For this purpose we introduce $\text{PDL}_{\mathcal{M},x}$, a variant of PDL that is parameterized by a PSST ATM $\mathcal{M}$ and an input x.

Definition 5.4. *Given a PSST ATM $\mathcal{M}$ and some input x of length n, the language $PDL_{\mathcal{M},x}$ is the language $PDL(\Pi_0^{\mathcal{M}}, \Phi_0^{\mathcal{M}})$, where:*

$$\begin{aligned}\Pi_0^{\mathcal{M}} &= \{NEXT\}\\ \Phi_0^{\mathcal{M}} &= \{\{STATE_i^q \mid \textit{for all } q \in Q \cup \{l,r\} \textit{ and } 0 \le i \le n^z + 1\},\\ &\quad \{SYMBOL_i^a \mid \textit{for all } a \in \Gamma \textit{ and } 0 \le i \le n^z + 1\},\\ &\quad ACCEPT\}.\end{aligned}$$

The atomic proposition $STATE_i^q$ says that the machine is currently in state q and scanning tape position i, we also introduce $STATE_i^l$ and $STATE_i^r$, with $l, r \notin Q$. The proposition $STATE_i^l$ says that the currently scanned cell is to the left of i, and $STATE_i^r$ says that the currently scanned cell is to the right of i. The atomic proposition $SYMBOL_i^a$ says that the symbol a is currently written on cell i. The atomic proposition ACCEPT is a proposition that will always be true at configurations from which there is an accepting computation of $\mathcal{M}$, and false at configurations from which there is a rejecting computation of $\mathcal{M}$. We interpret the atomic program NEXT as the next state relation.

We will interpret $\text{PDL}_{\mathcal{M},x}$ over a Kripke model $\mathcal{K}_T = (S, R, V)$ which represent the computation tree of the PSST ATM from which $\text{PDL}_{\mathcal{M},x}$ is parameterized, on an input x.

Lemma 5.5. *For all ATMs $\mathcal{M}$ and all inputs x, the computation tree T of the computation of $\mathcal{M}$ on input x can be represented as a Kripke model $\mathcal{K}_T$.*

Proof. Let $\mathcal{M}$ be a PSST ATM with some input string x. We can now define a Kripke model $\mathcal{K}_T$ as follows:
S is the set of states representing the different configurations in the computation tree of $\mathcal{M}$, given by (definition 4.5):

$$S = \{\beta \mid \sigma_{\mathcal{M}}(x) \vdash^* \beta, \text{in c or fewer steps}\}.$$

R is the function $R : \Pi_0 \to \mathcal{P}(S \times S)$, since $\text{PDL}_{\mathcal{M},x}$ has only one atomic program, $NEXT$, the set returned by R is equal to the relation R_{NEXT}. We define the relation R_{NEXT} to be equal to the relation $\vdash$.
The valuation function V takes an atomic proposition as parameter and returns the set of states for which that proposition holds true. Since every element in S is a configuration, and a configuration of $\mathcal{M}$ has the form (definition 4.2)

$$(q, [, a_1, ..., a_{n^z},], i),$$

we define the valuation function $V : \Phi_0 \to \mathcal{P}(S)$ on atomic propositions as follows:
$V(STATE_i^q) := \{u \mid u \in S, u \text{ is in state } q \text{ and } u \text{ is in position } i\}$
$V(STATE_j^l) := \{u \mid u \in S, 0 \le i < j \text{ and } u \text{ is in position } i\}$
$V(STATE_j^r) := \{u \mid u \in S, j < i \le n^z + 1 \text{ and } u \text{ is in position } i\}$
$V(SYMBOL_i^a) := \{u \mid u \in S, u \text{ has symbol } a \text{ written at tape position } i\}$
$V(ACCEPT) := \{u \mid u \in S, l^*(u) = 1\}$ (definition 4.7). □

For the remainder of this section we will assume that we have a PSST ATM $\mathcal{M}$ with an input x for which $\mathcal{M}$ has an accepting computation. $\mathcal{M}$ and x are used to parameterize $\text{PDL}_{\mathcal{M},x}$ and by lemma 5.5 the computation tree T of the accepting computation also corresponds to a Kripke model $\mathcal{K}_T$. The computation on $\mathcal{M}$ starts at configuration $\sigma_{\mathcal{M}}(x)$, therefore we will prove $ACCEPTS_{\mathcal{M},x}$ on the state corresponding to configuration $\sigma_{\mathcal{M}}(x)$, that is:

$$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models ACCEPTS_{\mathcal{M},x}. \tag{1}$$

The formula $ACCEPTS_{\mathcal{M},x}$ is defined as:

$$ACCEPTS_{\mathcal{M},x} = START \wedge [NEXT^*](CONFIG \wedge MOVE \wedge ACCEPTANCE) \wedge ACCEPT, \tag{2}$$

therefore:

$$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models START \wedge [NEXT^*](CONFIG \wedge MOVE \wedge ACCEPTANCE) \wedge ACCEPT. \tag{3}$$

This can be split into three different statements:

$$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models START \tag{4}$$
$$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models ACCEPT \tag{5}$$
$$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models [NEXT^*](CONFIG \wedge MOVE \wedge ACCEPTANCE) \tag{6}$$

Equation 5 is true by the definition of $V(ACCEPT)$, since T is an accepting computation tree. $START$, $CONFIG$, $MOVE$ and $ACCEPTANCE$ are short hand notations for formulas, for which we will give the descriptions below, together with the proof showing that $\mathcal{K}_T$ satisfies these formulas.

5.0.2 START

We will start with the formula $START$:

Definition 5.6. *We define the formula START as* $START = STATE_0^{q_0} \wedge \bigwedge_{1 \le i \le n} SYMBOL_i^{x_i} \wedge \bigwedge_{n+1 \le i \le n^z} SYMBOL_i^{\#}$.

The formula $START$ is true at the start configuration of $\mathcal{M}$ where the tape-head reads the left most cell, is in the start configuration q_0 and the contents of the input string $x = x_1, ..., x_n$ are written on the work tape.

Lemma 5.7. $\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models START$.

Proof.

$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models START$

iff $\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models STATE_0^{q_0} \wedge \bigwedge_{1 \le i \le n} SYMBOL_i^{x_i} \wedge \bigwedge_{n+1 \le i \le n^z} SYMBOL_i^{\#}$ (definition START)

iff $\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models STATE_0^{q_0}$ and $\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models \bigwedge_{1 \le i \le n} SYMBOL_i^{x_i}$ and $\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models \bigwedge_{n+1 \le i \le n^z} SYMBOL_i^{\#}$ (definition K12)

It will suffice to prove these three statements separately. We start by showing that $\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models STATE_0^{q_0}$.

$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models STATE_0^{q_0}$

iff $\sigma_{\mathcal{M}}(x) \in V(STATE_0^{q_0})$ (definition $\models$).

$V(STATE_0^{q_0})$ returns the set of all configurations which are currently in state q_0 and scanning cell 0. By definition 4.3 configuration $\sigma_{\mathcal{M}}(x)$ is in state q_0, therefore $\sigma_{\mathcal{M}}(x) \in V(STATE_0^{q_0})$ is true, and $\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models STATE_0^{q_0}$ holds.

Next, we prove that $\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models \bigwedge_{1 \leq i \leq n} SYMBOL_i^{x_i}$.

$$
\begin{aligned}
&\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models \bigwedge_{1 \leq i \leq n} SYMBOL_i^{x_i} \\
&\qquad \text{iff } \sigma_{\mathcal{M}}(x) \in V(\bigwedge_{1 \leq i \leq n} SYMBOL_i^{x_i}) && \text{(definition } \models) \\
&\qquad \text{iff } \sigma_{\mathcal{M}}(x) \in \bigcap_{1 \leq i \leq n} V(SYMBOL_i^{x_i}) && (K3) \\
&\qquad \text{iff for all i, such that } 1 \leq i \leq n : \sigma_{\mathcal{M}}(x) \in V(SYMBOL_i^{x_i}) && \text{(definition } \cap).
\end{aligned}
$$

$V(SYMBOL_i^{x_i})$ returns the set of all configurations for which their work tape contains symbol x_i at tape position i. By definition 4.3 the work tape of configuration $\sigma_{\mathcal{M}}(x)$ contains the symbol x_i corresponding to the i'th symbol of input x, with $1 \leq i \leq n$, on tape cells 1 to n, therefore: $\sigma_{\mathcal{M}}(x) \in \bigcap_{1 \leq i \leq n} V(SYMBOL_i^{x_i})$ is true.

$$
\begin{aligned}
&\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models \bigwedge_{n+1 \leq i \leq n^z} SYMBOL_i^{\#} \\
&\qquad \text{iff } \sigma_{\mathcal{M}}(x) \in V(\bigwedge_{n+1 \leq i \leq n^z} SYMBOL_i^{\#}) && \text{(definition } \models) \\
&\qquad \text{iff } \sigma_{\mathcal{M}}(x) \in \bigcap_{n+1 \leq i \leq n^z} V(SYMBOL_i^{\#}) && (K3) \\
&\qquad \text{iff for all } i, \text{ such that } n+1 \leq i \leq n^z : \sigma_{\mathcal{M}}(x) \in V(SYMBOL_i^{\#}) && \text{(definition } \cap)
\end{aligned}
$$

$V(SYMBOL_i^{\#})$ returns a set of all configurations for which the work tape contains the symbol # from position $n+1$ up till, and including n^z. By definition 4.3 the work tape of configuration $\sigma_{\mathcal{M}}(x)$ contains the symbol # from the position $n+1$ up till, and including n^z therefore: $\sigma_{\mathcal{M}}(x) \in \bigcap_{n+1 \leq i \leq n^z} V(SYMBOL_i^{\#})$ is true.

□

We now need to prove statement 6.

Lemma 5.8. $\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models [NEXT^*](CONFIG \wedge MOVE \wedge ACCEPTANCE)$

This formula is prefixed with $[NEXT^*]$ which means that for all states reachable by u in 0 or more steps $CONFIG \wedge MOVE \wedge ACCEPTANCE$ needs to hold. For this we will proof the following for all states α from S:

$$
\begin{aligned}
\mathcal{K}_T, \alpha &\models CONFIG && (7) \\
\mathcal{K}_T, \alpha &\models MOVE && (8) \\
\mathcal{K}_T, \alpha &\models ACCEPTANCE && (9)
\end{aligned}
$$

5.0.3 CONFIG

The formula CONFIG is defined as follows:

Definition 5.9. *The formula* $CONFIG=$

$$
\bigwedge_{0 \leq i \leq n^z+1} \bigvee_{a \in \Gamma} \left(SYMBOL_i^a \wedge \bigwedge_{\substack{b \in \Gamma \\ b \neq a}} \neg SYMBOL_i^b \right) \tag{10}
$$

$$
\wedge \left(SYMBOL_0^{[} \wedge SYMBOL_{n^z+1}^{]} \right) \tag{11}
$$

$$
\wedge \bigvee_{0 \leq i \leq n^z+1} \bigvee_{q \in Q} STATE_i^q \tag{12}
$$

$$\wedge \bigwedge_{0 \le i \le n^z+1} \bigvee_{q \in Q \cup \{l,r\}} \left(STATE_i^q \wedge \bigwedge_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} \neg STATE_i^p \right) \tag{13}$$

$$\wedge \bigwedge_{0 \le i \le n^z} \bigwedge_{q \in Q \cup \{l\}} \left(STATE_i^q \rightarrow STATE_{i+1}^l \right) \tag{14}$$

$$\wedge \bigwedge_{1 \le i \le n^z+1} \bigwedge_{q \in Q \cup \{r\}} \left(STATE_i^q \rightarrow STATE_{i-1}^r \right) \tag{15}$$

The subformula on line 10 is interpreted as "every cell contains exactly one symbol", the subformula on line 11 states that the work tape of a configuration is enclosed by the end markers, the subformula on line 12 states that a configuration is scanning at least one tape cell and is in atleast one state q from Q, the subformula on line 13 states that for all tape cell positions i, the machine is either scanning i while in exactly one state q, with $q \in Q$, or cell i is either to the left or to the right of the currently scanned cell, and the subformula on line 14 states that when a configuration α is scanning a tape cell j, then, for all $i > j$, $STATE_i^l$ will be true at α. Subformula 15 states that when a configuration α is scanning a tape cell j, then for all $i < j$ $STATE_i^r$ will be true at α. Subformula (13) togheter with (14) and (15) state that a configuration is scanning at most one tape cell, this means that, togheter with (12), a configuration is in exactly one state and scanning exactly one tape cell.

Lemma 5.10. *For all configurations $\alpha \in S$, $\mathcal{K}_T, \alpha \models CONFIG$*

Proof. $\mathcal{K}_T, \alpha \models CONFIG$

$$\begin{aligned}
\text{iff } \mathcal{K}_T, \alpha \models & \bigwedge_{0 \le i \le n^z+1} \bigvee_{a \in \Gamma} \left(SYMBOL_i^a \wedge \bigwedge_{\substack{b \in \Gamma \\ b \neq a}} \neg SYMBOL_i^b \right) \\
& \wedge \left(SYMBOL_0^{[} \wedge SYMBOL_{n^z+1}^{]} \right) \\
& \wedge \bigvee_{0 \le i \le n^z+1} \bigvee_{q \in Q} STATE_i^q \\
& \wedge \bigwedge_{0 \le i \le n^z+1} \bigvee_{q \in Q \cup \{l,r\}} \left(STATE_i^q \wedge \bigwedge_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} \neg STATE_i^p \right) \\
& \wedge \bigwedge_{0 \le i \le n^z} \bigwedge_{q \in Q \cup \{l\}} \left(STATE_i^q \rightarrow STATE_{i+1}^l \right) \\
& \wedge \bigwedge_{1 \le i \le n^z+1} \bigwedge_{q \in Q \cup \{r\}} \left(STATE_i^q \rightarrow STATE_{i-1}^r \right)
\end{aligned}$$

This is a conjunction of multiple subformulas, therefore if we prove that configuration α satisfies every subformula separately, we will also have proven that $\mathcal{K}_T, \alpha \models CONFIG$.

Proof of $\mathcal{K}_T, \alpha \models \bigwedge_{0 \le i \le n^z+1} \bigvee_{a \in \Gamma} \left(SYMBOL_i^a \wedge \bigwedge_{\substack{b \in \Gamma \\ b \neq a}} \neg SYMBOL_i^b \right)$*:*

$$\begin{aligned}
\mathcal{K}_T, \alpha \models & \bigwedge_{0 \le i \le n^z+1} \bigvee_{a \in \Gamma} \left(SYMBOL_i^a \wedge \bigwedge_{\substack{b \in \Gamma \\ b \neq a}} \neg SYMBOL_i^b \right) \\
& \text{iff } \alpha \in V \left(\bigwedge_{0 \le i \le n^z+1} \bigvee_{a \in \Gamma} \left(SYMBOL_i^a \wedge \bigwedge_{\substack{b \in \Gamma \\ b \neq a}} \neg SYMBOL_i^b \right) \right) \qquad \text{(definition } \models\text{)}
\end{aligned}$$

$$\text{iff } \alpha \in \bigcap_{0 \le i \le n^z+1} V\left(\bigvee_{a\in\Gamma}\left(SYMBOL_i^a \wedge \bigwedge_{\substack{b\in\Gamma\\ b\neq a}} \neg SYMBOL_i^b\right)\right) \qquad (K3)$$

$$\text{iff } \alpha \in \bigcap_{0 \le i \le n^z+1} \bigcup_{a\in\Gamma} V\left(SYMBOL_i^a \wedge \bigwedge_{\substack{b\in\Gamma\\ b\neq a}} \neg SYMBOL_i^b\right) \qquad (K4)$$

$$\text{iff for all } i \text{ such that } 0 \le i \le n^z+1 \ : \alpha \in \bigcup_{a\in\Gamma} V\left(SYMBOL_i^a \wedge \bigwedge_{\substack{b\in\Gamma\\ b\neq a}} \neg SYMBOL_i^b\right) \qquad (\text{definition } \textstyle\bigcap)$$

Proving this for an arbitrary i, with $0 \le i \le n^z + 1$:

$$\alpha \in \bigcup_{a\in\Gamma} V\left(SYMBOL_i^a \wedge \bigwedge_{\substack{b\in\Gamma\\ b\neq a}} \neg SYMBOL_i^b\right)$$

$$\text{iff for some } a, \text{ such that } a \in \Gamma : \alpha \in V\left(SYMBOL_i^a \wedge \bigwedge_{\substack{b\in\Gamma\\ b\neq a}} \neg SYMBOL_i^b\right) \qquad (\text{definition } \textstyle\bigcup)$$

Let α be a configuration, and denote the symbol at tape cell i by a, with $a \in \Gamma$. We now need to prove that $\alpha \in V\left(SYMBOL_i^a \wedge \bigwedge_{\substack{b\in\Gamma\\ b\neq a}} \neg SYMBOL_i^b\right)$. By definition, we already know that:

$$\alpha \in V\left(SYMBOL_i^a\right) \text{ is true.} \qquad (16)$$

Since the following equivalences hold:

$$\alpha \in V\left(SYMBOL_i^a \wedge \bigwedge_{\substack{b\in\Gamma\\ b\neq a}} \neg SYMBOL_i^b\right)$$

$$\text{iff } \alpha \in V\left(SYMBOL_i^a \cap \bigwedge_{\substack{b\in\Gamma\\ b\neq a}} \neg SYMBOL_i^b\right) \qquad (K3)$$

$$\text{iff } \alpha \in V\left(SYMBOL_i^a\right) \text{ and } \alpha \in V\left(\bigwedge_{\substack{b\in\Gamma\\ b\neq a}} \neg SYMBOL_i^b\right) \qquad (\text{definition } \cap),$$

we only need to prove that $\alpha \in V\left(\bigwedge_{\substack{b\in\Gamma\\ b\neq a}} \neg SYMBOL_i^b\right)$. This is done as follows:

$$\alpha \in V\left(\bigwedge_{\substack{b\in\Gamma\\ b\neq a}} \neg SYMBOL_i^b\right)$$

$$\text{iff } \alpha \in \bigcap_{\substack{b\in\Gamma\\ b\neq a}} V\left(\neg SYMBOL_i^b\right) \qquad (K3)$$

$$\text{iff for all b, such that } b \in \Gamma \text{ and } b \neq a : \alpha \in V\left(\neg SYMBOL_i^b\right) \qquad (K3).$$

Proving this for an arbitrary $b \in \Gamma$ and $b \neq a$:

$$\begin{aligned} &\alpha \in V\left(\neg SYMBOL_i^b\right) \\ &\text{iff } \alpha \in S - V\left(SYMBOL_i^b\right) \qquad (K2) \end{aligned}$$

$SYMBOL_i^b$ gives the set of all configurations which have symbol b written on work tape cell i. $S - V\left(SYMBOL_i^b\right)$ therefore gives the set of all configurations which do not have symbol b written on tape cell i. Since we assume α to contain symbol a, which is not equal to b, on tape cell i, and definition 4.2 tells us that every work tape cell can only contain one symbol, it follows that:

$$\alpha \in S - V\left(SYMBOL_i^b\right) \text{ is true.}$$

We can now conclude that:

$$\mathcal{K}_T, \alpha \models \bigwedge_{0 \leq i \leq n^z+1} \bigvee_{a \in \Gamma} \left(SYMBOL_i^a \wedge \bigwedge_{\substack{b \in \Gamma \\ b \neq a}} \neg SYMBOL_i^b \right) \text{is true.} \qquad (17)$$

■

Proof of $\mathcal{K}_T, \alpha \models \left(SYMBOL_0^{[} \wedge SYMBOL_{n^z+1}^{]}\right)$*:*

$$\begin{aligned} &\mathcal{K}_T, \alpha \models \left(SYMBOL_0^{[} \wedge SYMBOL_{n^z+1}^{]}\right) && (18) \\ &\quad \text{iff } \alpha \in V\left(SYMBOL_0^{[} \wedge SYMBOL_{n^z+1}^{]}\right) && \text{(definition } \models) \quad (19) \\ &\quad \text{iff } \alpha \in V\left(SYMBOL_0^{[}\right) \cap V\left(SYMBOL_{n^z+1}^{]}\right) && (K3) \quad (20) \end{aligned}$$

$V\left(SYMBOL_0^{[}\right)$ gives the set of all configurations which have the symbol [written on tape cell 0, and $V\left(SYMBOL_{n^z+1}^{]}\right)$ gives the set of all configurations which have the symbol] written on tape cell n^z+1. By definition 4.2 all configurations have symbol [on tape cell 0 and symbol] on tape cell $n^z + 1$, therefore:

$$\mathcal{K}_T, \alpha \models \left(SYMBOL_0^{[} \wedge SYMBOL_{n^z+1}^{]}\right) \text{ is true.} \qquad (21)$$

■

Proof of $\mathcal{K}_T, \alpha \models \bigvee_{0 \leq i \leq n^z+1} \bigvee_{q \in Q} STATE_i^q$*:*

$$\begin{aligned} &\mathcal{K}_T, \alpha \models \bigvee_{0 \leq i \leq n^z+1} \bigvee_{q \in Q} STATE_i^q \\ &\quad \text{iff } \alpha \in V\left(\bigvee_{0 \leq i \leq n^z+1} \bigvee_{q \in Q} STATE_i^q \right) && \text{(definition } \models) \\ &\quad \text{iff } \alpha \in \bigcup_{0 \leq i \leq n^z+1} V\left(\bigvee_{q \in Q} STATE_i^q \right) && (K4) \\ &\quad \text{iff } \alpha \in \bigcup_{0 \leq i \leq n^z+1} \bigcup_{q \in Q} V\left(STATE_i^q\right) && (K4) \\ &\quad \text{iff for some i and some q, such that } 0 \leq i \leq n^z+1 \text{ and } q \in Q : \alpha \in V\left(STATE_i^q\right) && \text{(definition } \bigcup) \end{aligned}$$

$V\left(STATE_i^q\right)$ gives the set of all configurations which are in state q and reading cell i, by definition 4.2 every configuration is in one state and reading one cell, therefore for some state q from Q and some work tape position i, $\alpha \in V\left(STATE_i^q\right)$ holds, and therefore:

$$\mathcal{K}_T, \alpha \models \bigvee_{0 \leq i \leq n^z+1} \bigvee_{q \in Q} STATE_i^q \text{ is true.} \tag{22}$$

■

Proof of $\mathcal{K}_T, \alpha \models \bigwedge_{0 \leq i \leq n^z+1} \bigvee_{q \in Q \cup \{l,r\}} \left(STATE_i^q \wedge \bigwedge_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} \neg STATE_i^p \right)$:

$$\mathcal{K}_T, \alpha \models \bigwedge_{0 \leq i \leq n^z+1} \bigvee_{q \in Q \cup \{l,r\}} \left(STATE_i^q \wedge \bigwedge_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} \neg STATE_i^p \right)$$

$$\text{iff } \alpha \in V\left(\bigwedge_{0 \leq i \leq n^z+1} \bigvee_{q \in Q \cup \{l,r\}} \left(STATE_i^q \wedge \bigwedge_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} \neg STATE_i^p \right) \right) \qquad \text{(definition } \models\text{)}$$

$$\text{iff } \alpha \in \bigcap_{0 \leq i \leq n^z+1} V\left(\bigvee_{q \in Q \cup \{l,r\}} \left(STATE_i^q \wedge \bigwedge_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} \neg STATE_i^p \right) \right) \qquad (K3)$$

$$\text{iff } \alpha \in \bigcap_{0 \leq i \leq n^z+1} \bigcup_{q \in Q \cup \{l,r\}} V\left(STATE_i^q \wedge \bigwedge_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} \neg STATE_i^p \right) \qquad (K4)$$

$$\text{iff } \alpha \in \bigcap_{0 \leq i \leq n^z+1} \bigcup_{q \in Q \cup \{l,r\}} \left(V\left(STATE_i^q\right) \cap V\left(\bigwedge_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} \neg STATE_i^p \right) \right) \qquad (K3)$$

$$\text{iff } \alpha \in \bigcap_{0 \leq i \leq n^z+1} \bigcup_{q \in Q \cup \{l,r\}} \left(V\left(STATE_i^q\right) \cap \bigcap_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} V\left(\neg STATE_i^p\right) \right) \qquad (K3)$$

iff for all i, such that $0 \leq i \leq n^z + 1$, there is some q, with $q \in Q \cup \{l, r\}$, such that:

$$\alpha \in V\left(STATE_i^q\right) \cap \bigcap_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} V\left(\neg STATE_i^p\right)$$

iff for all i, such that $0 \leq i \leq n^z + 1$, there is some q, with $q \in Q \cup \{l, r\}$, such that:

$$\alpha \in V\left(STATE_i^q\right) \text{ and } \alpha \in \bigcap_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} V\left(\neg STATE_i^p\right) \tag{23}$$

We prove the two statements in (23) separately for some arbitrary i, such that $0 \leq i \leq n^z + 1$ and some q, with $q \in Q \cup \{l, r\}$.

$$\alpha \in V\left(STATE_i^q\right) \qquad \left(Split\ (23)\right) \tag{24}$$

Assume α is in some state y, with $y \in Q$, and scanning an arbitrary work tape cell j, then we will need to proof (24) for $i = j$ and $i < j$ and $i > j$.
If $i = j$ then, by definition of $V\left(STATE_i^q\right)$, $\alpha \in V\left(STATE_i^q\right)$ is true, iff $y = q$.
If $i < j$ then the currently scanned cell is to the right of i, and therefore, by the definition of $V\left(STATE_i^r\right)$, $\alpha \in V\left(STATE_i^q\right)$ is true, iff $q = r$.
If $i > j$ then the currently scanned cell is to the left of i, and therefore, by the definition of $V\left(STATE_i^l\right)$, $\alpha \in V\left(STATE_i^q\right)$ is true, iff $q = l$.
therefore:

$$\alpha \in V\left(STATE_i^q\right) \text{ is true. for } \alpha \text{ in some state } q \text{ and scanning any work tape cell i} \tag{25}$$

$$\alpha \in \bigcap_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} V\left(\neg STATE_i^p\right) \qquad \left(Split\ (23)\right)$$

$$\text{iff for all } p, \text{ such that } p \in Q \cup \{l,r\} \text{ and } p \neq q\text{: } \alpha \in V\left(\neg STATE_i^p\right) \qquad \left(\text{definition } \bigcap\right)$$

We prove this for an arbitrary p, such that $p \in Q \cup \{l,r\}$ and $p \neq q$:

$$\alpha \in V\left(\neg STATE_i^p\right)$$

Assume α is in some state y, with $y \in Q \cup \{l,r\}$ and scanning an arbitrary work tape cell j, then there are three options for q: $q \in Q, q = l$ and $q = r$.
If $q \in Q$ then $i = j$ and, by definition (4.2), a configuration is always in exactly one state and scanning exactly one work tape cell, therefore for every $p \in Q \cup \{l,r\}$ and $p \neq q : \alpha \in V\left(\neg STATE_i^p\right)$ is true.
If $q = l$ then $j < i$ and therefore, by definition of $V\left(STATE_i^q\right\}$ and $STATE_i^r\right)$, $\alpha \in V\left(\neg STATE_i^p\right)$ is true for $p \in Q \cup \{l,r\}$ and $p \neq l$.
If $q = r$ then $j > i$ and therefore, by definition of $V\left(STATE_i^q\right\}$ and $STATE_i^l\right)$, $\alpha \in V\left(\neg STATE_i^p\right)$ is true for $p \in Q \cup \{l,r\}$ and $p \neq r$
Therefore:

$$\alpha \in V\left(\neg STATE_i^p\right) \text{ is true.}$$

From this we can conclude:

$$\alpha \in \bigcap_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} V\left(\neg STATE_i^p\right) \text{ is true.} \tag{26}$$

From (25) and (26) we can conclude:

$$\mathcal{K}_T, \alpha \models \bigwedge_{0 \,\leq\, i \,\leq\, n^z + 1} \ \bigvee_{q \in Q \cup \{l,r\}} \left(STATE_i^q \wedge \bigwedge_{\substack{p \in Q \cup \{l,r\} \\ p \neq q}} \neg STATE_i^p \right) \text{ is true.}$$

■

Proof of $\mathcal{K}_T, \alpha \models \bigwedge_{1 \leq i \leq n^z} \bigwedge_{q \in Q \cup \{l\}} \left(STATE_i^q \rightarrow STATE_{i+1}^l \right)$.

$$\mathcal{K}_T, \alpha \models \bigwedge_{0 \leq i \leq n^z} \bigwedge_{q \in Q \cup \{l\}} \left(STATE_i^q \rightarrow STATE_{i+1}^l \right)$$

$$\text{iff } \alpha \in V\left(\bigwedge_{0 \leq i \leq n^z} \bigwedge_{q \in Q \cup \{l\}} \left(STATE_i^q \rightarrow STATE_{i+1}^l \right) \right) \qquad (\text{definition } \models)$$

$$\text{iff } \alpha \in \bigcap_{0 \leq i \leq n^z} V\left(\bigwedge_{q \in Q \cup \{l\}} \left(STATE_i^q \rightarrow STATE_{i+1}^l \right) \right) \qquad (K3)$$

$$\text{iff } \alpha \in \bigcap_{0 \le i \le n^z} \bigcap_{q \in Q \cup \{l\}} V\left(STATE_i^q \rightarrow STATE_{i+1}^l\right) \qquad (K3)$$

$$\text{iff for all i, such that } 0 \le i \le n^z : \alpha \in \bigcap_{q \in Q \cup \{l\}} V\left(STATE_i^q \rightarrow STATE_{i+1}^l\right) \qquad (\text{definition } \bigcap)$$

Proving this for any arbitrary work tape cell i, such that $0 \le i \le n^z$:

$$\alpha \in \bigcap_{q \in Q \cup \{l\}} V\left(STATE_i^q \rightarrow STATE_{i+1}^l\right) \qquad (27)$$

$$\text{iff for all q, such that } q \in Q \cup \{l\} : \alpha \in V\left(STATE_i^q \rightarrow STATE_{i+1}^l\right) \qquad (\text{definition } \bigcap)$$

Proving this for an arbitrary state q, such that $q \in Q$ or $q = l$:

$$\alpha \in V\left(STATE_i^q \rightarrow STATE_{i+1}^l\right) \qquad (28)$$

Assume $\alpha \in V\left(STATE_i^q\right)$, then: $\alpha \in V\left(STATE_i^q \rightarrow STATE_{i+1}^l\right)$ iff: $\alpha \in V\left(STATE_{i+1}^l\right)$

$V\left(STATE_{i+1}^l\right)$ gives the set of all configurations for which the currently scanned work tape cell is to the left of work tape cell $i + 1$, by our assumption α is either currently scanning cell i, for $q \in Q$, or scanning a cell to the left of i, for $q = l$, Therefore:

$$\alpha \in V\left(STATE_{i+1}^l\right) \text{ is true, iff } \alpha \in V\left(STATE_i^q\right)$$

Therefore:

$$\alpha \in V\left(STATE_i^q \rightarrow STATE_{i+1}^l\right) \text{ is true.}$$

We have proven (28) for an arbitrary q, such that $q \in Q$ or $q = l$, this means that it will hold for all q, such that $q \in Q \cup \{l\}$, therefore:

$$\alpha \in \bigcap_{q \in Q \cup \{l\}} V\left(STATE_i^q \rightarrow STATE_{i+1}^l\right) \text{ is true.}$$

This proves (27) for an arbitrary i, such that $0 \le i \le n^z$, this means it will hold for all i, such that $0 \le i \le n^z$, therefore:

$$\alpha \in \bigcap_{0 \le i \le n^z} \bigcap_{q \in Q \cup \{l\}} V\left(STATE_i^q \rightarrow STATE_{i+1}^l\right) \text{ is true.}$$

this proves that:

$$\mathcal{K}_T, \alpha \models \bigwedge_{0 \le i \le n^z} \bigwedge_{q \in Q \cup \{l\}} \left(STATE_i^q \rightarrow STATE_{i+1}^l\right) \text{ holds.}$$

■

Proof of $\mathcal{K}_T, \alpha \models \bigwedge_{1 \le i \le n^z+1} \bigwedge_{q \in Q \cup \{r\}} \left(STATE_i^q \rightarrow STATE_{i-1}^r\right)$.

$$\mathcal{K}_T, \alpha \models \bigwedge_{1 \le i \le n^z+1} \bigwedge_{q \in Q \cup \{r\}} \left(STATE_i^q \rightarrow STATE_{i-1}^r\right)$$

$$\text{iff } \alpha \in V\left(\bigwedge_{1 \le i \le n^z+1} \bigwedge_{q \in Q \cup \{r\}} \left(STATE_i^q \rightarrow STATE_{i-1}^r\right)\right) \qquad (\text{definition } \models)$$

$$\text{iff } \alpha \in \bigcap_{1 \leq i \leq n^z+1} V\left(\bigwedge_{q \in Q \cup \{r\}} \left(STATE_i^q \rightarrow STATE_{i-1}^r\right)\right) \qquad (K3)$$

$$\text{iff } \alpha \in \bigcap_{1 \leq i \leq n^z+1} \bigcap_{q \in Q \cup \{r\}} V\left(STATE_i^q \rightarrow STATE_{i-1}^r\right) \qquad (K3)$$

$$\text{iff for all i, such that } 0 \leq i \leq n^z : \alpha \in \bigcap_{q \in Q \cup \{r\}} V\left(STATE_i^q \rightarrow STATE_{i-1}^r\right) \qquad (\text{definition } \bigcap)$$

Proving this for any arbitrary work tape cell i, such that $1 \leq i \leq n^z + 1$:

$$\alpha \in \bigcap_{q \in Q \cup \{r\}} V\left(STATE_i^q \rightarrow STATE_{i-1}^r\right) \qquad (29)$$

$$\text{iff for all q, such that } q \in Q \cup \{r\} : \alpha \in V\left(STATE_i^q \rightarrow STATE_{i-1}^r\right) \qquad (\text{definition } \bigcap)$$

Proving this for an arbitrary state q, such that $q \in Q$ or $q = r$:

$$\alpha \in V\left(STATE_i^q \rightarrow STATE_{i-1}^r\right) \qquad (30)$$

Assume $\alpha \in V\left(STATE_i^q\right)$, then: $\alpha \in V\left(STATE_i^q \rightarrow STATE_{i-1}^r\right)$ iff: $\alpha \in V\left(STATE_{i-1}^r\right)$

$V\left(STATE_{i-1}^r\right)$ gives the set of all configurations for which the currently scanned work tape cell is to the right of work tape cell $i-1$, by our assumption α is either currently scanning cell i, for $q \in Q$, or scanning a cell to the right of i, for $q = r$, Therefore:

$$\alpha \in V\left(STATE_{i-1}^r\right) \text{ is true, iff } \alpha \in V\left(STATE_i^q\right)$$

Therefore:

$$\alpha \in V\left(STATE_i^q \rightarrow STATE_{i-1}^r\right) \text{ is true.}$$

We have proven (30) for an arbitrary q, such that $q \in Q$ or $q = r$, this means that it will hold for all q, such that $q \in Q \cup \{r\}$, therefore:

$$\alpha \in \bigcap_{q \in Q \cup \{r\}} V\left(STATE_i^q \rightarrow STATE_{i-1}^r\right) \text{ is true.}$$

This proves (29) for an arbitrary i, such that $1 \leq i \leq n^z + 1$, this means it will hold for all i, such that $1 \leq i \leq n^z + 1$, therefore:

$$\alpha \in \bigcap_{1 \leq i \leq n^z+1} \bigcap_{q \in Q \cup \{r\}} V\left(STATE_i^q \rightarrow STATE_{i-1}^r\right) \text{ is true.}$$

this proves that:

$$\mathcal{K}_T, \alpha \models \bigwedge_{1 \leq i \leq n^z+1} \bigwedge_{q \in Q \cup \{r\}} \left(STATE_i^q \rightarrow STATE_{i-1}^r\right) \text{ holds.}$$

■

We have now proven that every subformula in the conjunction of CONFIG holds, therefore we have proven that:

$$\mathcal{K}_T, \alpha \models CONFIG. \qquad (31)$$

□

5.0.4 MOVE

Definition 5.11. *The formula MOVE=*

$$\bigwedge_{0 \le i \le n^z+1} \big((STATE_i^l \vee STATE_i^r) \rightarrow \bigwedge_{a \in \Gamma} (SYMBOL_i^a \rightarrow [NEXT] SYMBOL_i^a)\big) \tag{32}$$

$$\wedge \bigwedge_{0 \le i \le n^z+1} \bigwedge_{\substack{a \in \Gamma \\ q \in Q}} \Bigg((SYMBOL_i^a \wedge STATE_i^q) \rightarrow \tag{33}$$

$$\Big(\bigwedge_{(p,b,d) \in \Delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p) \Big) \tag{34}$$

$$\wedge \quad [NEXT] \big(\bigvee_{(p,b,d) \in \Delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p) \big) \Bigg) \tag{35}$$

The function MOVE is true at all configurations α if the configurations reachable from α by $NEXT$ are exactly the same configurations as the configurations reachable from α by δ in one step. Equation 32 says that when a cell i is not currently being scanned then at the successor states the symbol at i will not be overwritten. Equations 33, 34 and 35 says that if, for all tape cells i, states q in Q and symbols a in Γ, the machine is in state q while scanning the symbol a on tape cell i (equation 33), then for every element in $\delta(q, a)$ there is at least one successor state that corresponds to the successor configuration of the ATM (equation 34), and all successor states will represent a successor configuration (equation 35). By combining the equations 34 and 35 we make sure that a state has exactly the same successor states as indicated by δ.

Lemma 5.12. *For all configurations $\alpha \in S, \mathcal{K}_T, \alpha \models MOVE$*

Proof.

$$\mathcal{K}_T, \alpha \models MOVE$$

$$\begin{aligned}
\text{iff } \mathcal{K}_T, \alpha \models & \bigwedge_{0 \le i \le n^z+1} \big((STATE_i^l \vee STATE_i^r) \rightarrow \bigwedge_{a \in \Gamma} (SYMBOL_i^a \rightarrow [NEXT] SYMBOL_i^a)\big) \\
& \wedge \bigwedge_{0 \le i \le n^z+1} \bigwedge_{\substack{a \in \Gamma \\ q \in Q}} \Bigg((SYMBOL_i^a \wedge STATE_i^q) \rightarrow \\
& \Big(\bigwedge_{(p,b,d) \in \Delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p) \Big) \\
& \wedge \quad [NEXT] \big(\bigvee_{(p,b,d) \in \Delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p) \big) \Bigg)
\end{aligned} \qquad \text{(definition MOVE)}$$

(36)

MOVE is a conjunction of two different statements, therefore it will suffice to prove them separately.

$$\mathcal{K}_T, \alpha \models \bigwedge_{0 \le i \le n^z+1} ((STATE_i^l \vee STATE_i^r) \rightarrow \bigwedge_{a \in \Gamma} (SYMBOL_i^a \rightarrow [NEXT] SYMBOL_i^a)) \qquad (Split\ (36))$$

$$\text{iff } \alpha \in V \Bigg(\bigwedge_{0 \le i \le n^z+1} ((STATE_i^l \vee STATE_i^r) \rightarrow \bigwedge_{a \in \Gamma} (SYMBOL_i^a \rightarrow [NEXT] SYMBOL_i^a)) \Bigg) \qquad (\text{definition } \models)$$

$$\text{iff } \alpha \in \bigcap_{0 \le i \le n^z+1} V\left((STATE_i^l \vee STATE_i^r) \rightarrow \bigwedge_{a\in\Gamma}(SYMBOL_i^a \rightarrow [NEXT]SYMBOL_i^a)\right) \qquad (K3)$$

iff for all i, such that $0 \le i \le n^z + 1$:

$$\alpha \in V\left((STATE_i^l \vee STATE_i^r) \rightarrow \bigwedge_{a\in\Gamma}(SYMBOL_i^a \rightarrow [NEXT]SYMBOL_i^a)\right) \qquad (\text{definition } \textstyle\bigcap)$$

We prove this for an arbitrary i, such that $0 \le i \le n^z + 1$:

$$\alpha \in V\left((STATE_i^l \vee STATE_i^r) \rightarrow \bigwedge_{a\in\Gamma}(SYMBOL_i^a \rightarrow [NEXT]SYMBOL_i^a)\right)$$

Assume $\alpha \in V\left(STATE_i^l \vee STATE_i^r\right)$, then:

$$\alpha \in V\left(\bigwedge_{a\in\Gamma}(SYMBOL_i^a \rightarrow [NEXT]SYMBOL_i^a)\right)$$

$$\text{iff } \alpha \in V\left(\bigcap_{a\in\Gamma}(SYMBOL_i^a \rightarrow [NEXT]SYMBOL_i^a)\right) \qquad (K3)$$

$$\text{iff for all a, such that } a \in \Gamma\text{: } \alpha \in V\left((SYMBOL_i^a \rightarrow [NEXT]SYMBOL_i^a)\right) \qquad (\text{definition } \textstyle\bigcap)$$

Assume $\alpha \in V\left(SYMBOL_i^a\right)$, then:

$$\alpha \in V\left([NEXT]SYMBOL_i^a\right)$$

$$\text{iff } \alpha \in \left\{u \mid R_{NEXT}(u) \subseteq V\left(SYMBOL_i^a\right)\right\} \qquad (K7)$$

$$\text{iff } R_{NEXT}(\alpha) \subseteq V\left(SYMBOL_i^a\right) \qquad (\text{definition } \in)$$

$V\left(SYMBOL_i^a\right)$ gives the set of all configurations which have symbol a written on work tape position i, $R_{NEXT}(\alpha)$ is a subset of $V\left(SYMBOL_i^a\right)$ iff all successor configurations of α also have symbol a written on work tape position i. By our assumption of $\alpha \in V\left(STATE_i^l \vee STATE_i^r\right)$, α is currently scanning a work tape position to the left or to the right of i, and therefore it is, by definition (4.2), not scanning cell i. this means, by definition 4.4, that the contents at work tape cell i will not be overwritten, and so:

$R_{NEXT}(\alpha) \subseteq V\left(SYMBOL_i^a\right)$ holds, for configuration α.

We can now conclude that:

$$\mathcal{K}_T, \alpha \models \bigwedge_{0 \le i \le n^z+1}((STATE_i^l \vee STATE_i^r) \rightarrow \bigwedge_{a\in\Gamma}(SYMBOL_i^a \rightarrow [NEXT]SYMBOL_i^a)) \text{ is true.} \qquad (37)$$

$$\mathcal{K}_T, \alpha \models \bigwedge_{0 \le i \le n^z+1} \bigwedge_{\substack{a\in\Gamma \\ q\in Q}} \Bigg((SYMBOL_i^a \wedge STATE_i^q) \rightarrow$$

$$\Big(\bigwedge_{(p,b,d)\in\delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p)\Big) \qquad (Split\ (36))$$

$$\wedge \quad [NEXT]\Big(\bigvee_{(p,b,d)\in\delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p)\Big)\Bigg)$$

iff, for all i such that $0 \leq i \leq n^z + 1$:

$$\mathcal{K}_T, \alpha \models \bigwedge_{\substack{a \in \Gamma \\ q \in Q}} \Bigg((SYMBOL_i^a \wedge STATE_i^q) \rightarrow$$
$$\Big(\bigwedge_{(p,b,d) \in \delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p) \Big) \qquad \text{(definition } \textstyle\bigwedge\text{)}$$
$$\wedge \quad [NEXT] \Big(\bigvee_{(p,b,d) \in \delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p) \Big) \Bigg)$$

We prove this for an arbitrary i, such that $0 \leq i \leq n^z + 1$:

$$\mathcal{K}_T, \alpha \models \bigwedge_{\substack{a \in \Gamma \\ q \in Q}} \Bigg((SYMBOL_i^a \wedge STATE_i^q) \rightarrow$$
$$\Big(\bigwedge_{(p,b,d) \in \delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p) \Big)$$
$$\wedge \quad [NEXT] \Big(\bigvee_{(p,b,d) \in \delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p) \Big) \Bigg)$$

iff, for all a and q, such that $a \in \Gamma$ and $q \in Q$:

$$\mathcal{K}_T, \alpha \models \Bigg((SYMBOL_i^a \wedge STATE_i^q) \rightarrow$$
$$\Big(\bigwedge_{(p,b,d) \in \delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p) \Big) \text{ (definition } \textstyle\bigwedge\text{)}$$
$$\wedge \quad [NEXT] \Big(\bigvee_{(p,b,d) \in \delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p) \Big) \Bigg)$$

We prove this for an arbitrary a and q, such that $a \in \Gamma$ and $q \in Q$:

$$\mathcal{K}_T, \alpha \models \Bigg((SYMBOL_i^a \wedge STATE_i^q) \rightarrow$$
$$\Big(\bigwedge_{(p,b,d) \in \delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p) \Big)$$
$$\wedge \quad [NEXT] \Big(\bigvee_{(p,b,d) \in \delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p) \Big) \Bigg)$$

iff

$$\alpha \in V \Bigg((SYMBOL_i^a \wedge STATE_i^q) \rightarrow$$
$$\Big(\bigwedge_{(p,b,d) \in \delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p) \Big) \qquad \text{(definition } \models\text{)}$$
$$\wedge \quad [NEXT] \Big(\bigvee_{(p,b,d) \in \delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p) \Big) \Bigg)$$

Assume $\alpha \in V\left(SYMBOL_i^a \wedge STATE_i^q\right)$, then:

$$\alpha \in V\Bigg(\Big(\bigwedge_{(p,b,d)\in\delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p)\Big) \wedge \quad [NEXT]\Big(\bigvee_{(p,b,d)\in\delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p)\Big)\Bigg)$$

iff

$$\begin{aligned} &\alpha \in V\left(\bigwedge_{(p,b,d)\in\delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p)\right) \\ &\cap \quad V\left([NEXT]\Big(\bigvee_{(p,b,d)\in\delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p)\Big)\right) \end{aligned} \tag{$K3$}$$

iff

$$\begin{aligned} &\alpha \in V\left(\bigwedge_{(p,b,d)\in\delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p)\right) && \text{(definition } \cap\text{)} \\ \text{and} \quad &\alpha \in V\left([NEXT]\Big(\bigvee_{(p,b,d)\in\delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p)\Big)\right) \end{aligned} \tag{38}$$

We prove the two statements of (38) separately:

$$\begin{aligned} &\alpha \in V\left(\bigwedge_{(p,b,d)\in\delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p)\right) && \left(Split\ (38)\right) \\ \text{iff } &\alpha \in \bigcap_{(p,b,d)\in\delta(q,a)} V\left(< NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p)\right) && (K3) \\ \text{iff for all } &(p,b,d), \text{ such that} (p,b,d) \in \delta(q,a) : \alpha \in V\left(< NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p)\right) && \text{(definition } \textstyle\bigcap\text{)} \end{aligned} \tag{39}$$

We prove (39) for an arbitrary (p,b,d), such that $(p,b,d) \in \delta(q,a)$:

$$\begin{aligned} &\alpha \in V\left(< NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p)\right) \\ \text{iff } &\alpha \in V\left(\neg[NEXT]\neg\left(SYMBOL_i^b \wedge STATE_{i+d}^p\right)\right) && \text{(definition } \langle NEXT\rangle\text{)} \\ \text{iff } &\alpha \in \left\{w \mid R_{NEXT}(w) \notin V\left(\neg\left(SYMBOL_i^b \wedge STATE_{i+d}^p\right)\right)\right\} && (K7) \\ \text{iff } &\alpha \in \left\{w \mid R_{NEXT}(w) \notin S - V\left(SYMBOL_i^b \wedge STATE_{i+d}^p\right)\right\} && (K2) \\ \text{iff } &\alpha \in \left\{w \mid R_{NEXT}(w) \notin S - \left(V\left(SYMBOL_i^b\right) \cap V\left(STATE_{i+d}^p\right)\right)\right\} && (K3) \\ \text{iff } &R_{NEXT}(\alpha) \notin S - \left(V\left(SYMBOL_i^b\right) \cap V\left(STATE_{i+d}^p\right)\right) && \text{(definition } \in\text{)} \\ \text{iff } &R_{NEXT}(\alpha) \cap V\left(SYMBOL_i^b\right) \cap V\left(STATE_{i+d}^p\right) \neq \emptyset && \text{(definition } \subseteq\text{)} \end{aligned}$$

For $R_{NEXT}(\alpha) \cap V\left(SYMBOL_i^b\right) \cap V\left(STATE_{i+d}^p\right) \neq \emptyset$ to be true, at least one successor of α needs to be in state p, scanning work tape cell $i+d$, and have symbol b written on work tape cell i. Definition (4.4) states that all successor configurations of a state α are decided by the next move function δ, therefore, iff $(p, b, d) \in \delta(q, a)$ then there is at least one successor configuration which is in state p while scanning work tape cell $i+d$ and which has symbol b written on work tape cell i. Therefore:

$$R_{NEXT}(\alpha) \cap V\left(SYMBOL_i^b\right) \cap V\left(STATE_{i+d}^p\right) \neq \emptyset \text{ is true.}$$

And so we can conclude that:

$$\alpha \in V\left(\bigwedge_{(p,b,d)\in\delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p)\right) \text{ is true.} \tag{40}$$

$$\alpha \in V\left([NEXT]\Big(\bigvee_{(p,b,d)\in\delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p)\Big)\right) \qquad \big(Split\ (38)\big)$$

$$\text{iff } \alpha \in \left\{w \mid R_{NEXT}(w) \subseteq V\left(\bigvee_{(p,b,d)\in\delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p)\right)\right\} \qquad (K7)$$

$$\text{iff } R_{NEXT}(\alpha) \subseteq V\left(\bigvee_{(p,b,d)\in\delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p)\right) \qquad (\text{definition } \in)$$

$$\text{iff } R_{NEXT}(\alpha) \subseteq \bigcup_{(p,b,d)\in\delta(q,a)} V\left(SYMBOL_i^b \wedge STATE_{i+d}^p\right) \qquad (K7)$$

$$\text{iff } R_{NEXT}(\alpha) \subseteq \bigcup_{(p,b,d)\in\delta(q,a)} \left(V\left(SYMBOL_i^b\right) \cap V\left(STATE_{i+d}^p\right)\right) \qquad (K3)$$

Assume α is in state q and currently scanning symbol a. We need to prove that for all configurations w, such that $w \in R_{NEXT}(\alpha)$, there is some triple $(p, b, d) \in \delta(q, a)$ for which:

$$w \in V\left(SYMBOL_i^b\right) \cap V\left(STATE_{i+d}^p\right) \qquad (K3)$$

Definition (4.4) tells us that all successor configurations of a state α are decided by the next move function δ. Since $w \in R_{NEXT}(\alpha)$, this means that it is a successor of α and therefore there is a corresponding triple (p, b, d) in $\delta(q, a)$ which states that w is in state p, scanning work tape cell $i+d$ and has symbol b written on work tape cell i, therefore:

$$w \in V\left(SYMBOL_i^b\right) \cap V\left(STATE_{i+d}^p\right) \text{ is true.}$$

From this we can conclude that:

$$\alpha \in V\left([NEXT]\Big(\bigvee_{(p,b,d)\in\delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p)\Big)\right) \text{is true.} \tag{41}$$

By (40) and (41) we can conclude

$$\begin{aligned} \mathcal{K}_T, \alpha \models \bigwedge_{0 \le i \le n^z+1} \bigwedge_{\substack{a\in\Gamma \\ q\in Q}} \Bigg(& (SYMBOL_i^a \wedge STATE_i^q) \rightarrow \\ & \Big(\bigwedge_{(p,b,d)\in\delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p)\Big) \\ \wedge \quad & [NEXT]\Big(\bigvee_{(p,b,d)\in\delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p)\Big)\Bigg) \text{ is true.} \end{aligned} \tag{42}$$

By (37) and (42), we can conclude that:

$$\text{iff } \mathcal{K}_T, \alpha \models \bigwedge_{0 \le i \le n^z+1} \big((STATE_i^l \vee STATE_i^r) \rightarrow \bigwedge_{a \in \Gamma} (SYMBOL_i^a \rightarrow [NEXT] SYMBOL_i^a)\big)$$

$$\wedge \bigwedge_{0 \le i \le n^z+1} \bigwedge_{\substack{a \in \Gamma \\ q \in Q}} \Bigg((SYMBOL_i^a \wedge STATE_i^q) \rightarrow$$

$$\Big(\bigwedge_{(p,b,d) \in \Delta(q,a)} < NEXT > (SYMBOL_i^b \wedge STATE_{i+d}^p) \Big)$$

$$\wedge \quad [NEXT] \big(\bigvee_{(p,b,d) \in \Delta(q,a)} (SYMBOL_i^b \wedge STATE_{i+d}^p) \big) \Bigg) \text{ is true.}$$

And therefore it is true that:

$$\mathcal{K}_T, \alpha \models MOVE$$

□

5.0.5 ACCEPTANCE

Definition 5.13. *The formula ACCEPTANCE=*

$$\bigwedge_{0 \le i \le n^z+1} \bigwedge_{q \in E} \big(STATE_i^q \rightarrow (ACCEPT \leftrightarrow \langle NEXT \rangle ACCEPT)\big) \tag{43}$$

$$\wedge \bigwedge_{0 \le i \le n^z+1} \bigwedge_{q \in U} \big(STATE_i^q \rightarrow (ACCEPT \leftrightarrow [NEXT] ACCEPT)\big) \tag{44}$$

The formula ACCEPTANCE is the formula representing the acceptance conditions of an ATM. In here the sets E and U are the sets:

$$\begin{aligned} U &= \{q \mid \text{ with } q \in Q, \text{ and } g(q) = universal\} \\ E &= \{q \mid \text{ with } q \in Q, \text{ and } g(q) = existential\} \end{aligned}$$

Therefore they are the sets representing the existential and universal states. Equation 43 says that an existential configuration is accepting iff atleast one of its successors is accepting. Equation 44 says that an universal configurations is accepting iff all of its successors are accepting

Lemma 5.14. *For all configurations $\alpha \in S, \mathcal{K}_T, \alpha \models ACCEPTANCE$*

Proof. $\mathcal{K}_T, \alpha \models ACCEPTANCE$
iff $\mathcal{K}_T, \alpha \models$

$$\bigwedge_{0 \le i \le n^z+1} \bigwedge_{q \in E} \big(STATE_i^q \rightarrow (ACCEPT \leftrightarrow \langle NEXT \rangle ACCEPT)\big)$$

$$\wedge \bigwedge_{0 \le i \le n^z+1} \bigwedge_{q \in U} \big(STATE_i^q \rightarrow (ACCEPT \leftrightarrow [NEXT] ACCEPT)\big) \qquad \text{(definition ACCEPTANCE)}$$

iff

$$\mathcal{K}_T, \alpha \models \bigwedge_{0 \le i \le n^z+1} \bigwedge_{q \in E} \big(STATE_i^q \rightarrow (ACCEPT \leftrightarrow \langle NEXT \rangle ACCEPT)\big) \tag{45}$$

$$\text{and } \mathcal{K}_T, \alpha \models \bigwedge_{0 \le i \le n^z+1} \bigwedge_{q \in U} \big(STATE_i^q \rightarrow (ACCEPT \leftrightarrow [NEXT] ACCEPT)\big) \tag{46}$$

We can proof equation 45 and 46 separately, starting with 45:

$$\mathcal{K}_T, \alpha \models \bigwedge_{0 \le i \le n^z+1} \bigwedge_{q \in E} (STATE_i^q \rightarrow (ACCEPT \leftrightarrow \langle NEXT \rangle ACCEPT))$$

$$\text{iff } \alpha \in V\left(\bigwedge_{0 \le i \le n^z+1} \bigwedge_{q \in E} \left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow \langle NEXT \rangle ACCEPT)\right)\right) \qquad \text{(definition } \models\text{)}$$

$$\text{iff } \alpha \in \bigcap_{0 \le i \le n^z+1} V\left(\bigwedge_{q \in E} \left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow \langle NEXT \rangle ACCEPT)\right)\right) \qquad (K3)$$

$$\text{iff } \alpha \in \bigcap_{0 \le i \le n^z+1} \bigcap_{q \in E} V\left(\left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow \langle NEXT \rangle ACCEPT)\right)\right) \qquad (K3)$$

$$\text{iff for all i such that } 0 \le i \le n^z+1 : \alpha \in \bigcap_{q \in E} V\left(\left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow \langle NEXT \rangle ACCEPT)\right)\right)$$

Proving for an arbitrary i, such that $0 \le i \le n^z + 1$:

$$\alpha \in \bigcap_{q \in E} V\left(\left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow \langle NEXT \rangle ACCEPT)\right)\right) \qquad \text{(proof for i)}$$

$$\text{iff for all q such that } q \in E : \alpha \in V\left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow \langle NEXT \rangle ACCEPT)\right)$$

Proving for an arbitrary q, such that $q \in E$:

$$\alpha \in V\left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow \langle NEXT \rangle ACCEPT)\right) \qquad \text{(proof for q)}$$

To prove the above, assume $\alpha \in V\left(STATE_i^q\right)$. We then have to show that $\alpha \in V\left(ACCEPT \leftrightarrow \langle NEXT \rangle ACCEPT\right)$, which by definition of $\leftrightarrow$ is equivalent to proving: $\alpha \in V\left(ACCEPT\right)$ iff $\alpha \in V\left(\langle NEXT \rangle ACCEPT\right)$.
We have:

$$\begin{aligned}
&\alpha \in V\left(ACCEPT\right) \\
&\text{iff } \alpha \in \{u \mid u \in S, l^*(u) = 1\} && \text{(definition } V\left(ACCEPT\right)\text{)} \\
&\text{iff } l^*(\alpha) = 1 && \text{(definition } \in\text{)} \\
&\text{iff } \bigvee_{w \in R_{NEXT}(\alpha)} l^*(w) = 1 && \text{(definition } l^*(\alpha)\text{)} \\
&\text{iff } \bigvee_{w \in R_{NEXT}(\alpha)} w \in V\left(ACCEPT\right) && \text{(definition } V\left(ACCEPT\right)\text{)} \\
&\text{iff } \bigvee_{w \in R_{NEXT}(\alpha)} w \notin V\left(\neg ACCEPT\right) && \text{(negation)} \\
&\text{iff } R_{NEXT}(\alpha) \not\subseteq V\left(\neg ACCEPT\right) && \text{(definition } \subseteq\text{)} \\
&\text{iff } \alpha \in \{u \mid R_{NEXT}(u) \not\subseteq V\left(\neg ACCEPT\right)\} && \text{(definition } \in\text{)} \\
&\text{iff } \alpha \in V\left(\neg [NEXT] \neg ACCEPT\right) && (K7) \\
&\text{iff } \alpha \in \langle NEXT \rangle ACCEPT && \text{(definition } \langle\rangle\text{)}
\end{aligned}$$

therefore we can now conclude that:

$$\mathcal{K}_T, \alpha \models \bigwedge_{0 \le i \le n^z+1} \bigwedge_{q \in E} (STATE_i^q \rightarrow (ACCEPT \leftrightarrow \langle NEXT \rangle ACCEPT)) \text{is true.} \qquad (47)$$

$$\mathcal{K}_T, \alpha \models \bigwedge_{0 \le i \le n^z+1} \bigwedge_{q \in U} \left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow [NEXT] ACCEPT)\right)$$

$$\text{iff } \alpha \in V\left(\bigwedge_{0 \le i \le n^z+1} \bigwedge_{q \in U} \left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow [NEXT] ACCEPT)\right)\right) \qquad \text{(definition } \models\text{)}$$

$$\text{iff } \alpha \in \bigcap_{0 \le i \le n^z+1} V\left(\bigwedge_{q \in U}\left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow [NEXT]ACCEPT)\right)\right) \qquad (K3)$$

$$\text{iff } \alpha \in \bigcap_{0 \le i \le n^z+1} \bigcap_{q \in U} V\left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow [NEXT]ACCEPT)\right) \qquad (K3)$$

$$\text{iff for all i, such that: } 0 \le i \le n^z+1 \ : \alpha \in \bigcap_{q \in U} V\left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow [NEXT]ACCEPT)\right) \quad (\text{definition } \textstyle\bigcap) \qquad (48)$$

We prove (48) for an arbitrary i, such that $0 \le i \le n^z + 1$:

$$\alpha \in \bigcap_{q \in U} V\left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow [NEXT]ACCEPT)\right) \qquad (49)$$

$$\text{iff for all q, such that:} q \in U : \alpha \in V\left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow [NEXT]ACCEPT)\right) \quad (\text{definition } \textstyle\bigcap) \qquad (50)$$

We prove (50) for an arbitrary q, such that $q \in U$

$$\alpha \in V\left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow [NEXT]ACCEPT)\right)$$

To prove the above, assume $\alpha \in V\left(STATE_i^q\right)$. We then have to show that $\alpha \in V\,(ACCEPT \leftrightarrow [NEXT]ACCEPT)$, which by definition of $\leftrightarrow$ is equivalent to proving: $\alpha \in V\,(ACCEPT)$ iff $\alpha \in V\,([NEXT]ACCEPT)$.
We have:

$$\alpha \in V\,(ACCEPT)$$

$$\text{iff } \alpha \in \{u \mid u \in S, l^*(u) = 1\} \qquad (\text{definition } V\,(ACCEPT))$$

$$\text{iff } l^*(\alpha) = 1 \qquad (\text{definition } \in)$$

$$\text{iff } \bigwedge_{w \in R_{NEXT}(\alpha)} l^*(w) = 1 \qquad (\text{definition } l^*(\alpha))$$

$$\text{iff } \bigwedge_{w \in R_{NEXT}(\alpha)} w \in V\,(ACCEPT) \qquad (\text{definition } V\,(ACCEPT))$$

$$\text{iff } R_{NEXT}(\alpha) \subseteq V\,(ACCEPT) \qquad (\text{definition } \subseteq)$$

$$\text{iff } \alpha \in \{u \mid R_{NEXT}(u) \subseteq V\,(ACCEPT)\} \qquad (\text{definition } \in)$$

$$\text{iff } \alpha \in V\,([NEXT]ACCEPT) \qquad (K7)$$

therefore we can now conclude that:

$$\mathcal{K}_T, \alpha \models \bigwedge_{0 \le i \le n^z+1} \bigwedge_{q \in U}\left(STATE_i^q \rightarrow (ACCEPT \leftrightarrow [NEXT]ACCEPT)\right) \qquad (51)$$

By (47) and (51) we have proven both statements of ACCEPTANCE, therefore:

$$\mathcal{K}_T, \alpha \models ACCEPTANCE \text{ is true.} \qquad (52)$$

□

5.0.6 The Final Step

Now that we have proven lemmas: 5.10, 5.14, 5.12 we can now prove lemma 5.8:

Proof of lemma 5.8.

$$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models [NEXT^*](CONFIG \wedge MOVE \wedge ACCEPTANCE)$$
$$\text{iff } \sigma_{\mathcal{M}}(x) \in V\,([NEXT^*](CONFIG \wedge MOVE \wedge ACCEPTANCE)) \qquad (\text{definition } \models)$$

$$\text{iff } \sigma_{\mathcal{M}}(x) \in \{w \mid R_{NEXT^*}(w) \subseteq V((CONFIG \wedge MOVE \wedge ACCEPTANCE))\} \qquad (K7)$$
$$\text{iff } R_{NEXT^*}(\sigma_{\mathcal{M}}(x)) \subseteq V((CONFIG \wedge MOVE \wedge ACCEPTANCE)) \qquad (\text{definition } \in)$$

$R_{NEXT^*}(\sigma_{\mathcal{M}}(x))$ gives the set of all configurations which are reachable by $\sigma_{\mathcal{M}}(x)$ in 0 or more steps (definitions of the iteration operator and $NEXT$) By our definition of S in lemma (5.5) we get that all configurations reachable by $\sigma_{\mathcal{M}}(x)$ are in S, therefore $R_{NEXT^*}(\sigma_{\mathcal{M}}(x))$ is some subset of S:

$$R_{NEXT^*}(\sigma_{\mathcal{M}}(x)) \subseteq S \text{ is true.} \tag{53}$$

$V((CONFIG \wedge MOVE \wedge ACCEPTANCE))$ gives the set of configurations α for which:
$\mathcal{K}_T, \alpha \models (CONFIG \wedge MOVE \wedge ACCEPTANCE)$ holds. We have proven with lemmas (5.10), (5.12) and (5.14) that:

$$\begin{aligned} &\mathcal{K}_T, \alpha \models (CONFIG \wedge MOVE \wedge ACCEPTANCE) \text{ is true, for all } \alpha \in S \\ &\text{therefore: } V((CONFIG \wedge MOVE \wedge ACCEPTANCE)) = S \end{aligned} \tag{54}$$

By (53) and (54) we get:

$$R_{NEXT^*}(\sigma_{\mathcal{M}}(x)) \subseteq V(CONFIG \wedge MOVE \wedge ACCEPTANCE)$$

And so we can conclude:

$$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models [NEXT^*](CONFIG \wedge MOVE \wedge ACCEPTANCE)$$

□

We can now show that theorem 5.2 holds:

Proof of theorem 5.2. We have proven that, for all PSST ATMs $\mathcal{M}$ and all possible input x, there is a Kripke model $\mathcal{K}_T$ that represents the computation tree T of $\mathcal{M}$ on x (5.5).
If there is an accepting computation tree T of $\mathcal{M}$ on x, then at start configuration $\sigma_{\mathcal{M}}(x)$ the following holds by definition:

$$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models ACCEPT \tag{55}$$

We also proved, by lemma 5.7 that for an accepting computation the subformula $START$ holds at $\sigma_{\mathcal{M}}(x)$:

$$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models START \tag{56}$$

Taking the conjunction of these two subformulas at $\sigma_{\mathcal{M}}(x)$ gives:

$$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models START \wedge ACCEPT \tag{57}$$

By lemma 5.8 we have proven that at $\sigma_{\mathcal{M}}(x)$:

$$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models [NEXT^*](CONFIG \wedge MOVE \wedge ACCEPTANCE) \tag{58}$$

The conjunction of subformulas 57 and 58 gives:

$$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models START \wedge [NEXT^*](CONFIG \wedge MOVE \wedge ACCEPTANCE) \wedge ACCEPT \tag{59}$$

And by equation 1:

$$\mathcal{K}_T, \sigma_{\mathcal{M}}(x) \models ACCEPTS_{\mathcal{M},x} \tag{60}$$

□

6 Conclusion

In this thesis we have given a detailed presentation of the proof showing that when there is an PSST ATM $\mathcal{M}$, that has an accepting computation for some input x, then there is a Kripke model that satisfies the formula $ACCEPTS_{\mathcal{M},x}$. This is however only half of the proof of the lower bound time complexity of PDL. The other half shows that if there is a Kripke model that satisfies $ACCEPTS_{\mathcal{M},x}$, then there exists an accepting computation of the PSST ATM $\mathcal{M}$ on input x. Intuitively, a detailed presentation of this half of the proof could have been given by showing that the satisfying Kripke model corresponds to the computation tree of PSST ATM $\mathcal{M}$, which has an accepting computation for input x. This would have shown the non-triviality of the theorem 5.2 that we proved in this thesis, since one could simply construct an $\text{PDL}_{\mathcal{M},x}$ formula, for example the formula $ACCEPTS_{\mathcal{M},x} = STATE_i^q \vee \neg STATE_i^q$, and proof that this formula holds for all accepting computations on any input x at any configuration α in any PSST ATM $\mathcal{M}$, this formula will of course not hold when we include the other half of the proof.

Other interesting future work would be to show that the lower bound time complexity of the satisfiability problem of game logic[7] can be proven with the same method as for the satisfiability problem of PDL, this is due to the similarities of the syntax and semantics of PDL and game logic. We believe that the biggest challenge for obtaining this proof for game logic would be the switch from binary relations in PDL to neighbourhood functions in game logic. This switch would require a reformulation of the subformulas $MOVE$ and $ACCEPTANCE$, due to the use of the program $NEXT$. The subformula START could be used as is. How $MOVE$ and $ACCEPTANCE$ would be reformulated will of course depend on how the game logic language that takes the PSST ATM $\mathcal{M}$ and input x as parameter is defined, however. the differences (if any) between this game logic language and $PDL_{\mathcal{M},x}$ will most likely be very small.

REFERENCES

[1] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, January 1981.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[3] Walter Dean. Computational Complexity Theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2021 edition, 2021.

[4] David Harel, Jerzy Tiuryn, and Dexter Kozen. *Dynamic Logic*. MIT Press, Cambridge, MA, USA, 2000.

[5] Goldreich Oded. *Computational Complexity : A Conceptual Perspective.* Cambridge University Press, 2008.

[6] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[7] Marc Pauly and Rohit Parikh. Game logic: An overview. *Studia Logica: An International Journal for Symbolic Logic*, 75(2):165–182, 2003.