



UNIVERSIDAD DE CONCEPCIÓN
Dirección de Postgrado
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Matemática

ALGUNAS PROPIEDADES DINÁMICAS DE MODELOS DE MÁQUINAS DE TURING

(SOME DYNAMICAL PROPERTIES OF TURING MACHINE DYNAMICAL MODELS)

Tesis para optar al grado de
Doctor en Ciencias Aplicadas con mención en Ingeniería Matemática

Rodrigo Ariel Torres Avilés
Concepción-Chile
Enero, 2016

Profesor Guía: Anahí Gajardo Schulz
Departamento de Ingeniería Matemática
Universidad de Concepción, Chile

Co-tutor: Nicolas Ollinger
Collegium Sciences et Techniques
Université d'Orléans, France

Co-tutor: Eric A. Goles Chacc
Facultad de Ciencia y Tecnología
Universidad Adolfo Ibanñez, Chile

Some Dynamical Properties of Turing Machine Dynamical Models

Rodrigo Ariel Torres Avilés

Profesora Guía: Anahí Gajardo Schulz, Universidad de Concepción, Chile.
Co-tutor: Nicolas Ollinger, Université d'Orléans, France.
Co-tutor: Eric Goles Chacc, Universidad Adolfo Ibáñez, Chile.

Director del Programa: Raimund Bürger, Universidad de Concepción, Chile.

COMISIÓN EVALUADORA

Emmanuel Jeandel, Université de Lorraine, France.
Jarkko Kari, Turun Yliopisto, Suomen Tasavalta.
Marie-Pierre Béal, Université Paris-Est Marne-la-Vallée, France.
Petr Kůrka, Univerzita Karlova v Praze, Česká Republika.
Véronique Terrier, Université de Caen Normandie, France.

COMISIÓN EXAMINADORA

Firma: _____
Alejandro Maass, Universidad de Chile, Chile.

Firma: _____
Anahí Gajardo Schulz, Universidad de Concepción, Chile.

Firma: _____
Nicolas Ollinger, Université d'Orléans, France.

Firma: _____
Pierre Guillon, CNRS, Aix-Marseille, France.

Firma: _____
Xavier Vidaux, Universidad de Concepción, Chile.

Calificación: _____

Concepción, 8 de Enero de 2016

Agradecimientos

*"Pon en manos de Dios todas tus obras,
y tus proyectos se cumplirán". Proverbios 16:3.*

Agradezco a mi tutores de tesis, PhD. Anahí Gajardo Schulz y PhD. Nicolas Ollinger, los cuales, con paciencia, enseñanza, constancia y perseverancia, ayudaron enormemente a la realización y finalización de esta tesis.

Además, quiero agradecer a varios investigadores a quienes tuve la dicha de conocer y compartir, cuyo consejo y conversación enriquecieron y mejoraron mis resultados, como PhD. Pierre Guillon y PhD. Eric Goles Chacc. Además, de manera especial a PhD. Julien Cassaigne y PhD. Emmanuel Jeandel, cuyos resultados sentaron base para la construcción de los resultados expuestos en esta tesis.

También deseo agradecer al Centro de Investigación en Ingeniería Matemática (CI²MA), CAMPUS FRANCE, ECOS C12E05, MECESUP UCO 0713 y FONDECYT 1140684 por el apoyo económico en las distintas pasantías, viajes y conferencias.

Finalmente, quiero agradecer a mi familia, en especial a mi madre y mis hermanos, los cuales fueron el pilar fundamental en toda mi formación académica escolar y universitaria.

Dedico este trabajo a mi esposa Nadia, la cual fue el apoyo sentimental y espiritual durante todo el desarrollo de mis estudios de postgrado. Sin ella ni Dios de mi parte, esto no sería posible.

Abstract

This doctoral thesis is centered on the study of the dynamical properties concerning Turing machines. A Turing machine is quite simple, yet powerful, consisting in a bi-infinite tape of finite alphabet, finite internal states, a head pointing an unique position on the tape and under finite instructions. If the symbol under the head and the internal state match with any instruction, then it is applied, exchanging the symbol, the internal state, and moving the head one position to the left or the right. The Turing machine is the main mechanical model to study computation. As computation is a powerful tool to study large phenomena as the dynamic, therefore it is interesting and fruitfully to study dynamic through Turing machine.

It is not quite natural to see Turing machines as dynamical system, mainly due to headless configurations, as it is on other computation models (as cellular automata), therefore it is tackled from three different systems. The first is commonly called Turing machine with Moving Head (TMH), when the evolution is performed by moving the head, other called Turing machine with Moving Tape (TMT), where it evolves by moving the tape instead of the head, and another one called t -shift, which consist in words of pairs symbol-state, viewing only the content of the head and the internal state in orbits of configurations, and it evolves shifting the words.

The object of the thesis is to study some open questions regarding specific Turing machine dynamical systems, as surjectivity, periodicity in complete and reversible machines, topological entropy, topological transitivity and topological minimality.

The surjectivity in Turing machine is quite easy to decide, but this is not inherited by its t -shift dynamical system. To address this matter, it is defined a new concept called *blocking words*, which is a finite configuration that does not allow the head to visit a certain part of the tape. We prove that having a blocking word in a Turing machine is an undecidable question. Through a reduction from the previous problem, it is then possible to show that the surjectivity is an undecidable property for the t -shift. We prove, using an adaptation of the proof for blocking words that it is undecidable to know if a Turing machine has a positive entropy or not.

Later on, we study a machine created by Julien Cassaigne that we call *SMART*, and we prove that it has several good properties, as being aperiodic, time symmetric, transitive in all three dynamical models, minimal in TMT and with a substitutive t -shift. This machine is the first example of complete and reversible machine that has a transitive TMH dynamical model, a minimal TMT and t -shift dynamical model and it has not a periodic orbit. We show that its existence allows to study in depth the former matters.

With a technique, called *embedding*, we prove the undecidability of aperiodicity and transitivity on Turing machine dynamical systems by using a reduction from the emptiness and mortality problems. We also prove the undecidability of minimality for TMT and t -shift, but no for TMH, as there is no minimal TMH machine. We go further in the study of transitivity by showing that the classes of machines with transitive TMH, TMT and t -shift system are nested and we exhibit examples that prove the inclusions are strict.

In the transitive context, we study that class of the *coded systems*. We exhibit examples to show the diversity of the known subclasses of the coded systems inside t -shifts.

Contents

1	Introducción en Español	5
2	Introduction in English	13
3	Definitions	20
3.1	Dynamical System	21
3.1.1	Orbit	21
3.1.2	Dynamical relations	22
3.1.3	Subshifts, languages and words	22
3.2	Topology	24
3.2.1	Neighborhood, isolated points and closure	24
3.2.2	Topological transitivity, minimality and entropy	25
3.3	Turing machine	26
3.3.1	Determinism and completeness in Turing machine	29
3.3.2	Reversibility in Turing machine	31
3.4	Turing machine seen as dynamical system	33
3.4.1	Turing machine dynamical system	34
3.4.2	Turing machine with moving head (TMH)	34
3.4.3	Turing machine with moving tape (TMT)	35
3.4.4	Relations between dynamical systems of Turing machines	36
3.4.5	The t -shift	36
3.4.6	Turing machine dynamical properties	37
3.5	Arithmetical Hierarchy	40
3.5.1	Hardness and completeness	40
3.5.2	Turing machine examples	41
3.6	Coded systems	42
3.7	Discussion	44
4	Some undecidable problems about the trace-subshift associated to a Turing machine	45
4.1	Problems and concepts	46

4.1.1	Blocking words	46
4.1.2	Surjectivity	48
4.1.3	Positive entropy	50
4.2	Simulating counter machines	52
4.2.1	Construction of the reversible Turing machine that simulates a 2-RCM.	53
4.2.2	Reversing the computation	56
4.3	Undecidability of the problems	57
4.3.1	Undecidability of the blocking state problem in complete RTMs	57
4.3.2	Undecidability of the surjectivity of the subshift associated to a Turing machine	59
4.3.3	Undecidability of the entropy positiveness on reversible one-tape Turing machines	60
5	A SMART machine	63
5.1	A small aperiodic complete and reversible Turing Machine (SMART)	64
5.1.1	The SMART machine.	64
5.1.2	Basic movements of SMART	64
5.1.3	Aperiodicity	66
5.2	Other properties of the SMART machine	69
5.2.1	Some more lemmas	69
5.2.2	The t -shift is substitutive	73
5.3	An application of SMART	76
5.3.1	Proof techniques	76
5.3.2	Undecidability of the aperiodicity of complete reversible Turing machines	77
6	Transitivity and Computability in Turing Machine dynamical systems	80
6.1	The universe of machines with transitive t -shift	81
6.1.1	Machine of type a : Transitive on TMH model	82
6.1.2	Machine of type b : Transitive on TMT, but not in TMH, and without blocking words	87
6.1.3	Machine of type c : Transitive on TMT, with a blocking word	87
6.1.4	Machine of type d : Transitive just for the trace-shift	88
6.2	Undecidability	91
6.2.1	Undecidability results	93
6.3	Complexity of Transitivity Problem and Minimality Problem	98
6.3.1	Transitivity Problem and Minimality Problem are Π_2^0	98
6.4	Coded Systems associated to Turing machines	99
7	Conclusions	102
	Bibliography	107

List of Figures

1.1	Un esquema de una máquina de Turing.	6
2.1	A scheme of a Turing machine.	14
3.1	Instruction of Turing Machine	27
3.2	Example of Turing Machine	28
3.3	Deterministic and non-Deterministic Machine	29
3.4	Example of Turing Machine concepts	30
3.5	Inverting a Turing machine	33
3.6	Turing machine dynamical systems relations	36
3.7	Example of TM, TMH, TMT and t-shift	37
4.1	Transition function of M_C	53
4.2	The routine that writes the sequence “< >” in the tape.	53
4.3	Subrutine depending on sign.	56
4.4	Sequence of states added to M_C	60
5.1	The SMART machine	64
5.2	Two representations of the inverse of the SMART machine	69
5.3	Embedding used in the proof of Theorem 5.4	79
6.1	Universe of the topologically transitive machines.	82
6.2	The SMART machine.	83
6.3	The SMART machine with twice its movement.	87
6.4	Shift machine	88
6.5	The lexicographical ant machine.	88
6.6	Invited machine used in theorem 6.2.1.	96
6.7	Invited machine used in theorem 6.4.	97
6.8	Machine J	100
6.9	Machine J'	101

List of Tables

4.1	Sub-routines corresponding to the different adding instructions.	54
4.2	Sub-routines corresponding to subtraction instructions.	55
7.1	Research on Turing machine dynamical systems.	106



Chapter 1

Introducción en Español



El trabajo que usted está a punto de leer se centra en el estudio de algunas propiedades dinámicas, considerando a la máquina de Turing como el sistema dinámico. La máquina de Turing no solo es la primera definición del modelo computacional, sino que además es simple y funciona mecánicamente similar al computador actual: comenzando con una entrada finita, sólo puede escribir en un lugar dentro de su memoria y moverse a algún lugar vecino. La máquina de Turing clásica consiste en un arreglo unidimensional de celdas llamado cinta, lo cual es similar a la memoria de un computador (RAM), un cabezal con un estado interno apuntando hacia una celda, lo que puede ser interpretado como la CPU, un conjunto finito de instrucciones, que pueden corresponder al software, y dependiendo de la celda apuntada por el cabezal, puede escribir en dicha celda o moverse a una celda adyacente, las cuales corresponden a las posibles acciones dentro de la memoria del computador (para ver un esquema de una máquina de Turing, refiérase a la figura 2.1). La máquina de Turing ha sido utilizada en diferentes campos de investigación, como en complejidad ([44]), pero en esta tesis, estamos interesados en computabilidad y dinámica.

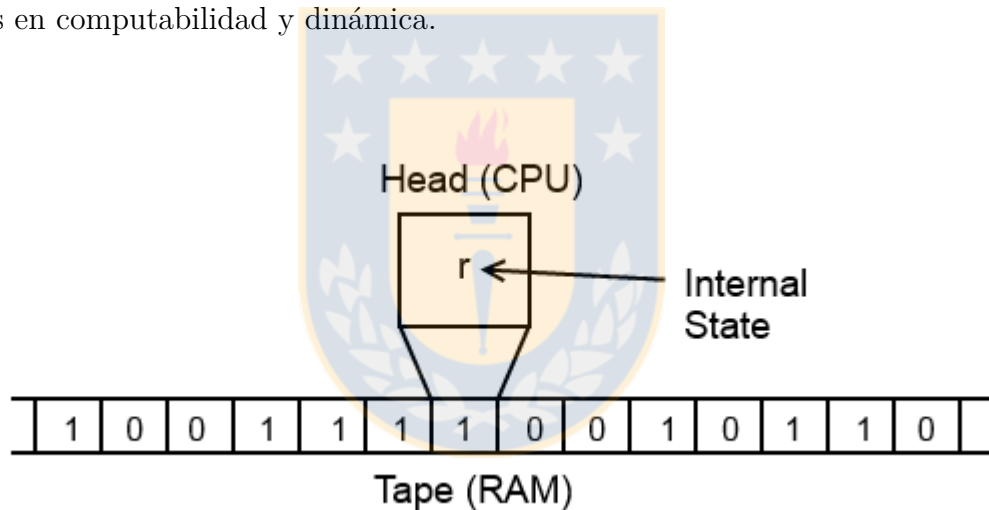


Figure 1.1: Un esquema de una máquina de Turing.

El estudio de la computabilidad comenzó con la concepción de la máquina de Turing. La noción de algoritmo, un procedimiento auto-contenido que corre paso a paso por un tiempo finito, existe hace milenios. La definición formal de algoritmo, fuertemente ligada al concepto de procedimiento efectivo, está relacionada con *entscheidungsproblem*.

El *entscheidungsproblem*, expresión en alemán para *problema de decisión* pide un algoritmo que decida la veracidad o falsedad de una expresión lógica de primer orden. Este fue concebida por *David Hilbert* y *Wilhelm Ackermann* en 1928 [21] en un estudio destinado a formalizar la lógica de primer orden. Este problema, aunque fue concebido más tarde, está incluido en la lista de los veintitres problemas propuestos por Hilbert en 1900, los que se enlazan a una amplia variedad de áreas en la matemática y muchos de ellos, más tarde, muy

importantes en su respectivo campo. Basados en la *enumeración de Gödel* [18] (la que asigna números naturales a fórmulas lógicas), Church mediante el λ -Cálculo [8] y más tarde Turing con máquinas de Turing [49] (de forma independiente) probaron que el cálculo de predicados es indecidible; esto es, no existe un algoritmo que decida el valor de verdad de sus proposiciones. Turing probó que su máquina de Turing es equivalente al λ -Cálculo de Church, y la tesis de Church-Turing establece que todo posible algoritmo (o procedimiento efectivo) es equivalente a una máquina de Turing y de Church. Entonces, el término *algoritmo* está ligado a las máquinas de Turing, de tal manera que hoy en día es considerada la definición formal de algoritmo, como establece Gödel en un *postscriptum* de su trabajo [9] «*El trabajo de Turing da un análisis del concepto de “procedimiento efectivo” (alias “algoritmo” o “procedimiento de computación” o “procedimiento finito combinatorial”) este concepto demostró ser equivalente al de máquina de Turing*». Turing usó un problema inherente a las máquinas de Turing para probar la indecidibilidad del *entscheidungsproblem*. El problema usado fue el *problema del alto*, que pide decidir si una máquina de Turing, comenzando en un estado dado con cierta entrada finita, esta se detiene. Con un argumento de diagonalización, se probó que esto último es indecidible. Hasta el día de hoy, este problema es utilizado para probar la indecidibilidad de varios otros problemas relacionados con computación.

Ya que los problemas relacionados con las máquinas de Turing son descritos en el cálculo de predicados, es posible definir una jerarquía entre los lenguajes descritos por las máquinas de Turing de acuerdo a la complejidad de las fórmulas lógicas que los describen. Esta jerarquización se denomina *Jerarquía Aritmética* [46], la que categoriza las relaciones por la cantidad de cuantificadores existenciales y universales intercalados. El nivel cero, sin cuantificadores, corresponde a los problemas decidibles por una máquina de Turing. El Problema del alto está clasificado en esta jerarquía con solo un cuantificador de existencia, También es posible encontrar problemas más complejos en esta jerarquía (por ejemplo ver [19]).

Antes de entrar en el campo de la dinámica de máquinas de Turing, introduciremos el concepto *sistema dinámico*, el cual consiste en un *espacio de estados* y una *regla fija* que determina el futuro *inmediato* del estado presente del sistema. Este concepto es muy general, el espacio y el tiempo pueden ser discretos o continuos, y la regla, puede ser determinista o no. Ejemplos típicos incluyen la modelación de un péndulo de un reloj, el tamaño de cierta población de animales, o el flujo de agua en una tubería.

Cuando el tiempo es medido en tiempo discreto, se hablará de *sistema dinámico discreto*. esta noción puede ser vista como tomar fotografías del sistema cada cierto intervalo de tiempo (una vez al año, una vez cada milisegundo, etc.). La regla en este caso transforma el estado del sistema en el tiempo n , en el estado del sistema en el tiempo $n + 1$, sin darnos la descripción del

sistema entre dichos tiempos. Para nociones, propiedades y ejemplos de este tipo de sistemas, referase a [31].

Nosotros estamos interesados en sistemas dinámicos discretos con espacio de estados discreto. En este trabajo, llamamos a este sistema simplemente *sistema dinámico*. Un ejemplo interesante, más bien clásico, de sistema dinámico son los *autómatas celulares*. Un autómata celular consiste en una grilla de celdas cada una en un estado perteneciente a un conjunto finito de estos. El estado de cada celda es actualizado a tiempo discreto de acuerdo a una regla determinista que depende de los estados de las celdas pertenecientes a una vecindad de la celda a actualizar. Este concepto fue introducido a partir de una discusión entre von Newman y Ulam en 1951 [41, 50], ambos contemporáneos en Los Alamos National Laboratory. Los autómatas celulares han sido usados como modelo para procesos de vida real en biología y física, como neuronas, turbulencia en sistemas hidrodinámicos y crecimiento ramificado de cristales [24].

Algunos trabajos importantes en autómatas celulares incluyen al autómata celular bidimensional *Game of Life* de Conway en 1970 [34] y el estudio de las reglas unidimensionales elementales por Wolfram en 1983 [53]. *Game of Life* es un juego bidimensional con dos estados y cero jugadores, que, aunque es simple, muestra una dinámica muy compleja. Es uno de los modelos más simples que ha demostrado ser universal (resuelve cualquier algoritmo que resuelve una máquina de Turing) [1] e incluso hubo un boletín dedicado completamente a los resultados de *Game of Life* (*Lifeline*, por Robert Wainwright, 1971-1973). El estudio de Wolfram mostró un comportamiento complejo inesperado en la dinámica de autómatas celulares muy simples, «*legitimizando el campo para la labor de investigación para físicos*» ([24] pag. 4). Más tarde, en 1900, Cook descubrió que una de estas reglas es universal [35].

Tan evidente como era, los autómatas celulares resultaron ser muy ricos en términos de su dinámica, lo que justifica la amplia variedad de estudios en el campo. Existen varios trabajos que prueban la indecidibilidad de ciertas propiedades dinámicas de los autómatas celulares unidimensionales (incluyendo las propiedades que estudiaremos en esta tesis para máquinas de Turing), como las propiedades transitividad topológica, mixing y sensibilidad por Lukkarila en 2009 [33], nilpotencia y periodicidad por Kari y Ollinger en 2008 [28] y reversibilidad y sobreyectividad en dos dimensiones en 1994 por Kari [27].

El estudio acerca de la dinámica de máquinas de Turing es relativamente nuevo. El primer acercamiento a este tópico fue hecho por Moore [36] en 1990. En ese trabajo, Moore presentó un sistema dinámico llamado *generalized shift*, en que es una generalización de la máquina de Turing. Más tarde, Kůrka [30] propuso dos diferentes topologías para máquinas de Turing, una de ellas le da preponderancia a las celdas alrededor del cabezal (Turing Moving Tape model

(TMT)) y la otra se enfoca en la celdas que rodean a la posición “0” de la cinta (Turing Moving Head model (TMH)). Más trabajos acerca de los sistemas dinámicos de Kůrka han aparecido, estudiando propiedades dinámicas como periodicidad [5, 28], entropía [43, 26] y equicontinuidad [14]. Además, otro sistema simbólico fue asociado con las máquinas de Turing [15], tomando el factor columna del TMT, llamado *t-shift*. Esta perspectiva aprovecha el hecho de que todos los cambios en una máquina de Turing ocurren solo en la posición en la q se encuentra el cabezal. Gajardo y sus co-investigadores han estudiado algunas propiedades en este sistema simbólico, como lo son el reconocimiento en tiempo real [14] y la soficidad [13].

Ahora, surge la segunda pregunta: ¿Por qué tomar en cuenta la dinámica en sistemas de computación? Normalmente, un programa computacional toma una entrada finita, y corre por una cantidad finita de tiempo. Surgen preguntas interesantes cuando la entrada o el tiempo se vuelven infinitos, ¿Podemos predecir el comportamiento del computador? ¿Podemos determinar si el programa alcanzará algún estado en el cómputo? ¿Podemos predecir si el cómputo caerá en algún loop? Si el cómputo no cae en un loop, ¿alcanzará todo estado posible de cómputo? ¿Qué tanto nos dirá acerca del cómputo el punto de vista dinámico? Por otra parte, el computador es una poderosa herramienta que nos permite analizar variados fenómenos naturales, en este sentido muchos sistemas y su evolución pueden ser analizadas a través del estudio de su comportamiento dinámico discreto, como la población de ciertos animales, plantas o incluso el universo mismo. En consecuencia, puede ser muy fructífero el analizar la dinámica desde un punto de vista computacional.

A pesar de las similitudes entre las máquinas de Turing y los sistemas dinámicos discretos, no es tan simple modelar una máquina de Turing como un sistema dinámico, ya que una transformación tan directa tiene sus inconvenientes. Una configuración de una máquina de Turing está normalmente definida por su estado interno, el contenido de la cinta y la posición del cabezal. Ya que la posición del cabezal es un elemento en \mathbb{Z} , una secuencia infinita de configuraciones con el cabezal arbitrariamente lejano tiene un punto límite sin cabezal, así el espacio definido por las configuraciones no es compacto, lo que constituye *un serio inconveniente en dinámica topológica* [30], ya que muchas propiedades topológicas no están presentes en espacios no compactos. Los tres modelos que mencionamos previamente son compactos para la topología de cantor. TMT mantiene su compacidad atando el cabezal a la posición central y TMH incorpora el cabezal a la cinta, incluyendo la configuración sin cabezal como un punto fijo. El modelo *t-shift* es compacto pues es un factor de TMT.

Tópicos comunes que aparecen dentro de los sistemas dinámicos incluyen propiedades como la inyectividad, sobreyectividad, periodicidad y propiedades topológicas. En este documento algunas de estas propiedades serán discutidas para máquinas de Turing, estudiando principal-

mente la existencia de una máquina que tenga la propiedad, y su decidibilidad: ¿es posible decidir si el sistema dinámico asociado a una máquina de Turing tiene la propiedad?

La inyectividad y sobreyectividad están fuertemente relacionadas en las máquinas de Turing, siendo incluso equivalentes cuando la máquina no se detiene. La sobreyectividad es una propiedad fácil de decidir en máquinas de Turing, solo requiere de analizar la lista de instrucciones, pero no es el caso con su *t-shift*, existen *t-shifts* sobreyectivos cuya máquina de Turing asociada no lo es. Bajo este tema, se introduce el concepto de *palabras bloqueantes*, un bloque de la cinta con un estado que no permite al cabezal atravesarlo. La relación exacta entre esos conceptos y detalles acerca de la sobreyectividad en *t-shifts* son cubiertos en el capítulo 4.

La entropía topológica para máquinas de Turing ya ha sido estudiada. Esta propiedad da una medida de cuán heterogéneo es el sistema. La entropía es un número no negativo que, en un subshift, describe la diversidad de los patrones finitos que contiene. Oprocha [43] probó que la entropía para máquinas de Turing con cinta móvil (TMT) es equivalente a la entropía del *t-shift*. Más tarde fue probado por Blondel et al. [4] que la entropía de una máquina de Turing con dos o más cintas es incomputable. Ahora, Jeandel [26] ha dado un algoritmo que aproxima la entropía de una máquina de Turing de una cinta por debajo. Este último trabajo muestra que la entropía de una máquina de Turing está relacionada con la tasa de nuevas celdas visitadas por el cabezal y usa una aproximación del grafo de secuencias cruzadas (como en [20]). La indecidibilidad de la positividad de la entropía topológica de una máquina de Turing con una cinta es demostrada en el capítulo 4.

Las propiedades restantes de esta tesis tienen fuerte relación con la reversibilidad. En términos dinámicos, un sistema dinámico se dice reversible si la regla es uno a uno, de este modo cada estado del sistema dinámico tiene un único predecesor. La reversibilidad tiene un trasfondo físico, pues las leyes de movimiento son todas reversibles, y la computación cuántica [42], que necesita una cantidad inalcanzable de disipación de calor para funcionar sin reversibilidad. Esta propiedad, apesar de la simpleza en máquinas de Turing, es altamente estudiada en sistemas dinámicos [27, 33] y sistemas computacionales [39, 37, 40]. Las máquinas de Turing reversibles de un cabezal son universales [39], en el sentido de que pueden realizar cualquier cómputo que una máquina de Turing arbitraria pueda hacer. Existen estudios acerca de sistemas dinámicos asociados a máquinas de Turing reversibles, como el trabajo de Kari y Ollinger [28] que consideran nilpotencia, periodicidad y orbitas periódicas en máquinas de Turing reversibles y otros sistemas computacionales reversibles.

Como se indicó antes, la periodicidad fue estudiada en varios modelos computacionales en [30],[28] y [5]. Kůrka conjeturó que toda máquina completa debe tener al menos una orbita

periódica para el modelo TMT; esta aseveración fue demostrada falsa por Blondel, Cassaigne y Nichitiu, quienes presentaron una máquina completa sin orbitas periódicas. Más tarde Kari y Ollinger presentaron una máquina de Turing reversible no completa sin orbitas periódicas, y probaron que decidir si una máquina de Turing completa, o una máquina de Turing reversible no necesariamente completa, tiene una órbita periódica es indecidible. La existencia y decidibilidad de la existencia de orbitas periódicas en máquinas de Turing reversibles completas fue conjeturada. Una demostración de ambas conjeturas se adjunta en el capítulo 5, gracias a una máquina de Turing particular creada por Julien Cassaigne.

Un sistema es topológicamente transitivo si admite una órbita densa. Como una característica global, la transitividad topológica ha sido estudiada en varios sistemas dinámicos, incluyendo autómatas celulares [33], pues es una propiedad importante para describir el comportamiento del sistema a largo plazo, además de estar relacionada con el *caos*. Aunque la transitividad puede parecer muy restrictiva, existe una amplia gama de sistemas dinámicos topológicamente transitivos, incluyendo sistemas minimales, sistemas con un conjunto denso de puntos periódicos o incluso sistemas sin puntos periódicos. Un estudio acerca de la transitividad topológica en sistemas dinámicos asociados a máquinas de Turing es presentado en el capítulo 6, junto con algunos ejemplos de existencia de máquinas transitivas en cada modelo dinámico y la indecidibilidad de la propiedad.

Un punto en común en las demostraciones de esta tesis es la técnica llamada *embedding*. Embedding consiste en poner una máquina de Turing entera entre dos o más estados de otra máquina de Turing. Si la primera mantiene una propiedad que se sabe indecidible, entonces la máquina entera tiene la propiedad que se desea probar indecidible. La existencia de orbitas periódicas, transitividad topológica, y minimalidad topológica son demostradas indecidibles usando *embedding*.

El documento está organizado en los próximos cuatro capítulos como sigue: Capítulo 3 introduce definiciones generales necesarias para entender los capítulos siguientes y poner la investigación en el contexto correcto. El capítulo 4 se centra en el estudio de la sobreyectividad, palabras bloqueantes y entropía del *t-shift*, constituyendo una transcripción directa del artículo «*Some dynamical properties on Turing machines dynamical systems*», aceptado en Journal Discrete Mathematics & Theoretical Computer Science, en 2015. En el capítulo 5 se discutirá una máquina particular que exhibe muchas propiedades dinámicas, como time-symmetry, minimalidad, transitividad y otras; y demuestra la conjetura de aperiodicidad en [28]. Esto es una traducción directa del artículo *A Small Minimal Aperiodic Reversible Turing machine*, sometido a un journal en 2014. El capítulo 6 presenta un estudio de la transitividad en diferentes modelos dinámicos de la máquina de Turing, para finalmente demostrar la indecidibilidad de la

propiedad en los tres modelos y también la indecidibilidad de la propiedad de minimalidad en TMT y t-shift. El último capítulo está dedicado a conclusiones y perspectivas.



Chapter 2

Introduction in English



The following work you are about to read is centered on studying some dynamical properties, using *Turing machines* as the dynamical system. The Turing machine is not only the first definition of computation model, but also it is simple and it works mechanically similar to a modern computer: Starting from a finite input, it just can write in a unique place inside its memory and move to a neighboring place. A classical Turing machine consists in an infinite uni-dimensional ribbon of cells called tape, which is similar to the memory of a computer (RAM), a head with an internal state pointing to a cell, which can be interpreted as the CPU, a finite set of instructions, which can correspond to the software, and depending on the cell pointed by the head, it can write in that cell or move to an adjacent one, which are the possible actions within a computer memory (to see a scheme of a Turing machine, refer to Figure 2.1). The Turing machine has been used as an object in different research fields, as *complexity* ([44]), but in this thesis, we are interested in *computability* and *dynamics*.

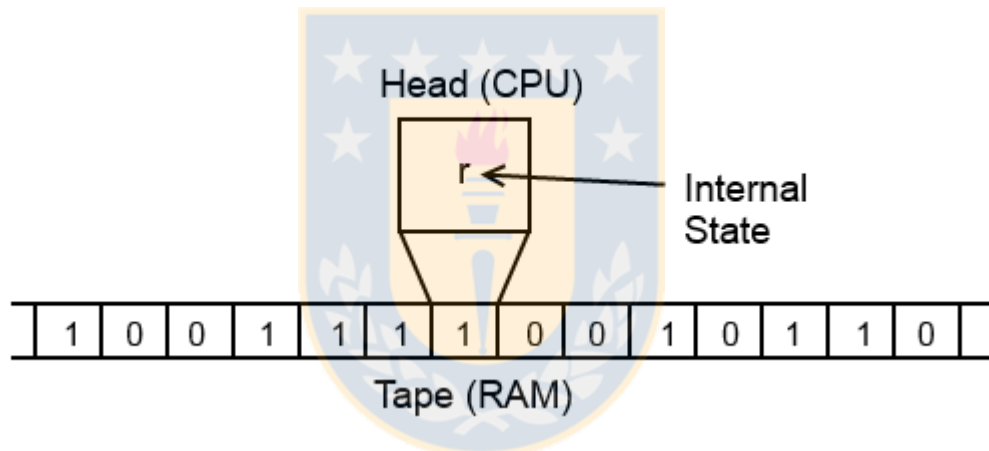


Figure 2.1: A scheme of a Turing machine.

The study about computability began even before the introduction of the Turing machine. The notion of *algorithm*, an step by step self-contained procedure that runs in a finite time, exists since millenniums. The formal definition of algorithm, strongly tied with the concept of effective procedure, is linked with the *entscheidungsproblem*.

The *entscheidungsproblem*, German expression for *Decision Problem*, asks for an algorithm which decides the true or falsehood of any given expression of first order logic. It was conceived by David Hilbert and Wilhelm Ackermann in 1928 [21], inside a study intended to formalize first order logic. This problem, although conceived later, is considered inside the list of twenty three problems given by Hilbert in 1900, which are binded to a wide variety of mathematic fields and most of them were very important afterward in each of its fields. Based on the *Gödel numbering* [18] (to assign numbers to logic formulas), Church through *λ -calculus* [8] and later on Turing with Turing machines [49] (independently) proved in 1936 that predicate calculus

is *undecidable*; that is, there exists no algorithm to decide the truth value of its propositions. Turing proved that its Turing machine is equivalent to Church's λ -calculus, and Church-Turing thesis says that every possible *algorithm* (or *effective procedure*) is equivalent to both Church and Turing *machines*. Then, the term algorithm was tied with Turing machines, in such a way that nowadays it is considered as the formal definition of algorithm, as stated by Gödel in a *Postscriptum* of his work in [9] «*Turing's work gives an analysis of the concept of "mechanical procedure" (alias "algorithm" or "computation procedure" or "finite combinatorial procedure"). This concept is shown to be equivalent to that of a Turing machine*».

Turing used a problem inherent of Turing machines to prove that the *entscheidungsproblem* was undecidable. The problem used was the *Halting Problem*, which asks to decide if a Turing machine, starting from a certain state with a certain finite input, halts. By an argument of diagonalization, the latter was proved to be undecidable. Till today, this problem is used to prove the undecidability of various other problems related with computation.

As the problems linked to Turing machines are described in predicate calculus, it is possible to hierarchize the languages described by a Turing machine according to the *complexity* of the logical formula that define them. This hierarchization is called *arithmetical hierarchy* [46], which categorizes relations by the amount of intercalated existential and universal quantifiers. The level zero, without quantifiers, are the problems that can be decided by a Turing machine. The halting problem is classified in the hierarchy with just one existential quantifier, and also it is possible to encounter problems higher in the hierarchy (for example, see [19]).

Before entering in the dynamical field of Turing machine, let us introduce *dynamical system*, which consists in a *state space* and a *fixed rule* that determines the *immediate* future of the present *state* of the system. This concept is very general, space and time can be discrete or continuous, and the rule, in addition, can be deterministic or not. Typical examples include modeling a pendulum of a clock, the size of an animal population, or the flow of the water inside a pipe.

When the time is measured in discrete time, it is called *discrete dynamical system*. This notion can be seen as taking a snapshot of the system from time to time (once a year, once a millisecond, etc.). The rule in this case transforms the system in time n , to the system in the state $n + 1$, giving us no description of the system in between. For notions, properties and examples in this type of systems, refer to [31].

We are interested in discrete dynamical system with a discrete state space. In this work, we call to this system simply as *dynamical system*. An interesting and rather classical example of dynamical systems are *cellular automata*. A cellular automaton consists in a grid of cells

each one with a state from a finite set of them. The state of each cell in the grid is updated in discrete time according to an homogeneous deterministic rule depending on the states of the cells on a neighborhood around the cell. The concept was introduced by a discussion between von Neumann and Ulam in 1951 [41, 50], both contemporaries at Los Alamos National Laboratory. Cellular automata have been taken as a model of real life process in biology and physics, as neurons, turbulence in hydrodynamical systems and dendritic crystal grow [24].

Some important works in cellular automata include two dimensional cellular automaton Game of Life by Conway in 1970 [34] and a study of the elementary one-dimensional rules by Wolfram in 1983 [53]. Conway's Game of Life is a zero-player two-dimensional two states game, which, although simple, shows a very complex dynamic. It is one of the simplest models proven to be *universal* (it resolves any algorithm that a Turing machine resolves) [1] and there was even a newsletter entirely dedicated to results of The Game of Life (Lifeline. by Robert Wainwright, 1971-1973). The study of Wolfram showed unexpected complex behavior in the dynamics of very simple cellular automata, «*legitimizing the field as research endeavor for physicists*» ([24], pag. 4). Later on, one of those rules was discover to be universal by Cook in 1990's [35].

As evident as it was, cellular automata are very rich in terms of its dynamics, which justifies the wide variety of studies in the field. There exist several works proving the undecidability of dynamical properties of one-dimensional cellular automata (including the properties that we will study in this thesis for Turing machines), as topological transitivity, mixing property and sensitivity by Lukkarila in 2009 [33], nilpotency and periodicity by Kari and Ollinger in 2008 [28] and Reversibility and Surjectivity in two dimensions in 1994 by Kari [27].

The study about the dynamics of Turing machines is relatively new. The first approach in this direction was made by Moore [36] in 1990. In that work, Moore presented a dynamical system called *generalized shift*, which is a generalization of Turing machines. Later on, Kůrka [30] proposed two different topologies for Turing machines, one of them gives preponderance to the cell contents around the head (Turing Moving Tape model (TMT)) and the other focuses on the cells that surround the position "0" of the tape (Turing Moving Head model (TMH)). More works about Kůrka's dynamical systems have arisen, studying dynamical properties like periodicity [5, 28], entropy [43, 26] and equicontinuity [14]. In addition, another symbolic system was associated with Turing machines [15], taking the column factor of TMT, called *t-shift*. This approach takes advantage of the fact that all the changes on the Turing machine happen only on the head position. Gajardo and her co-researchers have investigated some properties in this symbolic system, as real time recognition [14] and soficity [13].

Now, the second question arises: Why take into account the dynamics of computing sys-

tems? Normally, a computation program takes a finite input, and runs for a finite amount of time. Interesting questions emerge when the input or the time becomes infinite. Can we predict the behavior of the computer? Can we determine if the program will reach some state of the computation? Can we predict if the computation will fall into a loop? If the computation does not fall into loop, will it reach any possible state of the computation? How much the dynamical point of view can tell us about computation? In the opposite side, the computer is a powerful tool that allows us to analyze several natural phenomena; in that sense, many systems and their evolution can be analyzed through the study of its discrete dynamic behavior, such as populations of certain animals, plants or even the universe itself. Therefore, it may be fruitful to analyze dynamics from a computational point of view.

Despite the similarities between Turing machines and discrete dynamical systems, it is not that simple to model a Turing machine as a dynamical system, as a straightforward transformation has some drawbacks. A Turing machine configuration is normally defined by its internal state, the content of the tape and the position of the head. As the position of the head is an element of \mathbb{Z} , an infinite sequence of configurations with the head arbitrarily far has a limit point without head, thus the space defined by the configurations is not compact, which is «*a serious drawback in topological dynamics*» [30], as many topological properties are not present in non compact spaces. The three models that we have previously mentioned are compact for the Cantor topology. TMT keeps the compacity by tying the head to the central position and TMH incorporates the head into the tape, including the headless configuration as a fixed point. The t -shift model is compact as it is a factor of TMT.

Common topics that appear within dynamical systems include properties as injectivity, surjectivity, periodicity and topological properties. In this document some of these properties will be discussed for Turing machines, studying mainly the existence of a machine having the property, and decidability: is it possible to decide if a Turing machine dynamical system has the property?

Injectivity and surjectivity are highly related in Turing machines, being even equivalent when the machine does not halt. Surjectivity is an easily decidable property in Turing machines, one just need to analyze the list of instructions, but it is not the case with its t -shift. There exist t -shifts that are surjective, while the associated Turing machine is not. In this fashion, it is introduced the concept *blocking words*, a block of the tape with a state that prevents the head from passing through. The exact relation between these concepts and details about surjectivity on t -shifts is covered in Chapter 4.

Topological entropy of Turing machines has been already studied. This property gives a

measure of how heterogeneous is the system. The entropy is a non-negative number that, in a subshift, describes the diversity of finite patterns that it contains. Oprocha [43] proved that entropy for Turing machine with Moving Tape is equivalent to the t -shift entropy. Later, it was proved by Blondel et al. [4] that the entropy of a Turing machine with two or more tapes is uncomputable. Now, Jeandel [26] has given an algorithm that approaches the entropy of a one-tape Turing machine from below. This last work shows that the entropy of a Turing machine is related to the rate of new cells that the head visits and it uses an approximation of the graph of crossing sequences (as in [20]). The undecidability of the positiveness of the topological entropy of one-tape Turing machines is proved in Chapter 4.

The remaining properties of the thesis have a high relation with reversibility. In dynamical terms, a dynamical system is said to be reversible if the rule is one-to-one, therefore every state of the dynamical system has a unique predecessor. Reversibility has a physics background, as the laws of motion are all reversible, and the *Quantum Computation* [42], which needs an unreachable amount of heat dissipation to work without reversibility. This property, despite simple in Turing machines, is highly studied in dynamical systems [27, 33] and computing systems [39, 37, 40]. Reversible (one-head) Turing machines are *universal* [39], in the sense that they can perform all the computation that an arbitrary Turing machine can do. There exist studies about reversible Turing machine dynamical systems, as the work of Kari and Ollinger [28] which consider nilpotency, periodicity and periodic orbits in reversible Turing machines and other reversible computing systems.

As stated early, periodicity was studied in various computing models in [30],[28] and [5]. K urka conjectured that every complete machine must have at least one periodic orbit for the TMT model; this assertion was disproved by Blondel, Cassaigne and Nichitiu, who presented a complete Turing machine without a periodic orbit. Later on, Kari and Ollinger presented a reversible non-complete Turing machine without a periodic orbit, and proved that deciding if a complete Turing machine, or a reversible not necessary complete Turing machine, has a periodic orbit is undecidable. The existence and the decidability of the existence of periodic orbits in complete reversible Turing machine was conjectured. A proof of both conjectures is addressed in Chapter 5, thanks to a particular Turing machine created by Julien Cassaigne.

A system is topologically transitive if it admits a dense orbit. As a global characteristic, topological transitivity has been studied in various dynamical systems, including cellular automata [33], because it is an important property to describe the general behavior of the dynamics in the long term, in addition to be related with *Chaos*. Although transitivity may seem very restrictive, there exist a wide variety of topologically transitive dynamical systems, including minimal systems, systems with a dense set of periodic points or even systems with

no periodic point at all. A study about topological transitivity in Turing machine dynamical systems is given in Chapter 6, along with some examples of existence of transitive machines in each dynamical model and the undecidability of the property.

One common point of the proofs in this thesis is the proof technique, called *embedding*. Embedding consists in putting a whole Turing machine between two or more states of another Turing machine. If the former maintains some known undecidable property, then the whole machine keeps the property to be proven undecidable. The existence of periodic orbits, topological transitivity and topological minimality are proved to be undecidable by using *embedding*.

The document is organized in the next four chapters as follows: Chapter 3 introduces general definitions necessary to understand the following chapters and puts the investigation in the correct context. Chapter 4 will focus on the study of surjectivity, blocking words and entropy on the t -shift, being an almost direct transcription of the results in “*Some dynamical properties on Turing machine dynamical systems*”, accepted in the journal Discrete Mathematics & Theoretical Computer Science, in 2015. Chapter 5 will discuss a particular machine that meets many rich dynamical properties, as time-symmetry, minimality, transitivity and others; and proves the aperiodicity conjecture in [28]. This is a direct translation of the results of the article “*A Small Minimal Aperiodic Reversible Turing machine*”, submitted to a journal in 2014. Chapter 6 presents a study of transitivity in different dynamical models of the Turing machine, to finally prove the undecidability of the property on all three models and also the undecidability of the minimality property on TMT and t -shift. The last chapter is dedicated to conclusions and perspectives.

Chapter 3

Definitions



In this chapter, we present definitions which are the common factor of the different chapters. Note that the theory behind those subjects is wide and rich, and this chapter provides just about the definitions and gives a reduced state of the art that focuses on the more relevant topics of the presented research. Any specific notion related to just one of the next chapters is presented in that chapter.

3.1 Dynamical System

In a very general context, a *dynamical system* is a 3-tuple (X, T, M) , where X is a set called phase space, M is a monoid and $T : X \times M \rightarrow X$ is a function called *global transition function*, satisfying $\forall i, j \in M : T(x, i + j) = T(T(x, i), j)$. The phase space evolves in time M through the global transition function. This general concept about dynamical systems unifies many different types of global transition functions (or commonly called *rules*). The various ways of measuring the time and the different types of phase spaces determines a variety of different dynamical systems. In this document we focus on *discrete dynamical system* (from now on, called just dynamical system for simplicity), where the space is a compact metric space, and the time $M = \mathbb{N}$ is measured by positive discrete time (in this thesis we consider that $0 \notin \mathbb{N}$). We will omit M in the definition of Dynamical System and thus $T(x, m) = T^m(x)$.

3.1.1 Orbit

In a dynamical system (X, T) , the orbit $O_T(x)$ of a point $x \in X$ is simply defined as $O_T(x) = (T^n(x))_{n \in \mathbb{N}}$. Orbits are introduced to study the long-term behavior dynamics of a system, and such, an important topic to help us in this research.

Pre-periodic and periodic orbits: Also, there exist some interesting types of orbits. In a dynamical system (X, T) a point $x \in X$ has a *periodic orbit* or *cycle* if there exists $n \in \mathbb{N}$ such that $x = T^n(x)$. The point $x \in X$ is called a *periodic point* with period n . If $n = 1$, x is a *fixed point*.

If there exists two distinct numbers $n, m \in \mathbb{N}$ such that $T^n(x) = T^m(x)$, then $O_T(x)$ is called *pre-periodic orbit*, in which case $x \in X$ is called *pre-periodic point*.

If there exists a $n \in \mathbb{N}$ such that $T^n = Id$, the system itself is called *periodic*.

Aperiodic orbits: In a dynamical system (X, T) an orbit is *aperiodic* if no point is repeated

in the orbit.

3.1.2 Dynamical relations

Suppose that we have two dynamical systems (X, T) and (X', T') . Let $\tau : X \rightarrow X'$ be a continuous map such that: $T \circ \tau = \tau \circ T'$.

This function relates these two dynamical systems, and it allows us to study one system through the other one, depending on the character of τ .

The function τ is called a *conjugacy* if τ is a bijective map. It is called a *factor map* if τ is surjective. Finally, it is called a *embedding* if τ is injective.

3.1.3 Subshifts, languages and words

Another type of dynamical system is called *subshift*, which is based in a space of words evolving through the shift function. To give a formal definition, we need first emphasize in some definitions about words.

Given a finite set A , called *alphabet*, $A^{\mathbb{Z}}$ is the set of bi-infinite sequences of elements of A , called *bi-infinite words*. A^{ω} (${}^{\omega}A$) represents the set of right (left) infinite sequences of elements of A , called *infinite words to the right (left)*. The set of infinite words to the right can be also represented by $A^{\mathbb{N}}$; we will use both notations as appropriate to the context. Finally, A^* denotes the set of finite sequences of elements of A , called *finite words*, including the word of length 0; the empty word ϵ . Two finite words $v = v_0 \dots v_n$ and $v' = v'_0 \dots v'_m$ can be *concatenated* by just putting them one after the other: $vv' = v_0 \dots v_n v'_0 \dots v'_m$. A finite word v can also be concatenated with a right infinite word u : $vu = v_0 \dots v_n u_0 u_1 \dots$. A finite word v is said to be a *subword*¹ of another (finite or infinite) word u , if there exists two indexes $i < j$, such that $v = u_i u_{i+1} \dots u_j$. In this case we write: $v \sqsubseteq u$ (and $u \supseteq v$). We use the usual Cantor metric for bi-infinite words: $d(u, v) = 2^{-i}$, where $i = \min\{|n| : u_n \neq v_n, n \in \mathbb{Z}\}$. When it is needed, we mark the center of a (bi-infinite) word by a dot “.”: $\dots u_{-3} u_{-2} u_{-1} . u_0 u_1 u_2 \dots$

We will use two functions over words. First, the *shift* function σ , which is defined both in $A^{\mathbb{Z}}$ and $A^{\mathbb{N}}$ by $\sigma(y)_i = y_{i+1}$. It is a bijective function in the first case. Second, the *prefix*

¹The notion of *subword* usually refers to a sequence obtained by deleting (not necessarily adjacent) symbols in the original word. The former concept is normally called *factor*. In this work, however, we prefer to use *subword* at the place of *factor*, because the latter has another meaning in the context of dynamical systems.

function $()|_n : A^{\mathbb{N}} \rightarrow A^n$ by $z|_n = z_0z_1\dots z_{n-1}$. We also define this function for entire sets $()|_n : \mathcal{P}(A^{\mathbb{N}}) \rightarrow \mathcal{P}(A^n)$ as $S|_n = \{z|_n : z \in S\}$.

Subsets of A^* are called *languages*. Given a subset $S \subseteq A^{\mathbb{Z}}$ (or $S \subseteq A^{\mathbb{N}}$), the language of the subwords of S is defined as:

$$\mathcal{L}(S) = \{v \in A^* \mid \exists u \in S, v \sqsubseteq u\}.$$

Reciprocally, given a language $L \subseteq A^*$, a set of infinite sequences can be defined in $A^{\mathbb{M}}$ ($\mathbb{M} \in \{\mathbb{Z}, \mathbb{N}\}$):

$$\mathcal{S}_L^{\mathbb{M}} = \{u \in A^{\mathbb{M}} \mid \forall v \sqsubseteq u, v \in L\}.$$

When S satisfies $\mathcal{S}_{\mathcal{L}(S)}^{\mathbb{M}} = S$, it is called *subshift*. In this work, we omit the superscript \mathbb{M} , as we will center in subshifts defined for \mathbb{N} (also called *one-sided subshifts*).

Types of subshifts

Subshifts can be equivalently defined by *Forbidden Words*. The subshift defined by a set F of finite words, called *forbidden*, is the following:

$$S_{[F]} = \{u \in A^{\mathbb{N}} \mid \forall v \in F, v \not\sqsubseteq u\}.$$

In fact, any subshift can be defined by a set of forbidden words [31].

When F is finite, we say that $S_{[F]}$ is a *Subshift of Finite Type (SFT)*. An example of a *SFT* is the *golden mean shift*, defined by $S_{\{\{11\}\}} \subseteq \{0, 1\}^{\mathbb{N}}$ (subshift consisting in the sequences with no consecutive pair of 1s). One example of a non *SFT* is the *even shift*, defined by the set $\mathbb{F} = \{10^{2n+1}1 : n \geq 0\}$ (subshift consisting on all the sequences with no odd amount of 0s between 1s).

A factor of a *SFT* is called *Sofic shift*. Any *SFT* is, of course, a sofic shift. The even shift is a sofic shift as it is a factor of $S_{\{\{11\}\}}$, although it is not a *SFT*. One example of a non sofic shift is the *prime shift*, defined by the set $\mathbb{F} = \{10^n1 : n \text{ is prime}\}$.

There are more types of subshifts based on their structure. Two examples of that are *substitutive subshift* and *synchronized subshift*.

A *substitution* ϕ is a morphism $\phi : A^* \rightarrow A^*$, which can be extended to $A^{\mathbb{N}}$. A *fixed point* of ϕ is a word $w \in A^{\mathbb{N}}$ such that $\phi(w) = w$. A *substitutive subshift* is the closure of the orbit of

a fixed point of some substitution.

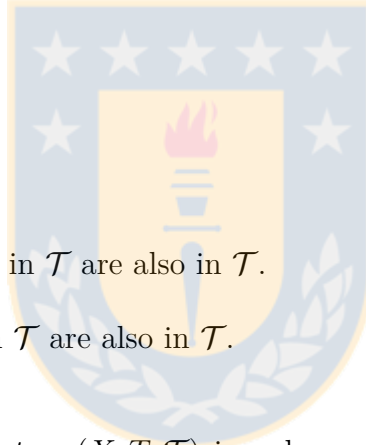
A word v is called *synchronizing* for a transitive subshift S if whenever $uv, vu' \in \mathcal{L}(S)$, then $uvu' \in \mathcal{L}(S)$. A *Synchronized Subshift* is a subshift with a synchronizing word.

3.2 Topology

When we work over dynamical systems, some interesting properties (mainly the ones that we are interested) need additional restrictions to be even defined. In this topic, let us introduce *topology*.

A set X , together with a collection of subsets \mathcal{T} , called *open subsets*, is a *Topological Space*, if it satisfies:

1. $\emptyset \in \mathcal{T}$
2. $X \in \mathcal{T}$
3. Finite intersections of sets in \mathcal{T} are also in \mathcal{T} .
4. Arbitrary unions of sets in \mathcal{T} are also in \mathcal{T} .



A *topological dynamical system* (X, T, \mathcal{T}) is a dynamical system (X, T) with a topology such that T is continuous. When we refer to a topological dynamical system, we omit the collection \mathcal{T} , which it is defined by the open sets of the metric $d : X \times X \rightarrow \mathbb{N}$ in our cases, by the *balls* B_x as: $\mathcal{T} = \{B_x(r) : x \in X, r > 0\}$, where $B_x(r) = \{y \in X : d(x, y) < r\}$. Topological dynamics is interested in the asymptotic properties of the Orbits.

3.2.1 Neighborhood, isolated points and closure

A *neighborhood* of a point $x \in X$ is a set $V \subseteq X$ for which there exists an open set $U \in \mathcal{T}$ such that:

$$p \in U \subseteq V.$$

It is important to note that V does not need to be an open set. If V is an open set, it is called an *open neighborhood*. We denote by $N(x)$ the collection of all neighborhoods of x .

If there exists a point x which have a neighborhood not containing any other point, x is called an *isolated point*.

The *closure* of a set $W \subseteq X$, the set \overline{W} , is defined by:

$$\overline{W} = \{x \in X \mid \forall V \in N(x), V \cap W \neq \emptyset\}.$$

A set W is called *closed* if $W = \overline{W}$.

A point $x \in X$ is the *limit* of a sequence $(x_{(n)})_{n \in \mathbb{N}}$ if: $\forall U \in N(x), \exists m \in \mathbb{N}, \forall n > m \mid x_n \in U$.

It is important to note that, if all convergence sequences $(x_{(n)})_{n \in \mathbb{N}}$ in a set W have a limit $x \in W$, then W is closed.

3.2.2 Topological transitivity, minimality and entropy

Now, we introduce three of the main properties aborded in this research. As stated earlier, these properties are *topological*, therefore, their definition is based in open sets.

In a topological dynamical system (X, T) , a *transitive point* $x \in X$ is a point in the system such that $\overline{O_T(x)} = X$. The existence of such a point in the system leads to some important concepts.

Topological transitivity: A dynamical system (X, T) is *topologically transitive* if it has at least one transitive point.

In the literature, however, there exists another non-equivalent definition for topological transitivity: $\forall U, V \in \mathcal{T}, \exists n \in \mathbb{N} : T^n(U) \cap V \neq \emptyset$. However, if we impose X to be a compact metric space without isolated points, the two definitions are equivalent.

Topological minimality: A dynamical system (X, T) is *topologically minimal* if the orbit of any point is transitive.

Now, let us define the following: For $\varepsilon \geq 0$ and $n \in \mathbb{N}$, we say $E \subset X$ is an (n, ε) -separated set if, for every $x, y \in E$, there exists $0 \leq i \leq n$ such that $d(T^i(x), T^i(y)) > \varepsilon$.

Topological entropy: The topological entropy of a dynamical system is a measure of the diversity of its dynamics. The topological entropy for a topological dynamical system is defined by:

$$H(T) = \lim_{\varepsilon \rightarrow 0} \left\{ \limsup_{n \rightarrow \infty} \frac{1}{n} \log N(n, \varepsilon) \right\},$$

where $N(n, \varepsilon)$ is the cardinality of the maximum (n, ε) -separated set.

For more details about dynamical systems, subshifts and topology, please refer to [51, 31, 29].

3.3 Turing machine

Turing machine (TM) constitutes a simple, yet powerful model of computation. Turing machines are mainly used to define computability (can this problem be solved by a computer?) and complexity (how much time/space is necessary for a machine to solve this problem?). In this thesis, the dynamics of Turing machine is studied, therefore some preliminary considerations have to be made in order to specify, in the best way, Turing machines in the context of dynamical systems.

An important syntactic consideration concerns the elementary action of a Turing machine. Turing machines classically write and move at the same time, but these can also be restricted to either write or move (but not both) at a given step. The latter is a more dynamical way to construct a Turing machine, as it is easy to reverse the time and therefore keep track of it. Both definitions are equivalent in computational terms, but they are not in dynamical terms. We consider both definitions.

A Turing machine M is a tuple (Q, Σ, δ) , where Q is a finite set of states, Σ is a finite set of symbols and δ is a relation that can be defined according the model of Turing machine.

Turing machine written in quadruples

In this model, $\delta \subseteq (Q \times \Sigma \times Q \times \Sigma) \cup (Q \times \{/\} \times Q \times \{-1, 0, +1\})$ is the writing/moving relation of the machine [39].

Turing machine written in quintuples

More classically, δ can be defined as the writing/moving relation considered as $\delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{-1, 0, +1\}$.

Configuration of a Turing machine

The machine works on a tape, usually bi-infinite, full of symbols from Σ .

A *configuration* is an element (r, i, w) of $Q \times \mathbb{Z} \times \Sigma^{\mathbb{Z}}$. A *finite configuration* is an element (r, i, v) of $Q \times \{0, 1, \dots, m-1\} \times \Sigma^m$, for some $m \in \mathbb{N}$. A *right semi-finite configuration* is an element (r, i, u) of $Q \times \mathbb{N} \times \Sigma^{\omega}$. A *left semi-finite configuration* is an element (r, i, u) of $Q \times (-\mathbb{N}) \times {}^{\omega}\Sigma$.

Instructions and evolution of a Turing machine

A writing instruction is a quadruple $(r, \alpha, r', \alpha') \in Q \times \Sigma \times Q \times \Sigma$; it can be applied to a configuration (r'', i, w) if $w_i = \alpha$ and $r = r''$, leading to the configuration (r', i, w') , where $w'_i = \alpha'$ and $w'_k = w_k$ for all $k \neq i$. A moving instruction is a quadruple $(r, /, r', c)$; it can be applied to a configuration (r'', i, w) if $r = r''$, leading to the configuration $(r', i + c, w)$. A quintuple instruction $(r, \alpha, r', \alpha', c)$ can be applied to a configuration (r'', i, w) if $w_i = \alpha$ and $r'' = r$, leading to the configuration $(r', i + c, w')$, where $w'_i = \alpha'$ and $w'_k = w_k$ for all $k \neq i$. It is said that the instruction $(r, \alpha, r', \alpha', c)$ *starts* from state r and *goes* to state r' .

It is forbidden to apply an instruction $(r, \alpha, r', \alpha', c)$ to a finite or (right or left) semi-finite configuration $(r'', i, u) \in Q \times \{0, \dots, m-1\} \times \Sigma^m$, if $i + c \notin \{0, \dots, m-1\}$.

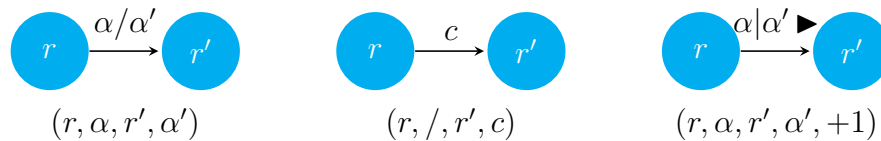


Figure 3.1: Graph representation of instructions in Turing machine. In quintuple context, we use the following symbols to represent movements: $\blacktriangleright = +1$, $\blacktriangleleft = -1$ and $\bullet = 0$.

Turing machines, when viewed as computing model, have a particular starting state r_0 , and a particular symbol called *blank symbol*; the computation is intended to start over a configuration

$(r_0, 0, w)$, where w represents the input and it is a word with a finite number of non-blank symbols, usually in the right part of the tape. The computation process stops correctly when the machine reaches another particular state: the halting state r_F . In general, we are omitting these three parameters, since we do not want the machine to halt in the next chapters, and we will study its dynamics for arbitrary initial configurations. In any case, the halting problem can be translated to the present context as the problem of deciding whether the machine reaches a particular state when starting in another particular state with an homogeneous configuration except for a finite number of cells. We will call this problem as *reachability problem*.

One can translate any machine written in quadruples into a machine written in quintuples in a simple way because *writing instructions* are just quintuple instructions that do not cause any movement, and *moving instructions* are those that do not modify the tape. The reverse transformation is also possible but a quintuple instruction will need to be replaced by a writing instruction followed by a moving instruction (if there is any movement), thus, if every quintuple instruction includes a movement, the set of states needs to be duplicated and the time is also multiplied by two. Therefore, both models are equivalent as computing system, but not as dynamical system. An example of the same Turing machine written in quintuples and in quadruples can be seen in Figure 3.2

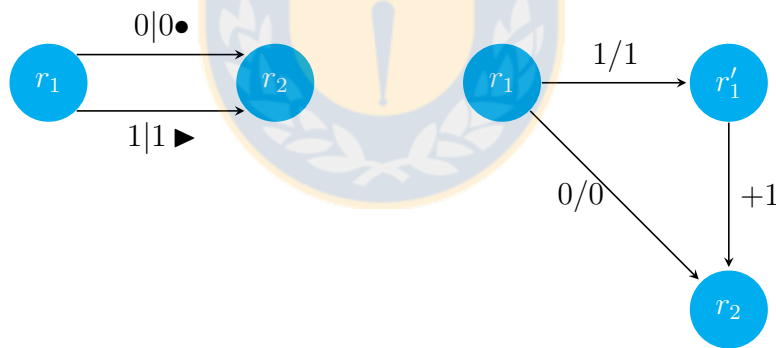


Figure 3.2: Graph representation of a Turing machine M . M calculates binary *AND*; M starts in r_1 with the input in the first two cells, and it gives the solution in the position of the head when it reaches state r_3 . At the left the quintuple representation of M , at the right the quadruple representation of it.

Definition 3.1. For configurations x, x' , we say that M reaches y from x if applying a finite amount of instructions of M to configuration x , configuration y is obtained. We denote this by $x \vdash^* y$. Analogously, for finite configurations z, z' , we say that M reaches z' from z , if, applying the instructions of M to the finite configuration z , the finite configuration z' is attained without exiting the interval $\{1, \dots, m\}$, and we also denote this by $z \vdash^* z'$.

3.3.1 Determinism and completeness in Turing machine

As we stated before, Turing machines have a set of instructions, where one, none or more of them can be applied to a configuration. In this sense, we can define special Turing machines considering restrictions in the amount of instructions that can be applied or lead to the configurations of the machine. There are plenty of studies in this sub-classes, as studies about complexity, for example, the well known conjecture about P vs NP , related with deterministic and non-deterministic Turing machine (see for example [23] and [17]).

Deterministic Turing machine

A Turing machine M is *deterministic* if, for any configuration $(r, i, w) \in X$, at most one instruction can be applied (regardless of the machine being written in quadruples or quintuples). In terms of quintuples, this is equivalent to give δ as a (possibly partial) function $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{-1, 0, +1\}$. This function δ can be projected into three components $\delta_Q : Q \times \Sigma \rightarrow Q$, $\delta_S : Q \times \Sigma \rightarrow \Sigma$ and $\delta_D : Q \times \Sigma \rightarrow \{-1, 0, +1\}$. An example for quintuple model can be seen in Figure 3.3.



Figure 3.3: Graph representation of δ (left) and δ' (right) of non-deterministic machine $M = (\{N, A\}, \{a, b\}, \delta)$ and deterministic machine $M' = (\{N, A\}, \{a, b\}, \delta')$ accepting a^+ . The machine accepts the input when it halts in state A .

From now on, we will work only with deterministic Turing machines. In this context we consider *defective* and *error* states.

A state $r \in Q$ of a Turing machine $M = (Q, \Sigma, \delta)$ is said to be *defective* if there exists a configuration x with state r , such that for all configurations y , applying any instruction to y will not produce x .

The characterization for quadruple model is straightforward: no movement instruction leads to state r , and some pair symbol-state including r does not belong to the image of δ . But for quintuple model it is a little more subtle, as the machine writes and moves at the same time. The characterization for each model can be seen below.

Definition 3.2. A state r is defective if:

1. (Quadruple model) There exist $\alpha' \in \Sigma$ such that neither $(r, 1)$, $(r, 0)$, $(r, -1)$ nor (r, α') belong to the image of δ .
2. (Quintuple model) There exist symbols $\alpha, \alpha', \alpha'' \in \Sigma$ such that neither $(r, \alpha, +1)$, $(r, \alpha', 0)$ nor $(r, \alpha'', -1)$ belong to the image of δ .

Given $c \in \{-1, 0, 1\}$, we also define the defective set of state r in direction c , as $D_c(r) = \{\alpha \in \Sigma : \forall r' \in Q, \lambda \in \Sigma, \delta(r', \lambda) \neq (r, \alpha, c)\}$.

Analogously, we define the error states for both models.

Definition 3.3. A state r' is an error state if:

1. (Quintuple model) There exists a symbol $\alpha' \in \Sigma$ such that $\delta(r', \alpha')$ is undefined.
2. (Quadruple model) There exists a symbol $\alpha' \in \Sigma$ such that both $\delta(r', \alpha')$ and $\delta(r', /)$ are undefined.

Definition 3.4. A state r is said to be reachable from the left (right) if there exist a state r' and symbols α, α' such that $\delta(r', \alpha) = (r, \alpha, 1)$ (resp. -1).

Examples can be seen in Figure 3.4

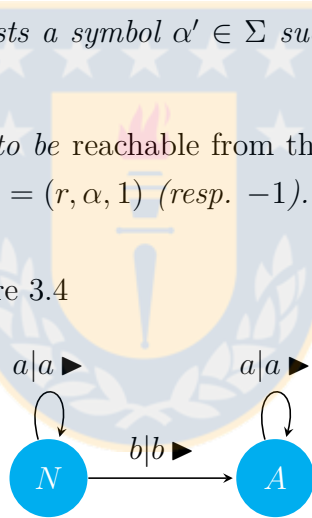


Figure 3.4: Graph representation of machine $M = (\{N, A\}, \{a, b, c\}, \delta)$ accepting a^*ba^* . The machine accepts the input when it halts in state A . Defective State: N , Error State: A , $D_{+1}(N) = \{b\}$, $D_0(N) = D_{-1}(N) = \{a, b\}$. Both states N and A are reachable from the left, but not from the right.

Complete Turing machine

In any of the two models, if no instruction can be applied, the machine halts. A Turing machine M is *complete* if for each configuration (r, i, w) , at least one instruction can be applied, *i.e.*, it never halts or equivalently it has no error state.

Analogous notions of complete and deterministic can be defined when going backward in time.

Backward deterministic Turing machine

A Turing machine M is *backward deterministic* if each configuration comes from at most one previous configuration. More formally, a Turing machine written in quadruples is backward deterministic (as seen in [39]) if and only if for any $r, r', r'' \in Q, \alpha, \alpha', \alpha'', \alpha''' \in \Sigma$, if (r, α, r', α') and $(r'', \alpha'', r', \alpha''')$ are two different instructions in δ , then:

$$\alpha \neq / \wedge \alpha'' \neq / \wedge \alpha' \neq \alpha'''.$$

A Turing machine in quintuple model is backward deterministic if and only if for all state $r \in Q$, r is reachable from at most one direction and for all $\alpha \in \Sigma$, (r, α, d) has at most one pre-image in δ .

Backward complete Turing machine

A Turing machine M is *backward complete* if each configuration comes from at least one pre-image or equivalently if it has no defective states.

3.3.2 Reversibility in Turing machine

We will put special emphasis in this subclass of Turing machines. Reversibility is an important property, as many natural process can be reversed without loss of information (as the laws of motion in physics, or quantum mechanics within weak nuclear force). Many studies have been made about reversible computing systems [33, 37, 38], in particular for Turing machines [39, 28, 40]. In this last context, reversibility gains special importance as it could improve the energy efficiency of computation, specially in Quantum Computation [42].

Reversible Turing machine

A Turing machine is reversible if it is deterministic both forward and backward. For a machine written in quadruples, reversing the quadruples gives the reverse machine. The reverse instruction of a writing instruction (r, α, r', α') is (r', α', r, α) . The reverse instruction of a movement instruction $(r, /, r', c)$ is $(r', /, r, -c)$. We call δ^{-1} to the set of reversed instructions. The machine $M^{-1} = (Q, \Sigma, \delta^{-1})$ is the inverse of $M = (Q, \Sigma, \delta)$, and $M \circ M^{-1} = Id$, the partial

Identity, defined for the subset of halting configurations.

Lemma 3.1. A reversible Turing machine is complete if and only if its reverse is complete.

Proof. If a Turing machine M is complete and deterministic, then there are $|\Sigma|$ instructions starting with state r , each with one different symbol (in the quadruple model, each movement instruction count as $|\Sigma|$ instructions). Then, we have $|\Sigma| * |Q|$ different instructions. Now, each state receives at most $|\Sigma|$ different instructions, because M is reversible, thus each state should receive exactly $|\Sigma|$ instructions. As such, the reverse machine will have $|\Sigma|$ different instructions (or one movement instruction) starting at each state, therefore it will be complete. If the reverse is complete, we can act analogously, taking the reverse machine as the original one. \square

In quintuple model, the inverse of a Turing machine is not, in general, a Turing machine. A quintuple instruction changes the current content at the head position, and moves the head to an adjacent position. If we want to reverse this, we need to, first move the head to the previous position, then change the content, as Turing machine can not work on other cell than the head position, this is not a valid instruction.

We propose to transform the quintuple model machine in an equivalent quadruple model machine, reverse it and then transform it to quintuple model again. As we stated before, for every quintuple instruction $(r, \alpha, r', \alpha', c)$ in a machine $M = (Q, \Sigma, \delta)$, we have two equivalent quadruple instructions $(r, \alpha, r_a, \alpha')$, $(r_a, /, r', c)$, with r_a a new auxiliary state that does not belong to set Q . Please note that, in this case, every writing instruction goes to auxiliary states, and every movement instruction goes to main states. Then, we reverse the instructions to obtain $(r_a, \alpha', r, \alpha)$ and $(r', /, r_a, -c)$.

Here, we can do two things: One option is to replace the writing instruction by a no movement quintuple instruction: $(r_a, \alpha', r, \alpha, 0)$ and the movement instruction by no writing quintuple instructions: $\forall \beta \in Q : (r, \beta, r_a, \beta, -c)$.

The second option is to merge the writing instruction $(r_a, \alpha', r, \alpha)$ with the inverse of the movement instruction that goes to r (if any). If we have a reversible and complete Turing machine, we can keep the same amount of instructions than in the original machine.

In this work, we use the second option. An example in reversing a complete and reversible Turing machine in quintuple model is presented in Figure 3.5. As you can note, the resulting machine has not the same states than the original machine, as we have to use the writing instruction of the auxiliary states to construct a proper quintuple model Turing machine. The inverse machine in this case is defined by $M^{-1} = (Q_a, \Sigma, \delta^{-1})$ where δ^{-1} is obtained as we

described. If we define the map $Aux : Q \times \mathbb{Z} \times \Sigma^{\mathbb{Z}} \rightarrow Q_a \times \mathbb{Z} \times \Sigma^{\mathbb{Z}}$ by $Aux(r, i, w) = (r_a, i + c, w)$, where $c \in \{-1, 0, +1\}$ is the opposite of the unique direction of the instructions reaching the state r , we can have the following: $M \circ Aux \circ M^{-1} \circ Aux^{-1} = Id$. This is always well defined, because in a reversible Turing machine, all the instructions reaching a state $r \in Q$ have the same movement direction.

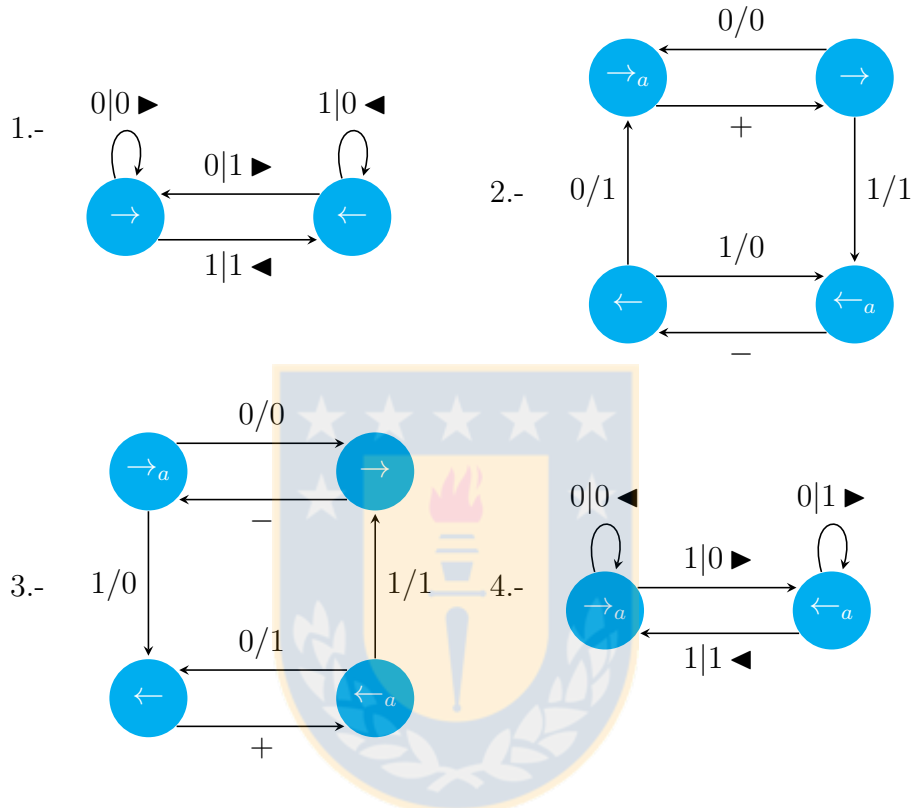


Figure 3.5: The process to invert a reversible and complete Turing machine that counts forward. This machine will be of great use in Chapter 6. 1.- Original machine M ; 2.- Quadruple representation of M , machine \bar{M} ; 3.- The inverse of \bar{M} , machine \bar{M}^{-1} and 4.- Machine M^{-1} , a quintuple representation of M^{-1} .

All of these last properties are local, and they can be checked by scanning the instructions set in a finite number of steps.

3.4 Turing machine seen as dynamical system

A Turing machine is a mechanical model of computation. The computation by itself is (ideally) a finite process, with a finite input and, hopefully, a finite amount of calculus to give an answer. It is not natural to view Turing machines as a dynamical system. Nevertheless, we

present several ways to tackle this problem, mostly addressed by Kůrka [30].

3.4.1 Turing machine dynamical system

A Turing machine $M = (Q, \Sigma, \delta)$ is modeled as a dynamical system as the pair (X, M) where $X = Q \times \mathbb{Z} \times \Sigma^{\mathbb{Z}}$ and M also represents one application of δ to X .

This model has some problems to be considered as a topological dynamical system. The first one is that X is a non compact space, because a headless configuration is the limit point of a series of configurations where the head position goes arbitrarily to the right (left). The second problem is that dynamical systems have total maps, and that M is partial when the original Turing machine M is not complete.

The first problem was solved by Kůrka in two different ways that we describe in sections 3.4.2 and 3.4.3. To solve the second problem, the following definition is introduced.

Immortal Turing machines and immortal configurations

A Turing machine is said *Mortal* if it eventually halts, regardless of the starting configuration. A Turing machine M is called *Immortal* otherwise (as seen in [28, 25]).

Given a Turing machine M , we can define the set $I(M) \subseteq X$ as the set of configurations where M never halts (*Immortal configurations* [25]). This set is closed for the application of M , as an immortal configuration cannot lead to a mortal configuration. As we will state later in section 3.5, to decide if a configuration belongs to this set is not possible.

Restricting a Turing machine to its set of immortal configurations, we obtain a dynamical system $(I(M), M)$.

3.4.2 Turing machine with moving head (TMH)

A way to avoid the first problem (as proposed in [30]) is to add the head as an element of the tape; the configuration (w, i, r) becomes the infinite word x defined by $\forall j < i, x_j = w_j$, $x_i = r$ and $\forall k \geq i, x_{k+1} = w_k$. Thus, the pair (X_h, M_h) consists in: $X_h \subset H = \{x \in (\Sigma \cup Q)^{\mathbb{Z}} : |\{i : x_i \in Q\}| \leq 1\}$, and M_h is one application of δ , taking the consideration that the head is at the right of the position of the unique state on the tape, if there is one, and it acts as the

identity map otherwise. The metric on TMH is the usual Cantor metric defined in the previous subsection.

An element of H is called a *TMH configuration*. We use a version of the operator $I()$ in this system by $I_H(M) \subset H$, such that $X_h = I_H(M)$ is the set of immortal TMH configurations.

We call *finite configuration* of TMH to a finite word $w = u.v$, where $u, v \in (\Sigma \cup Q)^*$ and $|\{i : w_i \in Q\}| \leq 1$, and M_h extends as a partial function to work over finite configuration (M_h is undefined if the head goes out of the domain of the configuration). The point represents the origin. The set of finite configurations of TMH is denoted by X_{h*} .

3.4.3 Turing machine with moving tape (TMT)

Another model of the Turing machine consists in putting the head at the center of the tape (the 0 position) and only moving the tape instead. The dynamical system (X_t, M_t) for TMT consists in: $X_t \subseteq {}^\omega\Sigma \times Q \times \Sigma^\omega$ and M_t is one application of δ by moving the tape instead of the head.

An element of ${}^\omega\Sigma \times Q \times \Sigma^\omega$ is called a *TMT configuration*. Here again, we use another version of operator $I()$ by $I_T(M) \subseteq {}^\omega\Sigma \times Q \times \Sigma^\omega$, such that $X_t = I_T(M)$ is the set of the immortal TMT configurations.

To understand better how the system works, we specify the way in which the instructions are applied: Instruction (r, u_0, s', r', c) is applied to a TMT configuration $(\dots w_2 w_1 w_0, r, u_0 u_1 u_2 \dots)$, resulting in:

- If $c = -1$, $(\dots w_3 w_2 w_1, r', w_0 s' u_1 \dots)$
- If $c = 0$, $(\dots w_2 w_1 w_0, r', s' u_1 u_2 \dots)$
- If $c = +1$, $(\dots w_1 w_0 s', r', u_1 u_2 u_3 \dots)$

We can define the following metric in X_t by $d : X_t \times X_t \rightarrow \mathbb{R}_0^+$, where $d((w, r, u), (w', r', u')) = 2^{-k}$, with $k = 0$ if $r' \neq r$ or $k = \min\{i \in \mathbb{N} : u_i \neq u'_i \text{ or } w_i \neq w'_i\}$. (X_t, d) is indeed a compact space [30].

We call *finite configuration* of TMT, a tuple $(v, r, v') \in \Sigma^* \times Q \times \Sigma^*$, and M_t is extended to these finite configurations respecting the domain of the configuration as in TMH. We denote X_{t*} the set of TMT finite configurations.

In this context, let us abuse the notation for words, and we will say that $x = (v, r, v') \in X_{t*}$ is a *subconfiguration* of a finite or infinite configuration $y = (w, r', w')$, denoted as $x \sqsubseteq y$, if: $\forall n < |v|, m < |v'| : w_n = v_n, w'_m = v'_m, r = r'$.

Remark 3.1. *Both Turing machine with Moving Tape and Turing machine with Moving Head are, in fact, topological dynamical systems. Indeed, TMT is homeomorphic to the Cantor middle third set and TMH is a subshift [30], and, as we know from the literature, every compact metric space is a continuous image of the Cantor set [52].*

3.4.4 Relations between dynamical systems of Turing machines

The relation between the Turing machine dynamical systems presented until now is showed in Figure 3.6. The map $\psi : I(M) \rightarrow X_h$ is injective but not surjective, because X_h can have a headless configuration. The map $\gamma : I(M) \rightarrow X_t$ is surjective but not injective, because the information of the head position is lost.

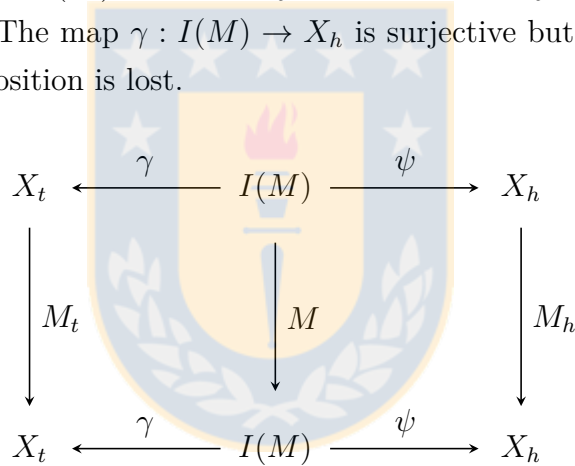


Figure 3.6: Turing machine dynamical systems relations. $(I(M), M)$ is embedded in (X_h, M_h) , (X_t, M_t) is a factor of $(I(M), M)$.

3.4.5 The t -shift

Taking into account the TMT dynamical system, we can define $\pi_t : X_t \rightarrow Q \times \Sigma$ by $\pi_t(w, r, w') = (r, w'_0)$.

Let us define the function $\tau : X_t \rightarrow (Q \times \Sigma)^{\mathbb{N}}$ by $\tau(x) = (\pi_t(M^n(x)))_{n \in \mathbb{N}}$. The t -shift S_M associated to M is $S_M = \{\tau(x) : x \in X_t\}$. The map τ is in fact a factor map from (X_t, T_t) to (S_M, σ) , therefore, S_M is a subshift, because it is σ -invariant and topologically closed [15].

We extended τ to finite words in the following way: $\tau(v) = (\pi(M^j(v)))_{j \in \{0, \dots, m\}}$, where

the number $m \in \mathbb{N}$ represents the amount of instructions that can be applied before the head reaches the limits of the finite configuration $v \in X_{t^*}$.

In Figure 3.7, a comparison between TMT, TMH and t -shift can be seen through an example.

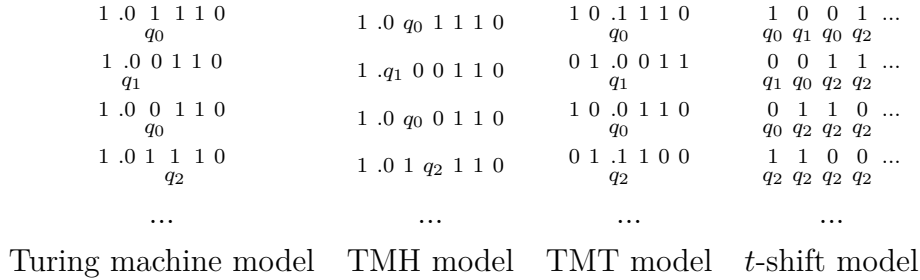


Figure 3.7: Examples of the evolution of a TM in its four dynamical models. The represented machine has the instructions $(q_0, 0, q_2, 1, +1)$, $(q_0, 1, q_1, 1, -1)$, $(q_1, 0, q_0, 0, +1)$ and $(q_2, \alpha, q_2, \alpha, +1)$, $\forall \alpha \in \Sigma$.

3.4.6 Turing machine dynamical properties

Here, we present some of the properties linked with Turing machine dynamical systems that we will study. The first one, *Time Symmetry* is related to reversibility, while the rest are linked to its topology.

Time symmetry

In physics systems, a property is observed in real dynamical systems that is stronger than being reversible, is *time symmetry*. This property «has been mostly neglected when considering (...) discrete dynamics in general.» ([16], pag. 180). When a system presents this property, it is undistinguished if the system goes forward or backward in time. The concept to be presented is an adaptation of the definition of time symmetry for cellular automata [16].

A reversible Turing machine $M = (Q, \Sigma, \delta)$ is said to be *time-symmetric* if there exist involutions $h_Q : Q \rightarrow Q$ and $h_\Sigma : \Sigma \rightarrow \Sigma$ such that:

$$(h_Q(r), h_\Sigma(s), h_\Sigma(s'), h_Q(r'), c) \in \delta^{-1} \iff (r, s, s', r', c) \in \delta.$$

In this case, we can define a function $h : X_t \rightarrow X_t$ by

$$h(w', r, w'') = (h_\Sigma(\dots w'_{-2} w'_{-1}), h_Q(r), h_\Sigma(w'_0 w'')),$$

if state r is reached from the left and

$$h(w', r, w'') = (h_\Sigma(w' w''_0), h_Q(r), h_\Sigma(w''_1 w''_2 \dots)),$$

if state r is reached from the right. The function h satisfy:

$$M_t \circ h = h \circ M_t^{-1}.$$

Topological transitivity

Topological transitivity in the Turing machine context has certain special implications. First, even when we restrict to immortal configurations to talk about Turing machine dynamical systems, we will center in complete Turing machines, as it is not natural to talk about a transitive Turing machine when it halts. Second, the reachability problem, the adaptation of halting problem for complete Turing machines, is meaningless in this context, as every possible input starting in any state has to reach any other possible state.

A second point to take into account, is the two different definitions in general topological dynamical systems. As we stated, the transitive point definition is equivalent to the open set definition when the topological dynamical system has no isolated points. By the definition of TMH and TMT, isolated points are not possible.

Finally, Devaney, in his book [10], call *chaotic* all continuous map with sensitivity on initial conditions, topological transitivity and density of periodic points.

For TMH and TMT dynamical models, the existence of a periodic point in TMH is restrictive for transitivity, as the machine works on a finite part of the tape, and thus unable to modify the rest of it.

In TMH, topological transitivity means that every finite configuration in X_{h^*} , can reach every other one:

$$\forall x, z \in X_h^*, \exists y \in X_h^*, \exists n \in \mathbb{N} : x \sqsubseteq y \wedge M_h^n(y) \sqsupseteq z.$$

In TMT, topological transitivity means that every finite configuration in X_t can reach every other one:

$$\forall (u, r, u'), (v, r', v') \in \Sigma^* \times Q \times \Sigma^*, \exists (w, r, w') \in \Sigma^* \times Q \times \Sigma^*, \exists n \in \mathbb{N} : (u, r, u') \sqsubseteq (w, r, w') \wedge M_t^n(w, r, w') \sqsupseteq (v, r', v').$$

In t -shift, however, we have two different definition. The open set definition: If every two finite words can be linked by a third one: $\forall u, v \in \mathcal{L}(S_T), \exists w \in \mathcal{L}(S_T) : uwv \in \mathcal{L}(S_T)$; and the transitive point definition: $\exists w \in S_T, \forall u \in \mathcal{L}(S_T) : u \sqsubseteq w$.

Nevertheless, the two definitions are equivalent in our context, because we are centered just in complete Turing machine, which implies topological transitive t -shift without isolated points.

Lemma 3.2. A topological transitive t -shift of a complete Turing machine has no transitive point.

Proof. Taking into account the word metric which define the topology in t -shift, there always exists more than one elements in the neighborhood of any point while the Turing machine sees new cells. If the Turing machine do not see new cells, then we are in a periodic point in TMH, making impossible to have topological transitivity. \square

Notation: If a particular Turing machine M has a particular topologically transitive dynamical system, we will say that M is (topologically) transitive *in* this system, either TMH, TMT or t -shift.

Topological minimality

Topological minimality is stronger than transitivity, as this property needs that every orbit is dense, and such, every finite configuration can be found in every orbit.

In TMT, topological minimality means that: $\forall x \in X_t, \forall (u, r', u') \in \Sigma^* \times Q \times \Sigma^*, \exists n \in \mathbb{N} : M_t^n(x) \sqsupseteq (u, r', u')$.

In the t -shift context, minimality means: $\forall x \in X, \forall v \in \mathcal{L}(S_T) : \tau(T(x)) \sqsupseteq v$.

It is important to note that TMH can not be topologically minimal, as the headless configuration can not reach any finite configurations with a head in it.

3.5 Arithmetical Hierarchy

A n -ary relation $R \subseteq \mathbb{N}^n$ is called *recursive* if there exists an algorithm that decides, in finite time, the truth or falsehood of $R(b)$, for all $b \in \mathbb{N}^n$, or equivalently, there exists a Turing machine that decides R . Now, a relation P is called *arithmetical* if it can be expressed as:

$$P(b) \iff \mathbb{Q}_1 x_1 \mathbb{Q}_2 x_2 \dots \mathbb{Q}_n x_n R(b, x_1, x_2, \dots, x_n),$$

where R is a recursive relation and $\mathbb{Q}_i \in \{\forall, \exists\}$ is a quantifier.

Two quantifiers of the same type can be contracted to one quantifier if they are adjacent. In this sense, an arithmetical relation belongs to Σ_n^0 (Π_n^0) if there is no two adjacent quantifiers of the same kind and the first quantifier is \exists (\forall). When no quantifier exists ($n = 0$), then $\Sigma_0^0 = \Pi_0^0$ are the sets of recursive relations.

Some important properties of the arithmetical hierarchy are:

1. If $P \in \Sigma_m^0$ or $P \in \Pi_m^0$, then P belongs to $P \in \Sigma_n^0$ and $P \in \Pi_n^0$ for all $n > m$.
2. For all $n > 0$, Σ_n^0 is closed for existential quantification and Π_n^0 is closed for universal quantification.
3. Σ_n^0 and Π_n^0 are closed for conjunction and disjunction.
4. If $P \in \Sigma_n^0$ then $\neg P \in \Pi_n^0$ and vice versa.
5. Σ_n^0 and Π_n^0 are closed under *bounded quantifiers* ($\mathbb{Q}x < a$).
6. The arithmetical hierarchy does not collapse.

When the quantifiers are applied to function variables, then the hierarchy is called *analytical*, being equivalent to use infinite no-numerable sets instead of natural sets (numerable). For more details about arithmetical and analytical hierarchy, refer to [46].

3.5.1 Hardness and completeness

A relation P is called Π_n^0 -*hard* if all relations $R \in \Pi_n^0$ can be *reduced* to P in the sense that an oracle for characteristic function of P can resolve the characteristic function of R . Also, it is said that P is Π_n^0 -*complete* if it is Π_n^0 -*hard* and $P \in \Pi_n^0$. The same is applicable to Σ_n^0 .

3.5.2 Turing machine examples

Now, let us list some examples of Turing machine problems in relation with the arithmetical hierarchy, in order to understand this relation.

Halting Problem

Given an homogeneous tape with a word $v \in \Sigma^*$ in the center position, a state r_0 and a machine M , decide whether M halts when starting in position 0 and state r_0 . Halting problem is Σ_1^0 -complete. The problem is also Σ_1^0 -complete for reversible Turing machines [40]. The arithmetical form of this problem is: $HP(M, x) \iff \exists t R(M, x, t)$, such that $R(M, x, t)$ is defined as M starting from finite configuration x , with homogeneous tape, halts on time t .

Mortality problem

Given a Turing machine M , decide whether M is a mortal (Section 3.4.1). Mortality problem is Σ_1^0 -complete [22]. The problem is also Σ_1^0 -complete for reversible Turing machines [28]. The arithmetical form of this problem is: $MP(M) \iff \exists t R(M, t)$, such that $R(M, t)$ is defined as M halts at time t , regardless of the starting configuration.

Immortality problem

Given a Turing machine M , decide whether machine M is an immortal machine. As the complement of Mortality Problem, this problem is also undecidable and is Π_1^0 -complete.

Periodicity orbit problem

Given a Turing machine M , decide whether machine M allows a periodic orbit. Periodic orbit problem is Σ_1^0 -complete [28]. The arithmetical form of this problem is: $PO(M) \iff \exists x \exists t R(M, x, t)$, where $R(M, x, t)$ is defined as x is a periodic point with period t for M .

Totality problem

Given a Turing machine M , decide whether M halts on every input. Totality problem is Π_2^0 -complete [6]. The arithmetical form of this problem is: $TT(M, r_0) \iff \forall x \exists t R(M, x, t)$, such that $R(M, x, t)$ is defined as M starting from finite configuration x , with homogeneous tape, halts on time t . In this way, we can say that the Totality problem is *more complex* than Halting problem.

3.6 Coded systems

In practical and coding problems, when we treat with subshifts, SFT and sofic shifts arise more naturally than other more general subshifts. Although, sofic shifts are just a small part of all the universe of subshifts. Also, as t -shifts are related with Turing machines, it is not likely to have sofic t -shifts from most of transitive TMT dynamical systems [13]. In this sense, we present a generalization of sofic shifts, called *coded systems*, introduced by Blanchard and Hansel in 1986 [2].

Countable labeled directed graph

A *countable labeled directed graph* is a directed multi-edge graph (G, L) with countable many states and edges together with a labeling L of the edges by a finite alphabet.

Irreducible graph

A countable labeled directed graph is called *irreducible* if there is a path from any state to another one (it is *strongly connected*).

Locally finite graph

A locally finite graph is a potentially infinite graph with finite in and out-degree from any of its nodes.

Follower set

A *follower set* of a left infinite word w in a subshift $S \subset \Sigma^{\mathbb{Z}}$, $fs(w)$, is the set of all possible right infinite words w' such that $ww' \in S$. Follower set can be defined for finite words in the same way. For a subshift S , we can define the collection of all follower sets in S as $Fs(S) = \{fs(w) : w \in {}^{\omega}\Sigma\}$. Depending on the amount of sets in $Fs(S)$, it is said that S has *countable many* follower sets or *uncountable many* follower sets.

Now, the set X of bi-infinite words presented by bi-infinite paths in (G, L) is not topologically closed, therefore it is not a shift space. The subshift *presented* by (G, L) is \overline{X} . It turns out that every shift can be presented in this way.

Coded System definition

A shift space that can be presented by an irreducible and countable labeled graph is called *coded system*. It is important to note that a shift presented by an irreducible labeled graph is transitive.

Remark 3.1. *If $v \in \mathcal{L}(S)$ is a synchronizing word, $vu \in \mathcal{L}(S)$ and wvu a left infinite word for S , it holds $fs(wvu) = fs(vu)$.*

types of Coded Systems

There exists distinct types of coded systems, depending on certain properties of the graph or the follower set. In particular, if the labeled graph is finite, the system is called *Irreducible Sofic Shift*. If the shift has a synchronizing word, then it is called a *Synchronized System*. Now, we can hierarchize the subclassification of coded systems by the next proposition.

Proposition 3.1. In the context of transitive shifts, the next relations hold [32]:

- X is sofic
- $\implies X$ has countable many follower sets of left-infinite words
- $\implies X$ is a synchronized system
- $\implies X$ is a coded system.

None of these implications can be reversed in the general shift context as proved in [2, 12].

3.7 Discussion

We took some decisions with respect to several of the concepts in this chapter, as skipping some components or deciding to include non canonical definitions. Here we want to discuss the reasons and implications of these elections.

A dynamical system needs not to be defined with a topology. The original dynamical system (X, M) of a Turing machine is more natural than TMT and TMH models of Kůrka, but the theory behind non topological dynamical system is poorer, as several properties can not be defined, as those related with neighborhood, topological entropy, transitivity and minimality.

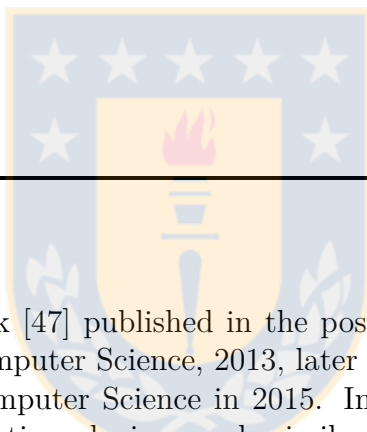
With respect to Turing machines in themselves, these are usually defined with 3 additional elements to the three that we use; an initial state, a set of final states and a special character called *blank*. We implicitly use the initial and final states in a more general context, as defective and error states. These can be calculated from δ , rather than being something explicitly defined. With regard to the blank character, it is meaningless for us because we care about the infinite inputs; the blank character is just another symbol within the set Σ . We discard to study problems related with finite initial conditions, because this kind of restriction is not compatible with compactness, thus not compatible with the dynamical properties that we consider.

In addition, the transition relation δ in Turing machines is normally defined in quintuples, being the model of quadruples a more comfortable way to work with reversible machines. This is not a conflict when dealing with computability, both models are fully equivalent due to the transformation between quintuple and quadruple models, but this is not always the case for the dynamics. Properties as transitivity and periodicity are unaffected by these transformation. Then, the only thing that concerns us is that the time taken for a certain path increases twofold from quintuple to quadruple model.

However, when we look at the t -shift, we are actually losing information with a quintuple model. In the model of quadruples the head does not leave its position after writing, whereas in the quintuple model, the head can leave the position as it write, so that information (which the head wrote) is lost if the head does not return to visit that position. This fact makes us consider both models as it has particular impact on the surjectivity, as will be seen in chapter 4.

When we talk about Turing machines, they are usually not complete. In this sense, we could only watch them as dynamical systems when they are complete. In order to overcome this problem, we define the set $I(M)$ of configurations where the machine does not halt. Although this set is uncomputable (thanks to the halting problem) we know that it is a topological closed set, so our dynamical system is well defined.

Some undecidable problems about the trace-subshift associated to a Turing machine



This chapter is based on a work [47] published in the post-proceedings of Reverse Computation 2012, Lecture Notes on Computer Science, 2013, later completed and accepted in Discrete Mathematics & Theoretical Computer Science in 2015. In this chapter will be presented the concept of *Blocking Words*, a notion playing a role similar to Blocking Words in Cellular Automata, a block of cells/symbols that does not allow information to pass through it (in our case, the head). Several results are presented: Undecidability of both Blocking Words existence, surjectivity on subshift associated to a Turing machine, and positive Topological Entropy in One Head Turing machine. The surjectivity as a Turing Machine property is very simple, it is enough to check the transition rule to decide if a Turing machine is surjective and has direct relation with its reversibility. But this property works more complex in the subshift associated to the Turing machine, depending on blocking words.

4.1 Problems and concepts

4.1.1 Blocking words

The idea of a *blocking* word that prevents the head from going beyond some limit appears in several contexts, and it is related to stability and information travel. In the context of Turing machines, there are several ways of defining it. In a first approach, in a sequence of cells that acts as a “wall”, one can think that if the head goes over these cells, it will read the symbols that they contain and it will “rebound”. But it can change these symbols; then the new configuration should be also “blocking”. On the other hand, the state of the machine is also important; the blocking effect can be a consequence of the combination between the state and the symbols in the tape. In order to fix this notion, we consider the following definitions.

Definition 4.1. *Given a word $u \in \Sigma^*$, we say that the partial configuration $(r, 0, v)$ is a blocking word to the left for a machine M if for every extension $w \in \Sigma^{\mathbb{Z}} : v \sqsubseteq w$ and every time n , the position of the head in $M^n(r, 0, w)$ is greater than or equal to 0. Analogously, we say that $(r, |v| - 1, v)$ is a blocking word to the right if for every extension w of v and every time n , the position of the head in $T^n(r, |v| - 1, w)$ is less than or equal to $|v| - 1$.*

- If $|v| = 1$ we say that (r, v_0) is a blocking pair to the left (right).
- If $v = \epsilon$, we just say that q is a blocking state to the left (right).

Finally, we say that r is just a blocking state if for every $\alpha \in \Sigma$, (r, α) is a blocking pair either to the left or to the right.

These definitions assume that the head starts at the left (right) border of the blocking word. This way of defining blocking words is arbitrary, because the position of the head is independent with the property of block the head to surpass the block of cells, but it allow us to relate this notion with surjectivity in Section 4.3.2.

We consider now the problems of deciding whether a configuration, pair or state is blocking.

- (BC- c) Given a Turing machine M and a partial configuration $(r, 0, v)$ (or $(r, |v| - 1, v)$ for $c = 1$), decide whether the configuration is blocking in direction $c \in \{-1, +1\}$ (or equivalently $\{\text{left, right}\}$).

- (BP-*c*) Given a Turing machine M and a pair (r, α) , decide whether (r, α) is a blocking pair in direction c .
- (BS-*c*) Given a Turing machine M and a state r , decide whether r is a blocking state in direction c .
- (BS) Given a Turing machine M and a state r , decide whether r is a blocking state.

It is important to note that the complexity of these problems does not depend on whether the machine is in the quadruple or quintuple model, since the blocking property talks about the long-term movements of the head.

Since (BS-*c*) reduces to (BS), (BC-*c*) and (BP-*c*), the undecidability of the first implies the undecidability of the other three. The undecidability of (BS-*left*) can be directly obtained by reduction from the Emptiness problem of Turing machines.

Lemma 4.1. (BS-*left*) is undecidable.

Proof. We prove undecidability by reduction from the Emptiness Problem. Let $M = (Q, \Sigma, \delta)$ be a Turing machine, and let $r_0, r_F \in Q$ be two states. We will assume, without loss of generality, that M is written in quintuples, and that starting with r_0 the head never goes to the left of position 0 (this is equivalent to say that the machine works only on the right side of the tape). Let us define M' just like M but with an additional state r_{aux} , and some small differences in its transition function δ :

$$\delta(r_F, \alpha) = (r_{aux}, \alpha, -1), \text{ and } \delta(r_{aux}, \alpha) = (r_{aux}, \alpha, -1), \text{ for every } \alpha \in \Sigma. \quad (4.1)$$

Thus, M reaches r_F for some input $(r_0, 0, w)$ if and only if the state r_0 is not a blocking state to the left for M' . \square

We can also consider the reversible versions of these problems. The importance of considering reversible Turing machines lies in the implications of blocking information in properties related to reversibility, such as surjectivity. This relation can be seen in the next section. The undecidability of the reversible version will be established in section 4.3.1.

4.1.2 Surjectivity

A function is called *surjective* if it is onto, *i. e.*, if every point has a pre-image; in other words, if $M(X) = X$. Surjectivity of Turing machines is a local property; one can decide if a configuration has or not a pre-image just by looking at the state of the head and its surrounding symbols.

Property 4.1. A machine M is surjective if and only if it has no defective state.

When δ is a total function –*i. e.*, when it is defined over its whole domain–, surjectivity of M is equivalent to its injectivity. This fact can be easily established through a combinatorial analysis.

A machine in quadruple model can be directly transformed into the quintuple model, because writing instructions can be replaced by instructions with 0 movement, and movement instructions can be replaced by $|Q|$ instructions of the quintuple model. The global function M does not change with this transformation.

A machine in quintuple model can also be transformed into the quadruple model; but the global function changes, because one instruction is needed to be changed by two instructions.

Surjectivity of (S_M, σ) .

In a subshift, surjectivity simply means that every element is *extensible to the left*: A subshift $S \subseteq A^{\mathbb{N}}$ is surjective if and only if: $\forall w \in S, \exists a \in A : aw \in S$.

Since (S_M, σ) is a factor of (M, X) , the surjectivity of M is inherited by σ in S_M . However, depending on the model, if M is not surjective, σ can still be surjective in S_M .

In the quadruple model, the surjectivity of M is held by the subshift S_M and vice versa. In fact, if we have a defective state r_0 , then there exist no moving instruction leading to r_0 . Thus, in the previous step, the head is on the same cell, and if (r, α) can precede (r_0, α_0) in S_M , one needs $\delta(r, \alpha) = (r_0, \alpha_0)$; thus r_0 is not defective.

However, in the quintuple model, there exist non surjective machines with a surjective trace-shift. Let us illustrate this in the following example.

Example 4.1. Let M be the Turing machine that simply moves to the right by always writing a 0. This machine is not surjective, but its associated subshift is. The unique state of the

machine is defective; it does not admit the symbol ‘1’ at the left of the head, but position -1 is never revisited. That is why any symbol can be appended at the beginning of any $w \in S_M$.

Surjectivity will be possible when defective states avoid the head from going into the “conflictive” positions. If $w = (r_0 r_1 \dots) \in S_M$ and $a = \binom{r}{\alpha}$, condition $aw \in S_M$ says that $\delta(r, \alpha) = (r_0, \beta, c)$ and that the configuration which produces w should have the symbol β at position $-c$. If the machine does not visit the position $-c$ any more, β can be any symbol. We next give a necessary and sufficient condition for a complete machine to have a surjective t -shift.

Proposition 4.1. Suppose M a complete Turing machine. The function σ is surjective on S_M , if and only if for each $r_0 \in Q$ at least one of the following holds:

1. r_0 is not defective.
2. r_0 is blocking to the left (right) and is reachable from the left (right).
3. for every $\alpha \in D_1(r_0)$, $\alpha' \in D_0(r_0)$ and $\alpha'' \in D_{-1}(r_0)$ either
 - (a) $(r_0, 0, \alpha'\alpha'')$ is blocking to the left and r_0 is reachable from the left, or
 - (b) $(r_0, 1, \alpha\alpha')$ is blocking to the right and r_0 is reachable from the right.

Proof. We first remark that 2 implies 3; thus 2 can be suppressed from this proposition. We chose to state it this way because property 2 will be relevant in the next sections.

(\Rightarrow) Let us suppose that σ is surjective in S_M , and let us suppose that neither 1 nor 3 hold for some r_0 . Let us suppose, of course, that r_0 is reachable; otherwise σ cannot be surjective. Let α , α' and α'' be on D_1, D_0 and D_{-1} , respectively, such that they make statement 3 false. Let us suppose first that $\delta(r_0, \alpha') = (r_1, \gamma, +1)$ for some $\gamma \in \Sigma$ and $r_1 \in Q$. Now, let x be any extension of $\binom{\alpha}{r_0} \alpha' \alpha''$ and $w = (r_0 r_1 \dots) = \tau(x)$. We can assume that r_0 is reachable from the left; if not, w cannot be extensible to the left because, with any other movement, α' or α'' should be written, which is not possible. Now, as $(r_0, 0, \alpha'\alpha'')$ is not blocking to the left, there is an extension of $(r_0, 0, \alpha'\alpha'')$ that makes the head go to position -1 . We can take x as this extension (at least over the cells in \mathbb{N}) and equal to α at -1 . We can see that w cannot be extensible because, if x is produced with a movement from the left, α should be written. Therefore, S_T is not surjective and we have a contradiction.

Now, if $\delta(r_0, \alpha') = (r_1, \gamma, -1)$, the proof is analogous. The last case is $\delta(r_0, \alpha') = (r_1, \gamma, 0)$. We have two possibilities: 1) The head eventually moves in some direction; we proceed as before

in that direction. 2) The head remains in the same cell forever; we have a contradiction again because, in this case, (r_0, α') is a blocking pair in both directions, and then r_0 is only reachable from the center; thus any predecessor of x should write α' and stay in place, which was supposed impossible.

(\Leftarrow) Let $w = (\alpha_0^r \alpha_1^r \dots)$ be an element of S_M , generated by a configuration x .

Let us suppose that 1, 2 or 3 hold for r_0 .

1. If r_0 is not defective, then, independently on the context, x can be reached from some configuration.
2. If r_0 happens to be a blocking state to the left (right), no configuration producing w is able to revisit the position -1 ($+1$) (with respect to the initial head position), so any $(r, \alpha) \in Q \times \Sigma$, such that $\delta(r, \alpha) = (r_0, \alpha', +1)$ ($(r_0, \alpha', -1)$) for some α' , can be appended at the beginning of w .
3. Let us suppose that $x = (\dots \beta_{-1} \beta_0 \beta_1 \dots)$ (with $\beta_0 = \alpha_0$). If $\beta_d \notin D_{-c}$, for some $d \in \{-1, 0, 1\}$, then we can extend w to the left (right) with a pair (r', λ) such that $\delta(r', \lambda) = (r_0, \beta_c, -c)$. If $\beta_c \in D_{-c}$, for all $d \in \{-1, 0, 1\}$, then, due to 3, we can extend w to the left (right) with any $(r, \lambda) \in Q \times \Sigma$, such that $\delta(r, \lambda) = (r_0, \lambda', c')$ for some $\lambda' \in \Sigma$, with c' depending on whether β_{-1}, β_0 and β_1 satisfy 3.a or 3.b.

Therefore, w has a preimage by σ . Since w is arbitrary, σ is surjective on S_M . \square

We consider the following problem.

(Surj) Given a deterministic and complete Turing machine written in quintuples, decide whether its t -shift is surjective.

(Surj) is related to (BS- c) and (BP- c) as can be appreciated in Proposition 4.1, but they are all undecidable, as we will see in Section 4.3.

4.1.3 Positive entropy

The *entropy* of a subshift S is given by the following limit,

$$H(S) = \lim_n \frac{1}{n} \log(\#(S|_n))$$

This definition is based on the *words* of the trace-shift, which correspond to partial developments of the machine dynamics. The interesting work of [7] proves that entropy of a subshift can be computed as a maximum over its elements, *i. e.*, over complete traces; more precisely, a maximum of the *Kolmogorov complexity* of the elements. The Kolmogorov complexity $K(v)$ is defined as the length of the shorter program that computes a finite word v . For an infinite sequence $u \in S$, the *upper complexity* of u is defined by $\overline{K}(u) = \limsup_n \frac{K(u|_n)}{n}$. Brudno proves that the entropy of a given subshift S is given by the next equality.

Theorem 4.1.

$$H(S) = \max_{u \in S} \overline{K}(u) \text{ [[7]]}$$

There is a rich theory about Kolmogorov complexity, a good reference on the subject is the book of [11]. Here we will only use two classical properties.

Property 4.2. 1. There exists a constant C such that, for all finite words v, v' , $K(vv') \leq K(v) + K(v') + C$.

2. For every $\alpha \in \Sigma$ and $n \in \mathbb{N}$, $K(\alpha^n) = O(\log(n))$.

These allow us to prove the next simple lemma.

Lemma 4.2. For any finite word v and symbol α , it holds that $\overline{K}(v\alpha^{\mathbb{N}}) = 0$.

Proof. If we define $u = v\alpha^{\mathbb{N}}$, from the last properties, we have

$$\overline{K}(u) = \limsup_n \frac{K(u|_n)}{n} \leq \limsup_n \frac{K(v) + K(\alpha^{n-|v|}) + C}{n} = 0.$$

□

In [26], an algorithm that approximates the entropy of the trace-shift of a one-dimensional Turing machine is developed. His result is strongly based on the next property.

Property 4.3 ([26]). 1. A positive upper complexity is reached on configurations on which the head visits each position of the tape finitely many times.

2. From the last assertion, we know that some of the configurations with maximal upper complexity never cross the origin.

We will show that, although computable by approximation, the entropy of a trace-shift cannot be distinguished from 0 in finite time. In other words, we will prove that the next problem is undecidable.

(PE) Given a deterministic and complete Turing machine T , decide whether $H(S_T) > 0$.

Remark 4.1. *It is important to point out that, if we transform a TM from the quintuple model to the quadruple model, the trace-subshift and its entropy change, but only by a factor of two. Thus, its positiveness is not affected. Thus the complexity of (PE) does not depend on the model in which T is expressed.*

4.2 Simulating counter machines

The main difficulty when proving undecidability of a TM property that comes from dynamical systems theory is that it has to do with the whole set of trajectories; it is not restricted to trajectories that start at particular points. The three undecidability proofs that we present here are based on a machine that persistently simulates a counter machine from a particular state. In this way, some of the dynamical properties of our Turing machine will talk about the behavior of the counter machine over its initial state, inheriting, in this way, some of its undecidable properties.

Definition 4.2. *A k -counter machine (k -CM) is a triple (Ω, k, R) , where Ω is a finite set of states, $k \in \mathbb{N}$ is the number of counters, and $R \subseteq \Omega \times \{0, +\}^k \times \{1, \dots, k\} \times \{-1, 0, +1\} \times \Omega$ is the transition relation. A configuration of the machine is a pair (s, ν) , where s is the current state and $\nu \in \mathbb{N}^k$ is the content of the k counters. By considering the function $\text{sign}: \mathbb{N}^k \rightarrow \{0, +\}^k$ defined by $\text{sign}(\nu)_j = 0$ if $\nu_j = 0$ and $+$ otherwise, an instruction $(s, \theta, i, c, t) \in R$ can be applied to a configuration (s, ν) if $\text{sign}(\nu) = \theta$, and the new configuration is (t, ν') where $\nu'_j = \nu_j$ for every $j \neq i$ and $\nu'_i = \nu_i + c$. R cannot contain the instruction $(s, \theta, i, -1, t)$ if $\theta_i = 0$.*

Since the transition relation is not necessarily a function, the defined system is not necessarily deterministic, nor defined for all configurations. We will restrict our attention to deterministic and one-to-one (reversible) 2-counter machines (2-RCM). [38] proves that a two-counter machine is reversible if for every state $s \in \Omega$ and $\theta \in \{0, +\}^2$, the machine can reach a configuration (s, ν) , with $\text{sign}(\nu) = \theta$, from at most one state.

In the next section, we define a reversible Turing machine that simulates a given arbitrary 2-RCM. The simulation is fairly standard, using special symbols to separate each counter, which are represented by series of cells containing 1s. In a first stage we define an incomplete RTM, but in section 5.3.1, we recall one of the methods presented in [28] that allows to complete any reversible transition rule.

4.2.1 Construction of the reversible Turing machine that simulates a 2-RCM.

Given a 2-reversible counter machine $C = (\Omega, 2, R)$, we construct a reversible Turing machine $M_C = (Q_C, \Sigma, \delta)$ defined in quadruples. Its state set is $Q_C = \Omega \cup Q_0 \cup \Omega_1 \cup \dots \cup \Omega_{|R|}$, where Q_0 is the set of states destined for initialization, and Ω_i is the set of states needed to simulate the i -th instruction of R . The symbol set is $\Sigma = \{<, |, >, 1\} \cup \{0, +\}^2$. The first set recreates the counters on the machine, and the second contains auxiliary symbols indicating the status of the counters. We describe the transition function of M_C with figures; the used notation is specified in Figure 4.1.

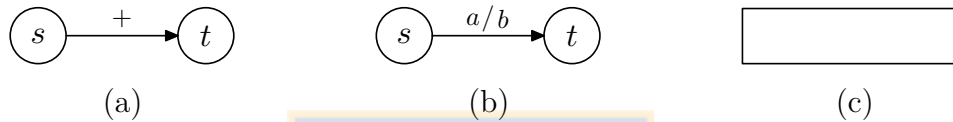


Figure 4.1: (a): Instruction $\delta(s, /) = (t, +)$. (b): Instruction $\delta(s, a) = (t, b)$. (c): Subroutine.

The idea behind this simulation is that each configuration $(s, (n, m))$ of the counter machine will be represented in the Turing machine by the configuration $(s, 0, \cdot(\text{sign}(n), \text{sign}(m))1^n|1^m > 111\dots)$. The machine simulates C starting from configuration $(s_0, (0, 0))$ by writing “< | >” on the tape. State $q_0 \in Q_0$ initiates a subroutine (see figure 4.2) that first writes “< | >” on the tape, and then comes back to replace “<” by “(0, 0)” and to pass to state s_0 . Subsequently, the machine adds and removes 1’s from the tape, according to the instructions of C .

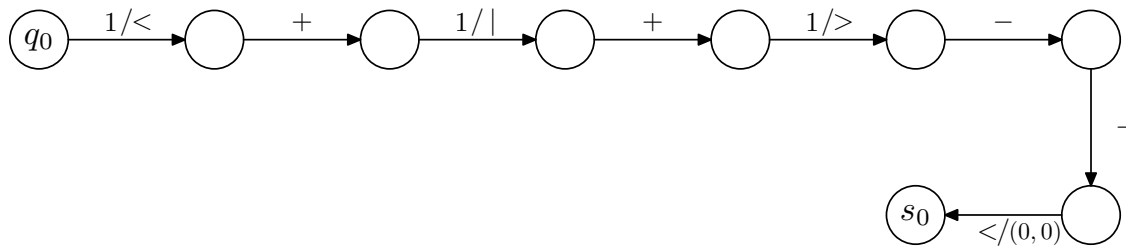


Figure 4.2: The routine that writes the sequence “< | >” in the tape.

Each sub-routine uses an exclusive set of states. The state set Ω_i is dedicated to perform the i -th instruction of R . There are several cases, depending on the action over the counters, the affected counter and its *sign*. There are eight addition instructions represented by four

subroutines that can be found in table 4.1. There are only four subtraction instructions, since the counter must be non-empty if we want to subtract from it. Subtractions are represented in table 4.2. Instructions with no action on the counter are simpler; instruction $(s, (c, c'), i, 0, t)$ is simply represented by the transition $\delta(s, (c, c')) = (t, (c, c'), 0)$.

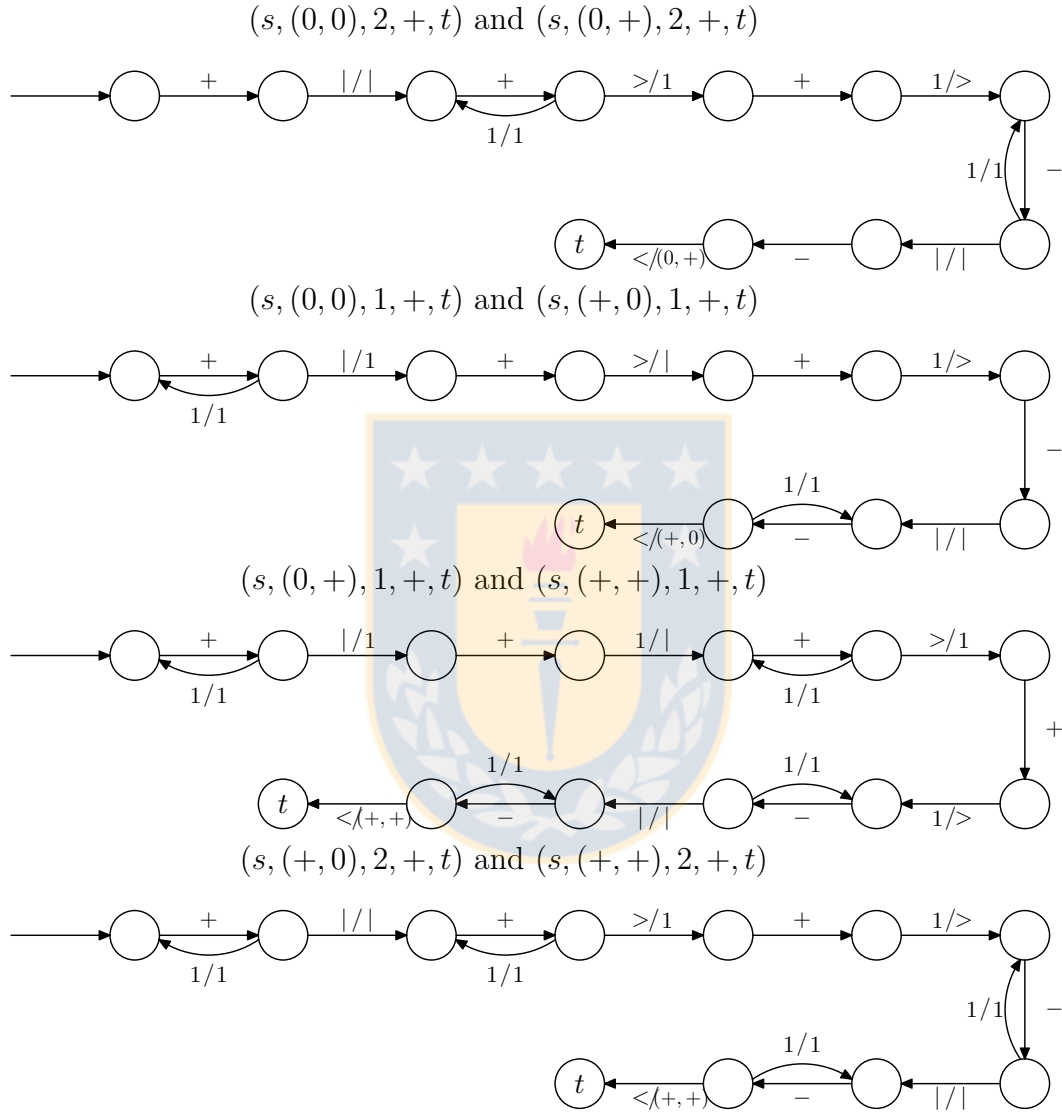


Table 4.1: Sub-routines corresponding to the different adding instructions.

The machine will work as long as the background is full of 1's (the new visited cells). If it encounters any other symbol, it halts. It is important to note that, before reaching any state of C , the machine replaces the symbol “<” by the pair $(c, c') \in \{0, +\}^2$ in order to indicate the *sign* of each counter at the end of each instruction. In this way, the machine knows which instruction of the counter machine is the next to be applied. If M_C is in a state of C , reading a sign (c, c') will call a subroutine which executes the corresponding instruction of R . This is illustrated in figure 4.3.

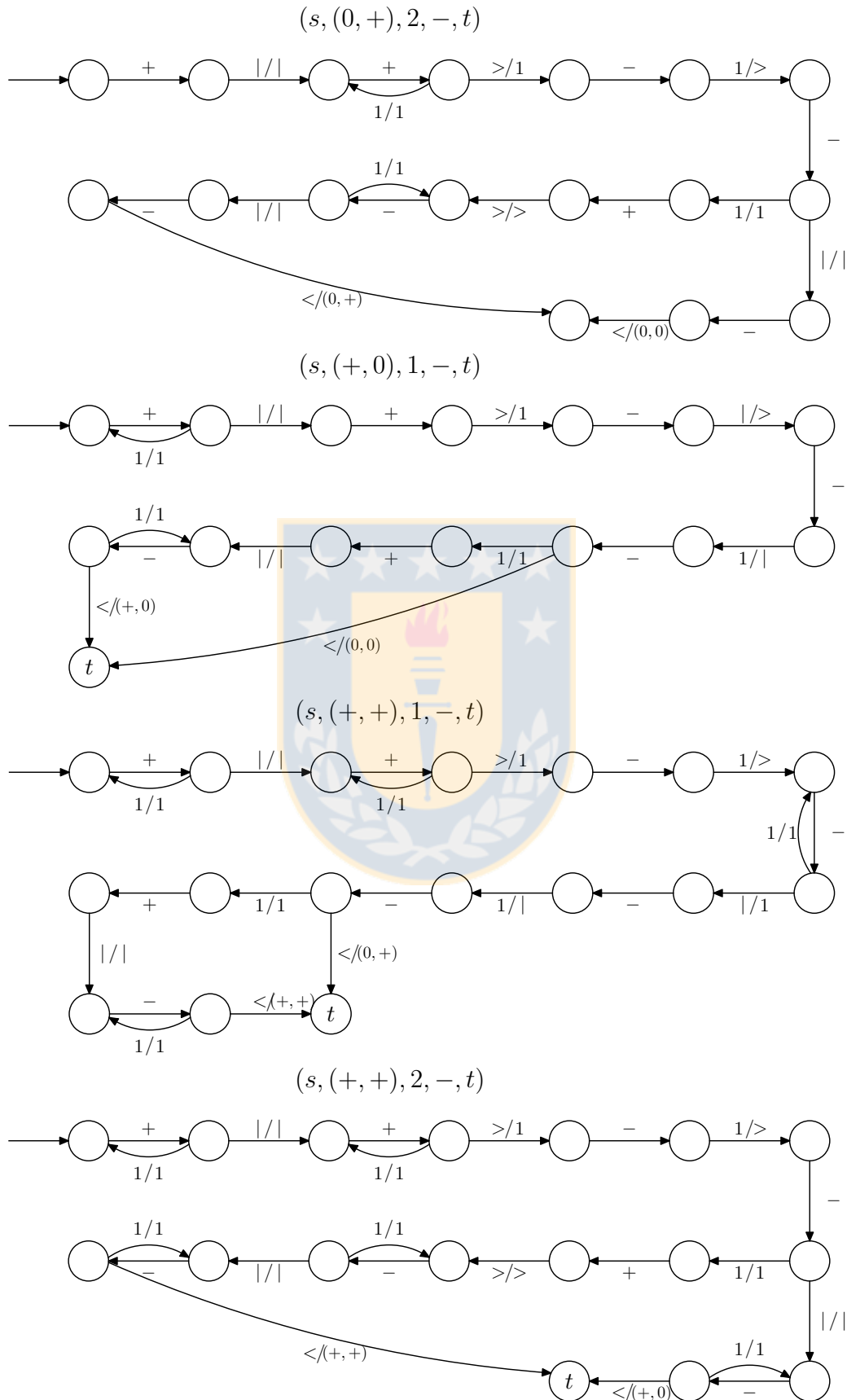


Table 4.2: Sub-routines corresponding to subtraction instructions.

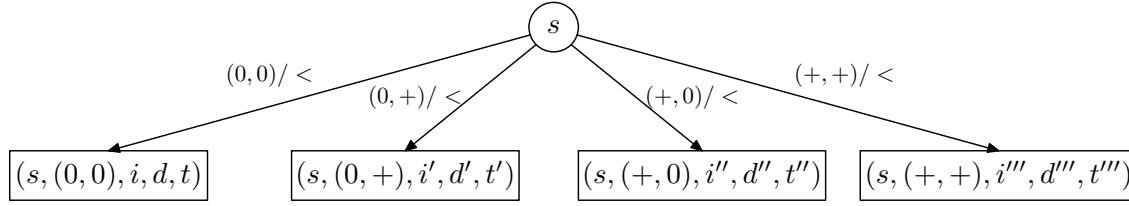


Figure 4.3: Depending on the *sign* of the counters, the machine performs the instruction $(s, (0, 0), i, d, t)$, $(s, (0, +), i', d', t')$, $(s, (+, 0), i'', d'', t'')$ or $(s, (+, +), i''', d''', t''')$.

The head will recurrently come back to the position where the symbol ' $<$ ' was placed. M_C will halt if at some moment it encounters an unexpected symbol. It will also halt if it attains a halting state of C .

Remark 4.1. *The Turing machine M_C that simulates 2-RCM machine C is reversible.*

In fact, in the quadruple model, a machine is reversible if for every two different instructions going to the same state $\delta(q_1, \alpha) = (q, \alpha')$, $\delta(q_2, \alpha'') = (q, \alpha''')$, we have that $\alpha' \neq \alpha'''$ and $\alpha \neq /$, $\alpha'' \neq /$. From tables 4.1 and 4.2, and figures 4.2 and 4.3 it can be directly verified that every state in $Q_0 \cup \Omega_1 \cup \dots \cup \Omega_{|R|}$ satisfies these requirements. We need to be more careful with the states in Ω . Different sub-routines can arrive to the same state in Ω , but they always write the sign of the counters on the tape. Therefore, since C is reversible, M_C will always write different signs when arriving to the same state from different subroutines, which imply that M_C is reversible.

4.2.2 Reversing the computation

In [28] a technique is presented that allows completing a reversible Turing machine. We recall it here. Given a reversible Turing machine $M = (Q, \Sigma, \delta)$ written in quadruples, a new machine $M' = (Q \times \{-, +\}, \Sigma, \delta')$ is defined as follows.

$$\begin{aligned} \delta(r, /) = (r', c) &\Rightarrow \delta'((r, +), /) = ((r', +), c) \text{ and } \delta'((r', -), /) = ((r, -), -c) \\ \delta(r, \alpha) = (r', \alpha') &\Rightarrow \delta'((r, +), \alpha) = ((r', +), \alpha') \text{ and } \delta'((r', -), \alpha') = ((r, -), \alpha) \end{aligned}$$

In other words, $(r, +)$ and $(r, -)$ represent M in state r running forwards or backwards, respectively.

Moreover, if at some iteration no instruction of M can be applied, the “time direction” is switched. This is performed by adding the next instructions.

$$\begin{aligned}
 \text{no transition defined for } r &\Rightarrow \delta'((r, +), /) = ((r, -), 0) \\
 \text{else if neither } \delta(r, /) \text{ nor } \delta(r, \alpha) \text{ are defined} &\Rightarrow \delta'((r, +), \alpha) = ((r, -), \alpha) \\
 \text{no transition arrives to state } r &\Rightarrow \delta'((r, -), /) = ((r, +), 0) \\
 \text{else if neither } (r, \alpha), (r, -1), (r, 0) \text{ nor } (r, 1) \text{ are in the image of } \delta &\Rightarrow \delta'((r, -), \alpha) = ((r, +), \alpha)
 \end{aligned}$$

We will apply this technique not only to machine M_C but also to some modified versions of it.

4.3 Undecidability of the problems

4.3.1 Undecidability of the blocking state problem in complete RTMs

The problem we study in this section is not exactly the one described in section 4.1.1. It is a restriction of the reversible version of (BS-*left*), thus it directly reduces to (BS-*left*). The interest behind this restriction will be clear in section 4.3.2.

(BS-*left Artm*) Given a complete and reversible Turing machine M and a state r that is reachable from the left, decide whether r is a blocking state to the left.

We prove the undecidability of this problem by reduction from the halting problem of reversible two-counter machines, which is proved undecidable in [38]. The halting problem in this case consists in determining, given an initial configuration (s, ν) , whether the machine reaches a given halting state s_f . It is undecidable for $k = 2$, even if the initial configuration is fixed to $(s_0, (0, 0))$.

(Halt2rcm) Given a 2-RCM C and two states s_0 and s_f , decide whether C arrives to s_f when starting from configuration $(s_0, (0, 0))$.

Theorem 4.2. *(BS-*left Artm*) is undecidable.*

Proof. Let $C = (\Omega, 2, R)$ be a 2-RCM, with initial configuration $(s_0, (0, 0))$ and final state $s_f \in \Omega$. We will define a Turing machine M and a state $r = r_0$ meeting the next conditions:

1. it simulates C on the right side of the tape,
2. it is reversible,
3. it reaches the position -1 starting at 0 from r_0 if and only if C halts (reaches s_f) starting from $(s_0, (0, 0))$,
4. it is complete, and
5. r_0 is reachable from the left.

If the machine meets the above objectives, r_0 will not be blocking to the left if and only if, C halts when it starts from $(s_0, (0, 0))$.

The machine M_C defined in section 4.2.1 satisfies the first 2 objectives.

Now, in order to reach the third goal, we add an extra state and an extra rule to M_C : $\delta(s_f, /) = (r_{aux}, -1)$. In this way, M_C is able to reach the position -1 , starting from r_0 , if and only if C halts when it starts from $(s, (0, 0))$.

We next apply the technique *reversing the computation* described in section 5.3.1 to obtain a complete machine $M'_C = (Q', \Sigma, \delta')$, with $Q' = Q_C \times \{+, -\}$.

The fifth goal is attained by modifying M'_C in only one instruction:

$$\delta'((r_0, -), /) = ((r_0, +), 0) \text{ is switched to } \delta'((r_0, -), /) = ((r_0, +), 1).$$

This instruction exists because r_0 is not attained by δ .

Thus, r_0 is a blocking state to the left, reachable from the left, for the complete reversible TM M'_C if and only if C does not halt (reaches the s_f state) from $(s_0, (0, 0))$.

□

We would like to remark that in this proof we assume M in the *quadruple* model; but M_C can be transformed to the *quintuple* model at the end of the proof, the result remaining the same.

4.3.2 Undecidability of the surjectivity of the subshift associated to a Turing machine

We will use problem (BS-left Artm) to prove the undecidability of the surjectivity on the trace-shift.

(Surj) Given a deterministic and complete Turing machine written in quintuples, decide whether its trace-shift is surjective.

Theorem 4.3. *(Surj) is undecidable.*

Proof. We prove this by reduction from (BS-left Artm). Let $M = (Q, \Sigma, \delta)$ be a complete and reversible Turing machine written in quintuples. Let r' be a state that is reachable from the left, and let r, α and α' be such that $\delta(r, \alpha) = (r', \alpha', +1)$. We know that this machine is surjective.

Now let us define M' as M , but with an additional state r_{aux} and the following new instructions:

$$(\forall \beta \in \Sigma) \delta(r_{aux}, \beta) = (r', \beta, 0) . \quad (4.2)$$

And changing

$$\delta(r, \alpha) = (r', \alpha', +1) \text{ by } \delta(r, \alpha) = (r_{aux}, \alpha', +1). \quad (4.3)$$

M' is not surjective, because the configuration (r_{aux}, i, w) has not preimage if $w_{i-1} \neq \alpha'$. So r_{aux} is the unique defective state of M' , with $D_1(q_{aux}) = \Sigma - \{\alpha'\}$ and $D_0(r_{aux}) = D_{-1}(r_{aux}) = \Sigma$.

In this way, if r' is a blocking state to the left for M , then so is r_{aux} for M' , and it is also reachable from the left. Therefore, by Proposition 4.1, $S_{M'}$ is surjective if and only if r' is a blocking state to the left for M .

□

4.3.3 Undecidability of the entropy positiveness on reversible one-tape Turing machines

(PE) Given a deterministic and reversible Turing machine, decide whether its entropy is positive.

Theorem 4.4. *(PE) is undecidable.*

Proof. We prove it by reduction from the halting problem with empty counters of 2-RCM.

Let $C = (\Omega, 2, R)$ be a 2-RCM, with initial configuration $(s_0, (0, 0))$ and final state $s_f \in \Omega$. For this proof we use the Turing machine M_C defined in section 4.2.1. This machine:

1. simulates C on the right side of the tape,
2. is reversible.

We distinguish four types of configurations:

1. Configurations where M_C finds error(s) in its computation and halts.
2. Configurations where M_C goes to unbounded searches of symbols like $<, |, >$.
3. Configurations where M_C has a successful infinite computation of C .
4. Configurations where M_C has a successful finite computation of C .

We slightly modify M_C by adding the rules depicted in figure 4.4. These rules allow a new computation to start if the initial one is achieved. A second simulation will correspond to the machine C starting from $(s_0, (0, 0))$.

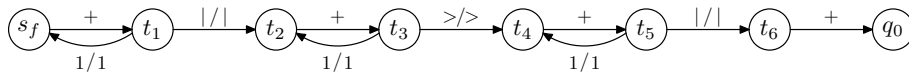


Figure 4.4: Sequence of states added to M_C .

Configurations type 1 have a finite computation, then it is not part of the machine t -shift. Now, the upper complexity of configurations type 2 is 0, by Lemma 4.2, as all the searches are done over symbol 1. Configurations type 3 revisit infinitely the starting cell, therefore its upper complexity is 0. Configurations type 4 will reach state t , and it starts a search of symbol $|$, and then it starts a simulation of the 2-RCM from configuration $(s_0, (0, 0))$.

If the machine C halts from $(s_0, (0, 0))$, M_C can repeat simulations of C several times. More precisely, we can consider a configuration $(s_0, 0, w)$, where $w \in (u + v)^{\mathbb{Z}}$, with $u = 1^m|$ and $v = 1^{|\tau((q_0, 0, u))|}$. If m is taken as the minimum space needed to perform a complete simulation of C from $(s_0, (0, 0))$, M_C will make a simulation of C with the same frequency as the word u appears in w . We thus have a lower bound for the entropy of $S_{M'_C}$:

$$H(M_C) = \lim_n \frac{1}{n} \log(\#(S_{M_C}|_n)) \geq \lim_k \frac{\log 2^k}{k(|\tau((q_0, 0, u))|)} = \frac{\log 2}{|\tau((q_0, 0, u))|} > 0$$

On the other hand, if the machine C does not halt from $(s_0, (0, 0))$, even if a configuration of type 4 reaches s_f , M_C will start a search of symbol $|$. Three cases appear:

- Symbol $|$ is found. In this case, the machine starts a new computation of C from $(s_0, (0, 0))$, the configuration becomes of type 1 or 3, and its upper complexity is 0.
- Symbol $|$ is not found. We are over a configuration of type 2. By Lemma 4.2, its upper complexity is 0.
- A symbol different from $|$ and 1 is found. The machine halts (type 1), therefore this trace is not part of the t -shift.

We conclude that C halts on $(s_0, 0, 0)$ if and only if M_C has positive entropy. □

(EntrC) Given a deterministic, complete and reversible Turing machine, decide whether its entropy is 0.

Corollary 4.1. *(EntrC) is undecidable.*

Proof. Take the same machine $M_C = (Q, \Sigma, \delta)$ from the previous result and 2-RCM C . We use again the technique *reversing the computation*, as described in section 5.3.1, to obtain a complete machine M'_C . In this new machine, configurations of type 1 simply loop into a periodic behavior, always visiting the same cells. Its upper complexity is 0.

As this new machine M'_C works in the same way than M' on every configuration (trace), but in the halting ones, then machine C halts on $(s_0, 0, 0)$ if and only if M'_C has positive entropy. \square



A SMART machine

This chapter is based in a work submitted in 2014 to a journal. It will be presented a certain Turing machine that is aperiodic, transitive for all its dynamical models, minimal for its t -shift, time symmetric, complete and reversible. This machine was created to prove that exists a reversible and complete machine without periodic behavior, proving one conjecture proposed by Jarko Kari and Nicolas Ollinger [28]. It will be also proved that is undecidable the periodicity problem for complete and reversible Turing Machine, proving the second part of the conjecture, by using two techniques to insert machines inside others.

5.1 A small aperiodic complete and reversible Turing Machine (SMART)

5.1.1 The SMART machine.

The machine which is the object of this chapter is described in figure 5.1. We remark the symmetry between states b and d , and between p and q . State b writes the same as d on the tape, but it goes in the opposite direction of d . The analogous occurs with p and q .

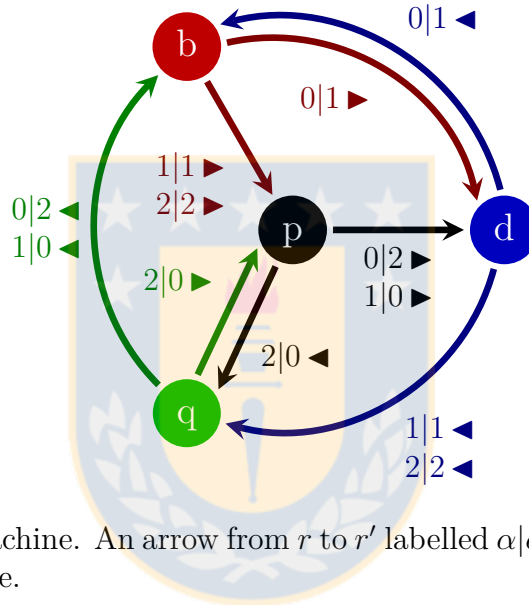


Figure 5.1: The SMART machine. An arrow from r to r' labelled $a|a'c$ represents the instruction $(r, \alpha, \alpha', r', c)$ of the machine.

5.1.2 Basic movements of SMART

The behavior of SMART consists in recursively applying different types of *bounded searches*. These can be described by the next four propositions. They say that the machine finally transverse every block of 0s. But in the proof of these propositions, we can see that it does it by recursive calling smaller bounded searches in a nested way.

$$B(n): (\forall \alpha_+ \in \{1, 2\})(\forall \alpha_* \in \{0, 1, 2\}) \left(\begin{smallmatrix} \alpha_* & 0^n & 0 & \alpha_+ \\ & & b & \end{smallmatrix} \right)^* \left(\begin{smallmatrix} \alpha_* & 0^{n+1} & \alpha_+ \\ & & b & \end{smallmatrix} \right)$$

$$D(n): (\forall \alpha_+ \in \{1, 2\})(\forall \alpha_* \in \{0, 1, 2\}) \left(\begin{smallmatrix} \alpha_+ & 0 & 0^n & \alpha_* \\ & & d & \end{smallmatrix} \right)^* \left(\begin{smallmatrix} \alpha_+ & 0^{n+1} & \alpha_* \\ & & d & \end{smallmatrix} \right)$$

$$P(n): (\forall \alpha_+ \in \{1, 2\}) \left(\begin{smallmatrix} 0 & 0^n & \alpha_+ \\ & & p \end{smallmatrix} \right)^* \left(\begin{smallmatrix} 0^{n+1} & \alpha_+ \\ & p \end{smallmatrix} \right)$$

$$\begin{aligned}
& \begin{pmatrix} 0 & 0 & 0^{n-2} & 0 & \alpha_+ \\ p & & & & \end{pmatrix} \\
& \text{One step} \\
& \begin{pmatrix} 2 & 0 & 0^{n-2} & 0 & \alpha_+ \\ d & & & & \end{pmatrix} \\
& \text{Apply } D(n-1) \\
& \begin{pmatrix} 2 & 0 & 0^{n-2} & 0 & \alpha_+ \\ & & & d & \end{pmatrix} \\
& \text{One step} \\
& \begin{pmatrix} 2 & 0 & 0^{n-2} & 0 & \alpha_+ \\ & & & q & \end{pmatrix} \tag{5.2} \\
& \text{Apply } Q(n-1) \\
& \begin{pmatrix} 2 & 0 & 0^{n-2} & 0 & \alpha_+ \\ q & & & & \end{pmatrix} \\
& \text{One step} \\
& \begin{pmatrix} 0 & 0 & 0^{n-2} & 0 & \alpha_+ \\ p & & & & \end{pmatrix} \\
& \text{Apply } P(n-1) \\
& \begin{pmatrix} 0 & 0 & 0^{n-2} & 0 & \alpha_+ \\ & & & p & \end{pmatrix}
\end{aligned}$$

□

It is important to remark that applying $B(n)$ always implies to apply $B(n-1)$ as the first step, and also $B(n-2)$ and $B(0)$. Analogously, applying $P(n)$ implies to apply $P(n-1)$ and $P(0)$ as the last step. Interestingly, just before applying $P(0)$ by the last time, i.e., 3 steps before finishing $P(n)$, the head is on the rightmost 0 with state p . This will be used in the future. Thus, we define the next two propositions which also hold.

$$P'(n): (\forall \alpha_+ \in \{1, 2\}) \begin{pmatrix} 0 & 0^n & \alpha_+ \\ p & & \end{pmatrix}^* \vdash \begin{pmatrix} 0^n & 0 & \alpha_+ \\ p & & \end{pmatrix}$$

$$Q'(n): (\forall \alpha_+ \in \{1, 2\}) \begin{pmatrix} \alpha_+ & 0^n & 0 \\ & & q \end{pmatrix}^* \vdash \begin{pmatrix} \alpha_+ & 0 & 0^n \\ & & q \end{pmatrix}$$

5.1.3 Aperiodicity

Following [28], we say that a Turing machine M is *aperiodic* if its associated moving tape system T has no periodic point. Let us remark that periodic points in the *moving tape* system can correspond to periodic configurations where the machine makes repetitive movements in a fixed direction. First, we prove the aperiodicity of two particular but important points.

Lemma 5.2. $\begin{pmatrix} \alpha_+ & 0 & 0^n & 1 & 0 \\ p & & & & \end{pmatrix}^* \vdash \begin{pmatrix} \alpha_+ & 0 & 0^{n+1} & 1 \\ p & & & \end{pmatrix}$

Proof.

$$\begin{aligned}
 & \begin{pmatrix} \alpha_+ & 0 & 0^n & 1 & 0 \\ & p & & & \end{pmatrix} \\
 & \qquad \text{Apply } P(n) \\
 & \begin{pmatrix} \alpha_+ & 0 & 0^n & 1 & 0 \\ & p & & & \end{pmatrix} \\
 & \qquad \text{Two steps} \\
 & \begin{pmatrix} \alpha_+ & 0 & 0^n & 0 & 1 \\ & & & b & \end{pmatrix} \tag{5.3} \\
 & \qquad \text{Apply } B(n+1) \\
 & \begin{pmatrix} \alpha_+ & 0 & 0^n & 0 & 1 \\ & b & & & \end{pmatrix} \\
 & \qquad \text{One step} \\
 & \begin{pmatrix} \alpha_+ & 0 & 0^n & 0 & 1 \\ & p & & & \end{pmatrix}
 \end{aligned}$$

□

Lemma 5.3. The semi-infinite configurations $\begin{pmatrix} 0 & 0^w \\ b & \end{pmatrix}$ and $\begin{pmatrix} 0 & 0^w \\ p & \end{pmatrix}$ are not periodic.

Proof. Starting with any of these configurations, the machine makes a few steps (≤ 9), leading to $\begin{pmatrix} \alpha_+ & 0 & 1 & 0^w \\ & p & & \end{pmatrix}$, with $\alpha_+ \in \{1, 2\}$ depending on the initial state (if p then $\alpha_+ = 2$, if b then $\alpha_+ = 1$). Now we can apply Lemma 5.2 to see that the evolution of the machine cannot be periodic. □

In order to generalize aperiodicity to any configuration, we will prove that, in the evolution of every configuration, arbitrary large blocks of 0s appear in a recurrent way.

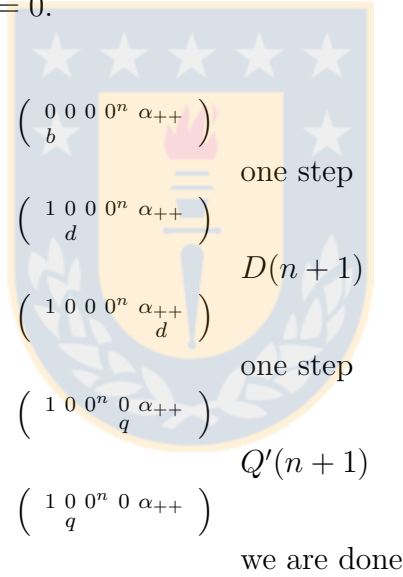
Lemma 5.4. If we define, for every $n \geq 0$, the set $C_n = \{x \mid x \in \begin{pmatrix} \alpha_+ & 0 & 0^n \\ & q & \end{pmatrix} \cup \begin{pmatrix} 0^n & 0 & \alpha_+ \\ & p & \end{pmatrix}\}$, then for every $x \in C_n$, either x or the orbit of x will eventually visit C_m for arbitrary large m .

Proof. We just make the proof for initial state q , since p is symmetrical. Let us start with n maximal such that $x \in C_n$. If there is no maximal n , then x is already on every C_m .

$$\begin{aligned}
 & \begin{pmatrix} \alpha_* & \alpha_+ & 0 & 0^n & \alpha_{++} \\ & & q & & \end{pmatrix} \\
 & \qquad \text{one step} \\
 & \begin{pmatrix} \alpha_* & \alpha_+ & 2 & 0^n & \alpha_{++} \\ & & b & & \end{pmatrix} \\
 & \qquad \text{one step}
 \end{aligned}$$

If $\alpha_+ = 1$ $\left(\begin{array}{c} \alpha_* \ 1 \ 2 \ 0^n \ \alpha_{++} \\ p \end{array} \right)$	If $\alpha_+ = 2$ $\left(\begin{array}{c} \alpha_* \ 2 \ 2 \ 0^n \ \alpha_{++} \\ p \end{array} \right)$
one step	one step
$\left(\begin{array}{c} \alpha_* \ 1 \ 0 \ 0^n \ \alpha_{++} \\ q \end{array} \right)$	$\left(\begin{array}{c} \alpha_* \ 2 \ 0 \ 0^n \ \alpha_{++} \\ q \end{array} \right)$
one step	one step
$\left(\begin{array}{c} \alpha_* \ 0 \ 0 \ 0^n \ \alpha_{++} \\ b \end{array} \right)$	$\left(\begin{array}{c} \alpha_* \ 0 \ 0 \ 0^n \ \alpha_{++} \\ p \end{array} \right)$
If $\alpha_* \neq 0$	$P'(n)$
$\left(\begin{array}{c} \alpha_* \ 0 \ 0 \ 0^n \ \alpha_{++} \\ p \end{array} \right)$	$\left(\begin{array}{c} \alpha_* \ 0 \ 0^n \ 0 \ \alpha_{++} \\ p \end{array} \right)$
$P'(n+1)$	we are done
$\left(\begin{array}{c} \alpha_* \ 0 \ 0^n \ 0 \ \alpha_{++} \\ p \end{array} \right)$	
we are done	

Now we study the case $\alpha_* = 0$.



$\left(\begin{array}{c} 0 \ 0 \ 0 \ 0^n \ \alpha_{++} \\ b \end{array} \right)$	$\left(\begin{array}{c} 0 \ 0 \ 0 \ 0^n \ \alpha_{++} \\ b \end{array} \right)$
one step	one step
$\left(\begin{array}{c} 1 \ 0 \ 0 \ 0^n \ \alpha_{++} \\ d \end{array} \right)$	$\left(\begin{array}{c} 1 \ 0 \ 0 \ 0^n \ \alpha_{++} \\ d \end{array} \right)$
$D(n+1)$	$D(n+1)$
$\left(\begin{array}{c} 1 \ 0 \ 0 \ 0^n \ \alpha_{++} \\ d \end{array} \right)$	$\left(\begin{array}{c} 1 \ 0 \ 0 \ 0^n \ \alpha_{++} \\ d \end{array} \right)$
one step	one step
$\left(\begin{array}{c} 1 \ 0 \ 0^n \ 0 \ \alpha_{++} \\ q \end{array} \right)$	$\left(\begin{array}{c} 1 \ 0 \ 0^n \ 0 \ \alpha_{++} \\ q \end{array} \right)$
$Q'(n+1)$	$Q'(n+1)$
$\left(\begin{array}{c} 1 \ 0 \ 0^n \ 0 \ \alpha_{++} \\ q \end{array} \right)$	$\left(\begin{array}{c} 1 \ 0 \ 0^n \ 0 \ \alpha_{++} \\ q \end{array} \right)$
we are done	we are done

□

Theorem 5.1. *The SMART machine has no periodic points.*

Proof. Consider an arbitrary configuration. After less than 9 steps, the head will be reading a 0 symbol in either state q or p , after which, by propositions P' and Q' , it arrives to one of the sets C_n described in Lemma 5.4. The amount of 0s will grow then, expanding to the right or to the left. At some point, the machine will either reach a configuration of the form $\left(\begin{array}{c} 0 \ 0^w \\ r \end{array} \right)$, with $r \in \{b, p\}$ (or its symmetric), which we know to be aperiodic from Lemma 5.2, or it will pass by an infinite sequence of configurations of the form $\left(\begin{array}{c} 0 \ 0^{n-1} \ \alpha_+ \\ r \end{array} \right)$ with $r \in \{b, p\}$ (or its symmetric), implying that its behaviour is not periodic. □

Corollary 5.1. *There exists a complete reversible Turing machine without any periodic point.*

5.2 Other properties of the SMART machine

5.2.1 Some more lemmas

We will denote some other results about the behavior of this machine to simplify the proofs of minimality and substitution.

Lemma 5.5. The SMART machine is time-symmetric.

Proof. Using involutions: $h_{\Sigma}(0) = 0$, $h_{\Sigma}(1) = 2$, $h_Q(d) = q$ and $h_Q(b) = p$, we will have that the SMART machine is time-symmetric as can be seen on the diagram of the inverse machine in Fig. 5.2. □

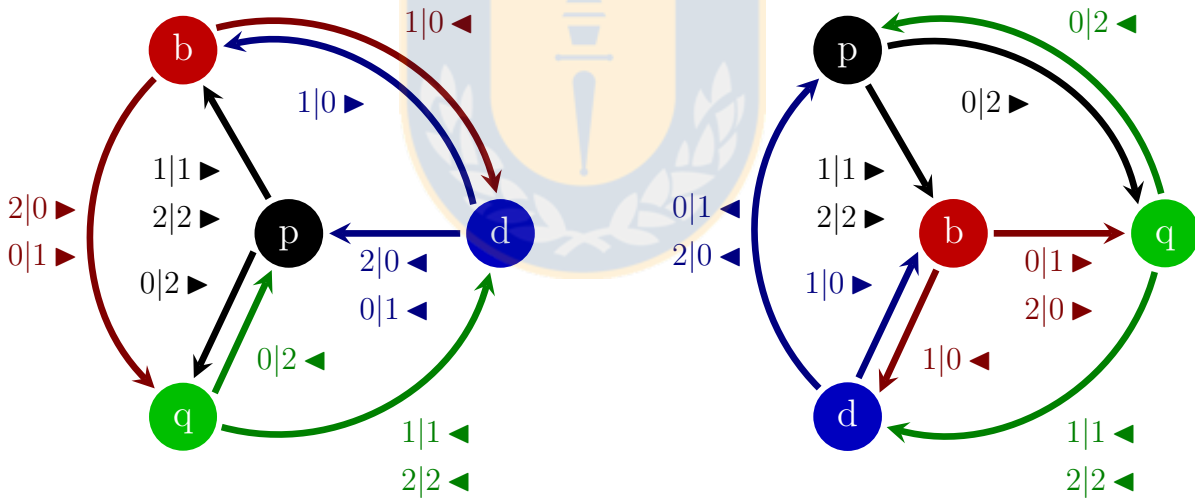


Figure 5.2: Two representations of the inverse of the SMART machine

With the previous result, we can now prove that every finite configuration can be reached from a block of 0s of the appropriate size.

Lemma 5.6. For every finite word $v' \in \{0, 1, 2\}^*$ of length n , and every $i \in \{1, \dots, n\}$, there exist $k_1, k_2 \in \mathbb{N}$ and $r \in Q$ such that $\left(\begin{smallmatrix} 2 & 0^{n'} & 0 & 2 \\ & & & b \end{smallmatrix} \right)^* \vdash \left(\begin{smallmatrix} 2^{k_1} & v'_1 & \dots & v'_i & \dots & v'_n & 2^{k_2} \\ & r & & & & & \end{smallmatrix} \right)$, where $n' = k_1 + k_2 + n - 3$.

Proof. First, we will use the fact that the SMART machine is time-symmetric, so applying $h \circ T^t \circ h$ is the same as applying T^{-t} , for any time $t \in \mathbb{N}$. So now we just have to prove that $h \left(\begin{smallmatrix} 2^{k_1} & v'_1 & \dots & v'_i & \dots & v'_n & 2^{k_2} \\ & r & & & & & \end{smallmatrix} \right) = \left(\begin{smallmatrix} 1^{k_1} & h_\Sigma(v'_1) & \dots & \dots & \dots & h_\Sigma(v'_n) & 1^{k_2} \\ & & & h_Q(r) & & & \end{smallmatrix} \right)$ will eventually reach $h \left(\begin{smallmatrix} 2 & 0^{n'} & 0 & 2 \\ & b & & \end{smallmatrix} \right) = \left(\begin{smallmatrix} 1 & 0^{n'} & 0 & 1 \\ & p & & \end{smallmatrix} \right)$.

Based on the proof of Theorem 5.1, we know that we can generate an increasing amount of 0 symbols in the tape. For this reason, we know that $\left(\begin{smallmatrix} 1 & h_\Sigma(v'_1) & \dots & \dots & \dots & h_\Sigma(v'_n) & 1 \\ & & & h_Q(r) & & & \end{smallmatrix} \right)$ will eventually reach one of the configurations considered in Lemma 5.4 with a block of 0s that will grow until arriving to one of the next configurations:

$$\left(\begin{smallmatrix} 1 & 0^j & 0 & v_1 & \dots & v_l & 1 \\ & p & & & & & \end{smallmatrix} \right) \text{ or } \left(\begin{smallmatrix} 1 & 0 & 0^j & v_1 & \dots & v_l & 1 \\ & q & & & & & \end{smallmatrix} \right) \text{ or } \left(\begin{smallmatrix} 1 & v_1 & \dots & v_l & 0 & 0^j & 1 \\ & & & & q & & \end{smallmatrix} \right) \text{ or } \left(\begin{smallmatrix} 1 & v_1 & \dots & v_l & 0^j & 0 & 1 \\ & & & & p & & \end{smallmatrix} \right).$$

Thus, applying either $Q(0)$ or $P(0)$ we arrive to one of the next situations:

$$\begin{matrix} (i) & & (ii) & & (iii) & & (iv) \\ \left(\begin{smallmatrix} 1 & 0^j & 0 & v_1 & \dots & v_l & 1 \\ & p & & & & & \end{smallmatrix} \right) & \text{or} & \left(\begin{smallmatrix} 1 & 0 & 0^j & v_1 & \dots & v_l & 1 \\ & q & & & & & \end{smallmatrix} \right) & \text{or} & \left(\begin{smallmatrix} 1 & v_1 & \dots & v_l & 0 & 0^j & 1 \\ & & & & q & & \end{smallmatrix} \right) & \text{or} & \left(\begin{smallmatrix} 1 & v_1 & \dots & v_l & 0^j & 0 & 1 \\ & & & & p & & \end{smallmatrix} \right), \end{matrix} \quad (5.4)$$

for some $v \in \{0, 1, 2\}^{n-j-1}$, and $v_1 \neq 0$ in situations (i) and (ii), and $v_l \neq 0$ in the other two.

In order to reach $\left(\begin{smallmatrix} 1 & 0^{n'} & 0 & 1 \\ & p & & \end{smallmatrix} \right)$, we will need a certain amount of 1 symbols at the left or right of the initial configuration; this amount depends on v . Let $k(v)$ be the function that gives the amount of non-0 symbols in v . As before, we will do the proof only for configurations of the form (i) and (ii).

Case (i). We will prove first that every iteration will replace at least one element of v for a 0.

- Case (i).1 $v_2 \neq 0$ or $v_1 = 2$.

$$\left(1^{k(v)} \ 1 \ 0 \ 0^j \ \underset{p}{v_1} \ v_2 \ \dots \ v_l \ 1 \right)$$

Regardless of the value of $v_1 \neq 0$

in 1 or 2 steps we reach the next, with $i = 0$ or 1

$$\left(1^{k(v)} \ 1 \ 0^{j-i+1} \ 0 \ 0^i \ v_2 \ \dots \ v_l \ 1 \right)$$

Apply $Q(j - i)$

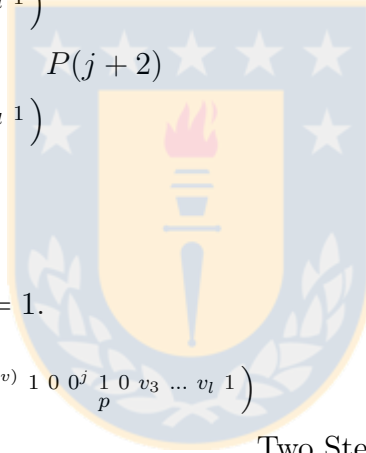
$$\left(1^{k(v)} \ \underset{q}{1} \ 0^j \ 0 \ 0 \ v_2 \ \dots \ v_l \ 1 \right)$$

(5.5)

Two steps

$$\left(1^{k(v)} \ 0 \ 0^j \ 0 \ 0 \ v_2 \ \dots \ v_l \ 1 \right)$$

$$\left(1^{k(v)} \ 0 \ 0^j \ 0 \ 0 \ \underset{p}{v_2} \ \dots \ v_l \ 1 \right)$$



- Case (i).2 $v_2 = 0$ and $v_1 = 1$.

$$\left(1^{k(v)} \ 1 \ 0 \ 0^j \ \underset{p}{1} \ 0 \ v_3 \ \dots \ v_l \ 1 \right)$$

Two Steps

$$\left(1^{k(v)} \ 1 \ 0 \ 0^j \ 0 \ 1 \ v_3 \ \dots \ v_l \ 1 \right)$$

Apply $B(j + 1)$

$$\left(1^{k(v)} \ \underset{b}{1} \ 0 \ 0^j \ 0 \ 1 \ v_3 \ \dots \ v_l \ 1 \right)$$

(5.6)

One Step

$$\left(1^{k(v)} \ 1 \ 0 \ 0^j \ 0 \ 1 \ v_3 \ \dots \ v_l \ 1 \right)$$

Apply $P(j + 1)$

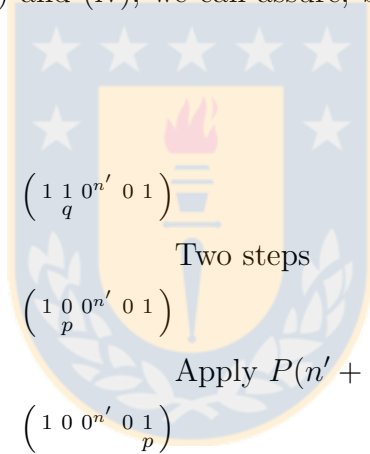
$$\left(1^{k(v)} \ 1 \ 0 \ 0^j \ 0 \ \underset{p}{1} \ v_3 \ \dots \ v_l \ 1 \right)$$

Finally, we repeat this steps l times and we are done.

Case (ii)

$$\begin{aligned}
 & \left(\begin{array}{cccccccc} 1^{k(v)} & 1 & 0 & 0^j & v_1 & v_2 & \dots & v_l & 1 \end{array} \right) \\
 & \qquad \qquad \qquad \text{Two steps} \\
 & \left(\begin{array}{cccccccc} 1^{k(v)} & 0 & 0 & 0^j & v_1 & v_2 & \dots & v_l & 1 \end{array} \right) \\
 & \qquad \qquad \qquad P(j+1) \\
 & \left(\begin{array}{cccccccc} 1^{k(v)} & 0 & 0 & 0^j & v_1 & v_2 & \dots & v_l & 1 \end{array} \right) \\
 & \qquad \qquad \qquad \text{Which reduces to case (i)}
 \end{aligned} \tag{5.7}$$

In this way, we have proved that, for (i) and (ii), we will always reach $\left(\begin{array}{cccc} 1 & 0^{n'} & 0 & 1 \\ q & & p & \end{array} \right)$, with $n' = j + l + k(v)$. For cases (iii) and (iv), we can assure, by symmetry, that the machine will reach $\left(\begin{array}{cccc} 1 & 0^{n'} & 0 & 1 \\ q & & p & \end{array} \right)$, then:



$$\begin{aligned}
 & \left(\begin{array}{cccc} 1 & 1 & 0^{n'} & 0 & 1 \\ q & & & & \end{array} \right) \\
 & \qquad \qquad \qquad \text{Two steps} \\
 & \left(\begin{array}{cccc} 1 & 0 & 0^{n'} & 0 & 1 \\ p & & & & \end{array} \right) \\
 & \qquad \qquad \qquad \text{Apply } P(n'+1) \\
 & \left(\begin{array}{cccc} 1 & 0 & 0^{n'} & 0 & 1 \\ q & & p & & \end{array} \right)
 \end{aligned} \tag{5.8}$$

Concluding that, for the last two cases, we need just one additional 1 symbol to reach the desired configuration. \square

Lemma 5.7. The orbits of configurations $\left(\begin{array}{cc} 0 & 0^\omega \\ r_1 & \end{array} \right)$ and $\left(\begin{array}{cc} \omega & 0 \\ & r_2 \end{array} \right)$ are dense in X , with $r_1 \in \{b, p\}$ and $r_2 \in \{d, q\}$.

Proof. We just need to prove that any finite configuration can be reached from $\left(\begin{array}{cc} 0 & 0^\omega \\ b & \end{array} \right)$. The other cases can be proved by symmetry and time symmetry. Since any finite configuration can be reached from $\left(\begin{array}{cc} 2 & 0^n & 0 & 2 \\ & b & & \end{array} \right)$ for some n , we just need to prove that $\left(\begin{array}{cc} 0 & 0^\omega \\ b & \end{array} \right)$ can reach $\left(\begin{array}{cc} 2 & 0^n & 0 & 2 \\ & b & & \end{array} \right)$, for any $n \in \mathbb{N}$.

$$\begin{array}{l}
\begin{pmatrix} 0 & 0^\omega \\ b & \end{pmatrix} \\
\text{One step} \\
\begin{pmatrix} 1 & 0 & 0^\omega \\ d & \end{pmatrix} \\
\text{Apply } D(n+3) \\
\begin{pmatrix} 1 & 0 & 0^{n+1} & 0 & 0 & 0^\omega \\ & & & d & \end{pmatrix} \\
\text{One step} \\
\begin{pmatrix} 1 & 0 & 0^{n+1} & 0 & 1 & 0^\omega \\ & & & b & \end{pmatrix} \\
\text{Apply } B(n+2) \\
\begin{pmatrix} 1 & 0 & 0 & 0^{n+1} & 1 & 0^\omega \\ b & \end{pmatrix} \\
\text{Two steps} \\
\begin{pmatrix} 1 & 2 & 0 & 0^{n+1} & 1 & 0^\omega \\ & & d & \end{pmatrix} \\
\text{Apply } D(n+1) \\
\begin{pmatrix} 1 & 2 & 0^{n+1} & 0 & 1 & 0^\omega \\ & & & d & \end{pmatrix} \\
\text{Two steps} \\
\begin{pmatrix} 1 & 2 & 0^n & 0 & 2 & 1 & 0^\omega \\ & & & b & \end{pmatrix}
\end{array}$$

□

Corollary 5.2. *The SMART machine is transitive as well as its t -shift.*

Theorem 5.2. *The SMART machine is minimal as well as its t -shift.*

Proof. As we know from Lemma 5.4 and Theorem 5.1, we can “create” an arbitrary amount of 0s starting from any initial configuration. Now, from Lemma 5.6, if the correct amount of 0 is provided, we can reach any finite configuration. This proves that every point is transitive, and so SMART is minimal. Since its t -shift is a factor of (X, M) , it inherits minimality. □

5.2.2 The t -shift is substitutive

In this part, we will prove that the t -shift is not only minimal, but also substitutive. For this, we will present a substitution and, with a pair of results, prove that the t -shift of SMART is the closure of the shift orbit of a fixed point of that substitution.

First, we recursively define the following functions.

- $B : \{1, 2\} \times \mathbb{N} \rightarrow (Q \times \Sigma)^*$

$$B(\alpha_+, n) = B(\alpha_+, n-1) \begin{matrix} 0 \\ b \end{matrix} D(1, n-1) \begin{matrix} \alpha_+ \\ d \end{matrix} Q(1, n-1) \begin{matrix} 1 \\ q \end{matrix}$$

$$B(\alpha_+, 0) = \begin{matrix} 0 & \alpha_+ & 1 \\ b & d & q \end{matrix}$$

- $D : \{1, 2\} \times \mathbb{N} \rightarrow (Q \times \Sigma)^*$
 $D(\alpha_+, n) = D(\alpha_+, n-1) \begin{smallmatrix} 0 \\ d \end{smallmatrix} B(1, n-1) \begin{smallmatrix} \alpha_+ \\ b \end{smallmatrix} P(1, n-1) \begin{smallmatrix} 1 \\ p \end{smallmatrix}$
 $D(\alpha_+, 0) = \begin{smallmatrix} 0 & \alpha_+ & 1 \\ d & b & p \end{smallmatrix}$
- $P : \{1, 2\} \times \mathbb{N} \rightarrow (Q \times \Sigma)^*$
 $P(\alpha_+, n) = \begin{smallmatrix} 0 \\ p \end{smallmatrix} D(2, n-1) \begin{smallmatrix} \alpha_+ \\ d \end{smallmatrix} Q(2, n-1) \begin{smallmatrix} 2 \\ q \end{smallmatrix} P(\alpha_+, n-1)$
 $P(\alpha_+, 0) = \begin{smallmatrix} 0 & \alpha_+ & 2 \\ p & d & q \end{smallmatrix}$
- $Q : \{1, 2\} \times \mathbb{N} \rightarrow (Q \times \Sigma)^*$
 $Q(\alpha_+, n) = \begin{smallmatrix} 0 \\ q \end{smallmatrix} B(2, n-1) \begin{smallmatrix} \alpha_+ \\ b \end{smallmatrix} P(2, n-1) \begin{smallmatrix} 2 \\ p \end{smallmatrix} Q(\alpha_+, n-1)$
 $Q(\alpha_+, 0) = \begin{smallmatrix} 0 & \alpha_+ & 2 \\ q & b & p \end{smallmatrix}$

Lemma 5.8. $B(\alpha_+, n)$ is the trace corresponding to applying proposition $B(n)$ to $\left(\begin{smallmatrix} \alpha_* & 0^n & 0 \\ & b & \alpha_+ \end{smallmatrix} \right)$ until $\left(\begin{smallmatrix} \alpha_* & 0^{n+1} & \alpha_+ \\ & b & \end{smallmatrix} \right)$. The analogous goes for $D(\alpha_+, n)$, $P(\alpha_+, n)$ and $Q(\alpha_+, n)$.

Proof. It is enough to see the proof of lemma 5.1 and take the trace. □

Now, let us define the substitution.

$$\phi : (Q \times \Sigma)^* \rightarrow (Q \times \Sigma)^*$$

$$\phi \left(\begin{smallmatrix} 0 \\ b \end{smallmatrix} \right) = \begin{smallmatrix} 0 & 0 & 1 & 1 \\ b & d & b & p \end{smallmatrix} = \begin{smallmatrix} 0 \\ b \end{smallmatrix} D(1, 0)$$

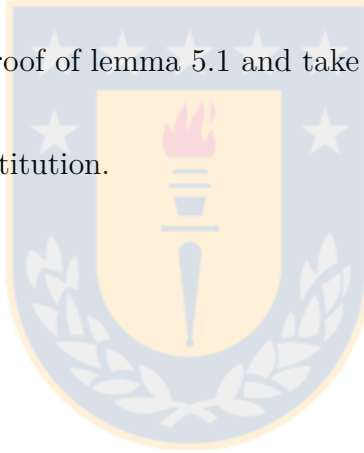
$$\phi \left(\begin{smallmatrix} \alpha_+ \\ b \end{smallmatrix} \right) = \begin{smallmatrix} \alpha_+ \\ b \end{smallmatrix}$$

$$\phi \left(\begin{smallmatrix} 0 \\ p \end{smallmatrix} \right) = \begin{smallmatrix} 0 & 0 & 2 & 1 \\ p & d & b & p \end{smallmatrix} = \begin{smallmatrix} 0 \\ p \end{smallmatrix} D(2, 0)$$

$$\phi \left(\begin{smallmatrix} \alpha_+ \\ p \end{smallmatrix} \right) = \begin{smallmatrix} 0 & \alpha_+ & 2 & \alpha_+ \\ p & d & q & p \end{smallmatrix} = P(\alpha_+, 0) \begin{smallmatrix} \alpha_+ \\ p \end{smallmatrix}$$

$$\phi \left(\begin{smallmatrix} 0 \\ d \end{smallmatrix} \right) = \begin{smallmatrix} 0 & 0 & 1 & 1 \\ d & b & d & q \end{smallmatrix} = \begin{smallmatrix} 0 \\ d \end{smallmatrix} B(1, 0)$$

$$\phi \left(\begin{smallmatrix} \alpha_+ \\ d \end{smallmatrix} \right) = \begin{smallmatrix} \alpha_+ \\ d \end{smallmatrix}$$



$$\phi\left(\begin{smallmatrix} 0 \\ q \end{smallmatrix}\right) = \begin{smallmatrix} 0 & 0 & 2 & 1 \\ q & b & d & q \end{smallmatrix} = \begin{smallmatrix} 0 \\ q \end{smallmatrix} B(2, 0)$$

$$\phi\left(\begin{smallmatrix} \alpha_+ \\ q \end{smallmatrix}\right) = \begin{smallmatrix} 0 & \alpha_+ & 2 & \alpha_+ \\ q & b & p & q \end{smallmatrix} = Q(\alpha_+, 0) \begin{smallmatrix} \alpha_+ \\ q \end{smallmatrix}$$

Lemma 5.9. For all $\alpha_+ \in \{1, 2\}$ and for all $n \in \mathbb{N}$

$$B(\alpha_+, n) = B(\alpha_+, 0)\phi(B(\alpha_+, n-1))$$

$$D(\alpha_+, n) = D(\alpha_+, 0)\phi(D(\alpha_+, n-1))$$

$$P(\alpha_+, n) = \phi(P(\alpha_+, n-1))P(\alpha_+, 0)$$

$$Q(\alpha_+, n) = \phi(Q(\alpha_+, n-1))Q(\alpha_+, 0)$$

Proof. It is enough to prove the lemma for $B(\alpha_+, n)$ and $P(\alpha_+, n)$, the other cases can be proved by symmetry.

$$\begin{aligned} B(\alpha_+, 0)\phi(B(\alpha_+, n)) &= B(\alpha_+, 0)\phi(B(\alpha_+, n-1) \begin{smallmatrix} 0 \\ b \end{smallmatrix} D(1, n-1) \begin{smallmatrix} \alpha_+ \\ d \end{smallmatrix} Q(1, n-1) \begin{smallmatrix} 1 \\ q \end{smallmatrix}) \\ &= B(\alpha_+, 0)\phi(B(\alpha_+, n-1) \begin{smallmatrix} 0 \\ b \end{smallmatrix} D(1, 0)\phi(D(1, n-1)) \begin{smallmatrix} \alpha_+ \\ d \end{smallmatrix} \phi(Q(1, n-1))Q(1, 0) \begin{smallmatrix} 1 \\ q \end{smallmatrix}) \\ &= B(\alpha_+, n) \begin{smallmatrix} 0 \\ b \end{smallmatrix} D(1, n) \begin{smallmatrix} \alpha_+ \\ d \end{smallmatrix} Q(1, n) \begin{smallmatrix} 1 \\ q \end{smallmatrix} = B(\alpha_+, n+1) \end{aligned}$$

$$\begin{aligned} \phi(P(\alpha_+, n))P(\alpha_+, 0) &= \phi\left(\begin{smallmatrix} 0 \\ p \end{smallmatrix} D(2, n-1) \begin{smallmatrix} \alpha_+ \\ d \end{smallmatrix} Q(2, n-1) \begin{smallmatrix} 2 \\ q \end{smallmatrix} P(\alpha_+, n-1)\right)P(\alpha_+, 0) \\ &= \begin{smallmatrix} 0 \\ p \end{smallmatrix} D(2, 0)\phi(D(2, n-1)) \begin{smallmatrix} \alpha_+ \\ d \end{smallmatrix} \phi(Q(2, n-1))Q(2, 0) \begin{smallmatrix} 2 \\ q \end{smallmatrix} \phi(P(\alpha_+, n-1))P(\alpha_+, 0) \\ &= \begin{smallmatrix} 0 \\ p \end{smallmatrix} D(2, n) \begin{smallmatrix} \alpha_+ \\ d \end{smallmatrix} Q(2, n) \begin{smallmatrix} 2 \\ q \end{smallmatrix} P(\alpha_+, n) = P(\alpha_+, n+1) \end{aligned}$$

□

Theorem 5.3. *The t -shift of SMART is the closure of a fixed point of substitution ϕ .*

Proof. It is enough to prove that $\phi^n\left(\begin{smallmatrix} 0 \\ b \end{smallmatrix}\right) = \begin{smallmatrix} 0 \\ b \end{smallmatrix} D(1, n-1)$ for all $n \in \mathbb{N}$, because, from lemma 5.8, $\begin{smallmatrix} 0 \\ b \end{smallmatrix} D(1, n-1)$ is the trace of $\left(\begin{smallmatrix} 0 \\ b \end{smallmatrix}\right)^{\omega}$ over the first steps and, as the configuration is transitive, the orbit of this configuration is dense. We will prove it by induction.

$$\text{Base of induction: } \phi\left(\begin{smallmatrix} 0 \\ b \end{smallmatrix}\right) = \begin{smallmatrix} 0 & 0 & 1 & 1 \\ b & d & b & p \end{smallmatrix} = \begin{smallmatrix} 0 \\ b \end{smallmatrix} D(1, 0)$$

Induction hypothesis: $\phi^n\left(\begin{smallmatrix} 0 \\ b \end{smallmatrix}\right) = \begin{smallmatrix} 0 \\ b \end{smallmatrix}D(1, n-1)$

Induction thesis:

$$\begin{aligned}
 \phi^{n+1}\left(\begin{smallmatrix} 0 \\ b \end{smallmatrix}\right) &= \phi\left(\phi^n\left(\begin{smallmatrix} 0 \\ b \end{smallmatrix}\right)\right) // \text{Induction hypothesis} \\
 &= \phi\left(\begin{smallmatrix} 0 \\ b \end{smallmatrix}D(1, n-1)\right) \\
 &= \begin{smallmatrix} 0 \\ b \end{smallmatrix}D(1, 0)\phi(D(1, n-1)) // \text{Lemma 5.9} \\
 &= \begin{smallmatrix} 0 \\ b \end{smallmatrix}D(1, n)
 \end{aligned} \tag{5.9}$$

□

5.3 An application of SMART

We will prove the second part of the conjecture in [28]: “it is undecidable whether a given complete RTM admits a periodic configuration”.

In order to ease the understanding of the proof, we will present two key proof techniques.

5.3.1 Proof techniques

Reversing the computation

This technique is used in [28]. It takes a reversible Turing machine $M = (Q, \Sigma, \delta)$, and creates a new reversible and complete machine $M' = (Q \times \{-, +\}, \Sigma, \delta')$, where $(r, +)$ and $(r, -)$ states represent M in state q running forwards or backwards in time, respectively. If at some iteration no instruction of M can be applied, the time direction is switched.

It is noteworthy to say that we can choose to switch the direction on just a set of pairs (state,symbol), defining in this way a possibly incomplete machine. For example, we can just switch the sign on the halting state of M ; then M' will come back to the initial configuration if M halts from it.

Embedding

This technique consists in inserting a machine M inside a machine N to produce a new machine \bar{N} , in such a way that some property of M will be translated into a property of \bar{N} . In order to do this, some states of N are duplicated and connected to particular states of M .

For example, in order to insert $M = (Q, \Sigma, \delta)$ inside $N = (Q', \Sigma, \delta')$, we select a state $r \in Q'$ and split it into two new states, r' and r'' , and connect the input instructions of r to r' and the outputs to r'' . Now one can connect r' with the starting state of M , and r'' with the halting state of M . The connection can be done by a no movement and no writing instruction $((r', *, *, r_0, 0))$. We can also connect other states of M to those of N . In order to do this, we can use another state from N or we can split one of the new states r' or r'' into series.

5.3.2 Undecidability of the aperiodicity of complete reversible Turing machines

In [28], it is proved that the aperiodicity of (non-complete) reversible Turing machines is undecidable. There the proof is performed by reduction from the **reachability problem of aperiodic reversible Turing machines**: *Given an aperiodic and reversible Turing machine $M = (Q, \Sigma, \delta)$ and two states r_1, r_2 , decide whether from r_1 M can reach r_2 or not*, which is proved undecidable in the same paper. The proof uses the technique of “reversing the computation” between r_1 and r_2 in order to define a reversible machine which has a periodic point if and only if (M, r_1, r_2) satisfies *reachability*. Now we will combine this idea with the technique of “embedding” and the SMART machine to define a complete reversible machine with the same characteristic.

Theorem 5.4. *It is undecidable whether a given complete RTM admits a periodic orbit.*

Proof. Let us take an aperiodic RTM machine $M = (Q, \Sigma, \delta)$ and two states $r_1, r_2 \in Q$. Let us suppose that M has m defective states and n error states. Now, let us remove all transitions starting at state r_2 and all transitions arriving to state r_1 . Using the technique of “reversing the computation”, we create a new RTM machine $M' = (Q \times \{+, -\}, \Sigma, \delta')$ such that machine M is simulated forwards and backwards in time as we explained before, but now the direction is switched from - to + only in state r_1 and from + to - in state r_2 .

The machine M' is next duplicated to obtain two machines M'_1 and M'_2 . We will denote their forward and backward parts by M'_i+ and M'_i- respectively.

The embedding into SMART is performed as follows. We invite the reader to look at figure 6.6 for a better understanding. We first split two states of SMART as many times as needed to have the necessary connections. We will denote by q'_i, q''_i , with $i \in \{1, \dots, m\}$ the set of split states of the first group, and by p'_j and p''_j , with $j \in \{1, \dots, n\}$, the set of split states of the second group.

- States q'_i are connected to defective states of M'_1+ and states p'_j are connected to M'_1- .
- Error states of machine M'_1 are directly connected to defective states of machine M'_2 (going from M'_1+ to M'_2- , and from M'_1- to M'_2+).
- Error states of machine M'_2- are connected to states q''_i , and error states of M'_2+ are connected to p''_j .

The obtained machine will be called \overline{SMART} .

We first remark that if the machine M can reach r_2 from r_1 , then \overline{SMART} has a periodic point. Now let us suppose that r_2 is not reachable from r_1 through M .

This new machine is constructed such that, if we enter M'_1 through one of the split states r'_i and we exit, then we exit by r''_i and we find that the tape and the position of the head are not modified. We call this property *innocuity* of the embedding.

Let us prove innocuity for the split states q'_i (the proof for states p'_j is analogous). If we enter M'_1+ by state q'_i we have three possibilities:

- The first one is to stay inside M'_1+ and never exit.
- The second is to go to M'_1- through state r_2 ; then the dynamics is reversed and we are forced to go to M'_2+ . Now the dynamics is repeated and we go again to state r_2 of M'_2 to finally exit by state q''_i . As any computation is next reversed, the tape and the position of the head do not change.
- The third is to go through error states of M'_1+ to M'_2- . Here the computation is reversed to exit by state q''_i . Again the tape and the position of the head do not change.

Thus, if we start on a state of SMART, the SMART dynamics is respected (except in the case we enter an infinite computation of M at some moment, but since M is aperiodic, this is not a problem). Therefore, since SMART is aperiodic, no periodic points appear. Let us study now the case when we start inside M . Three possibilities appear again.

- The first one is to exit the M'_i system. Then we arrive to the previous case, and we already know that the dynamics will not be periodic.
- The second is to fall into an infinite computation inside one of the parts of M'_1 or M'_2 . This again cannot be periodic because M has no periodic points.
- The third is to move internally through the different machines M'_1+ , M'_1- , M'_2+ and M'_2- . But, by construction, the only way of doing this is to alternate between M'_i+ and M'_i- , and this needs to go from r_1 to r_2 , which is supposed impossible.

We conclude that r_2 is not reachable from r_1 by M , if and only if \overline{SMART} has no periodic points. \square

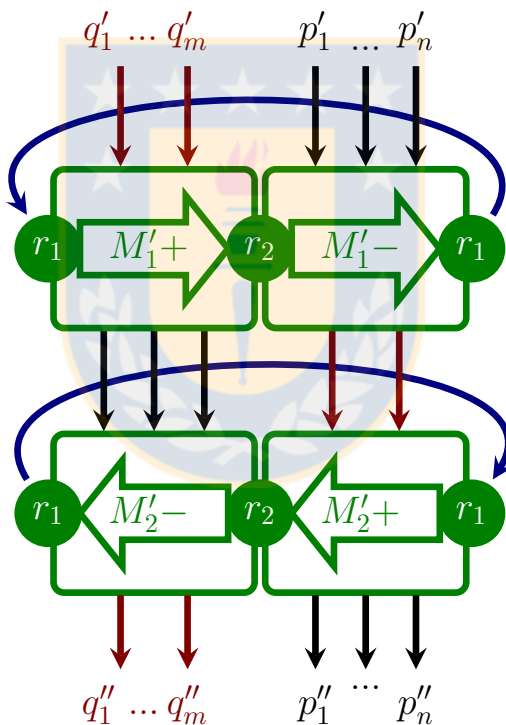


Figure 5.3: Embedding used in the proof of Theorem 5.4

Transitivity and Computability in Turing Machine dynamical systems

In this chapter will be presented the study for topological transitivity in the dynamical models for Turing machines. Blocking words (as defined in Chapter 1) take an important role in differentiate topological transitivity in different dynamical models. It will also prove that transitivity and minimality are undecidable properties for dynamical models of Turing machines.

This chapter is based in an accepted article [48] published in the post-proceedings of Mathematical Foundations of Computer Science 2015, Lecture Notes on Computer Science, 2015.

6.1 The universe of machines with transitive t -shift

Transitivity in Turing machines models has relation with various other properties. One particularity of Turing models is the relation between transitivity and periodic points of T_h . In these points, the head is enclosed in a finite part of the tape. Any perturbation of the configuration that does not affect this part of the tape will not perturb the head. Thus, no periodic point can be attained by a non periodic orbit, and the system cannot be transitive. Moreover, when T_h has a periodic point, transitivity is excluded both from (X_t, T_t) and (S_t, σ) [15].

Proposition 6.1. Given a Turing machine M , the next implications hold.

$$(X_h, T_h) \text{ transitive} \xrightarrow{1} (X_t, T_t) \text{ transitive} \xrightarrow{2} (S_t, \sigma) \text{ transitive}$$

Proof. $(\Rightarrow)_1$ Any finite configuration of X_t corresponds to several finite configurations of X_h , thus if a point exists that visits any finite configuration of X_h , the same point will visit any finite configuration of X_t .

$(\Rightarrow)_2$ (S_t, σ) is a factor of (X_t, T_t) thus it inherits its transitivity. □

Blocking words are also relevant to transitivity as the next propositions show. The idea of a “blocking” configuration that avoids the head from going beyond some limit appears in several contexts, and it is related to stability and information travel. The next definition gives a concept that is results to be significant for transitivity, as we will see below. We state it in the original model of Turing machines.

Proposition 6.2. If M has a blocking word, then (X_h, T_h) is not transitive.

Proof. If $M = (Q, \Sigma, \delta)$ has a blocking word to the left $(r, 0, u)$, then no extension of $.ru$ can visit a finite configuration of the form $(r'\alpha.u') \in X_h$, at any time, for any $r' \in Q$, $\alpha \in \Sigma$, $u' \in \Sigma^*$. This means that (X_h, T_h) is not transitive. □

Proposition 6.3. If M has no blocking word, then the next equivalence holds.

$$(X_t, T_t) \text{ is transitive} \Leftrightarrow (S_t, \sigma) \text{ is transitive}$$

Proof. The left to right implication was already proved, thus let us prove the converse. Let us assume that (S_t, σ) is transitive and that M has no blocking word. Let (u, r, u') and (v, r', v')

be two finite configurations of X_t . Since M has no blocking words, there exist finite extensions $(su, r, u's')$ and $(wv, r', v'w')$ such that the head visits all the cells of (u, r, u') and (v, r', v') respectively. Let us consider now the words $\tau(su, r, u's')$ and $\tau(wv, r', v'w')$. Since (S_t, σ) is transitive, there exists a point $(x, r, y) \in X_t$ such that $\tau(x, r, y)$ starts with $\tau(su, r, u's')$ and contains $\tau(wv, r', v'w')$ afterward. As we have commented, $\tau(su, r, u's')$ determines the state of all the cells that the head visits when producing it, then (x, r, y) needs to be an extension of (u, r, u') . By the same reason, there exist a time t such that $T_t^t(x, r, y)$ is an extension of (v, r', v') , which concludes the proof. \square

With all of these results, we can depict the universe of Turing machines with a transitive trace-shift as figure 6.1 shows. None of the classes are empty, in the next four section we exhibit examples at each of them.

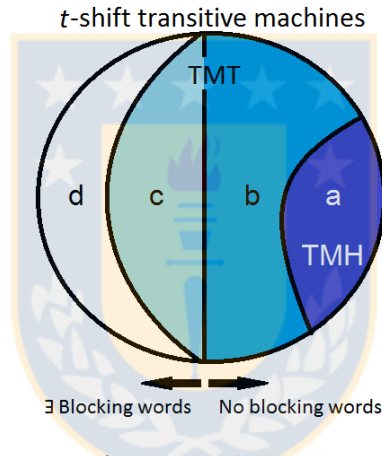


Figure 6.1: Universe of the topologically transitive machines.

6.1.1 Machine of type a : Transitive on TMH model

The machine in figure 6.2 is called *SMART* machine, it has several strong properties as it is shown in Chapter 5. In particular, its TMT system is minimal and trace-shift is substitutive. Now, we will prove that its TMH model is also transitive.

Basic movements of SMART

The behavior of SMART can be described by the next four propositions, as it was proven in Chapter 5, which takes values for any $n \in \mathbb{N}$. They basically says that the head glides over the lagoons of 0s either to the right or to the left, depending on its states and the surrounding symbol.

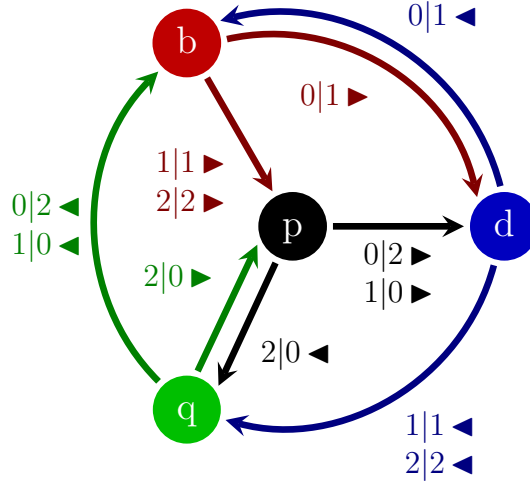


Figure 6.2: The SMART machine.

$$B(n): (\forall s_+ \in \{1, 2\})(\forall s_* \in \{0, 1, 2\}) \left(\begin{smallmatrix} s_* & 0^n & 0 & s_+ \\ & & b & \end{smallmatrix} \right)^* \vdash \left(\begin{smallmatrix} s_* & 0^{n+1} & s_+ \\ & & b \end{smallmatrix} \right)$$

$$D(n): (\forall s_+ \in \{1, 2\})(\forall s_* \in \{0, 1, 2\}) \left(\begin{smallmatrix} s_+ & 0 & 0^n & s_* \\ & & d & \end{smallmatrix} \right)^* \vdash \left(\begin{smallmatrix} s_+ & 0^{n+1} & s_* \\ & & d \end{smallmatrix} \right)$$

$$P(n): (\forall s_+ \in \{1, 2\}) \left(\begin{smallmatrix} 0 & 0^n & s_+ \\ p & & \end{smallmatrix} \right)^* \vdash \left(\begin{smallmatrix} 0^{n+1} & s_+ \\ p & \end{smallmatrix} \right)$$

$$Q(n): (\forall s_+ \in \{1, 2\}) \left(\begin{smallmatrix} s_+ & 0^n & 0 \\ & & q \end{smallmatrix} \right)^* \vdash \left(\begin{smallmatrix} s_+ & 0^{n+1} \\ & q \end{smallmatrix} \right)$$

Transitivity of the SMART machine in the TMH model

In order to prove the topological transitivity of SMART in the TMH model, we will prove that the configuration $\left(\begin{smallmatrix} w^2 & .2 & 2 & 2^w \\ p & & & \end{smallmatrix} \right)$ reaches any finite configuration of X_h . In order to accomplish this, we first prove that $\left(\begin{smallmatrix} w^2 & .2 & 2 & 2^w \\ p & & & \end{smallmatrix} \right)$ will progressively visit a family of patterns with a growing amount of 0s in any given position. Then we use a lemma proved in the chapter 5, that roughly says that from this family, we can attain any finite configuration.

Lemma 6.1. The configuration $\left(\begin{smallmatrix} w^2 & .2 & 2 & 2^w \\ p & & & \end{smallmatrix} \right)$ reaches each of the configurations of the family $\left\{ \left(\begin{smallmatrix} w^2 & 0^k & .0^k & 0 & 2 & 0 & 2^w \\ b & & & & & & \end{smallmatrix} \right) \right\}_{k \in \mathbb{N}} \cup \left\{ \left(\begin{smallmatrix} w^2 & 0^k & .0^{k-1} & 0 & 2 & 0 & 0 & 2^w \\ b & & & & & & & \end{smallmatrix} \right) \right\}_{k \in \mathbb{N}}$.

Proof. We will prove this by induction over k . For $k = 0$, it is enough to simulate the machine during 7 steps. Now, let us suppose that the assertion is true for $k - 1$, and let us prove that it is also true for k .

$$\begin{aligned}
& \left(\begin{array}{c} w 2 2 2 2^{k-1} . 2 2 2^{k-1} 2 2 2 2^w \\ p \end{array} \right) \\
& \text{Induction hypothesis} \\
& \left(\begin{array}{c} w 2 2 0 0^{k-1} . 0^{k-1} 0 2 0 2 2 2^w \\ b \end{array} \right) \\
& \text{Apply } B(2k) \\
& \left(\begin{array}{c} w 2 \frac{2}{b} 0 0^{k-1} . 0^{k-1} 0 2 0 2 2 2^w \\ b \end{array} \right) \\
& \text{One step} \tag{6.1} \\
& \left(\begin{array}{c} w 2 2 0 0^{k-1} . 0^{k-1} 0 2 0 2 2 2^w \\ p \end{array} \right) \\
& \text{Apply } P(2k) \\
& \left(\begin{array}{c} w 2 2 0 0^{k-1} . 0^{k-1} 0 2 0 2 2 2^w \\ p \end{array} \right) \\
& \text{One step} \\
& \left(\begin{array}{c} w 2 2 0 0^{k-1} . 0^{k-1} 0 0 0 2 2 2^w \\ q \end{array} \right)
\end{aligned}$$

From this we obtain the “even” case in one step: $\left(\begin{array}{c} w 2 2 0 0^{k-1} . 0^{k-2} 0 2 0 0 2 2 2^w \\ b \end{array} \right)$.

Continuing from the last step of the former development we obtain the “odd” case.

$$\begin{aligned}
& \text{Apply } Q(2k) \\
& \left(\begin{array}{c} w 2 2 0 0^{k-1} . 0^{k-1} 0 0 0 2 2 2^w \\ q \end{array} \right) \\
& \text{One step} \\
& \left(\begin{array}{c} w 2 0 0 0^{k-1} . 0^{k-1} 0 0 0 2 2 2^w \\ p \end{array} \right) \\
& \text{Apply } P(2k + 2) \tag{6.2} \\
& \left(\begin{array}{c} w 2 0 0 0^{k-1} . 0^{k-1} 0 0 0 2 2 2^w \\ p \end{array} \right) \\
& \text{Two steps} \\
& \left(\begin{array}{c} w 2 0 0 0^{k-1} . 0^{k-1} 0 0 2 0 2 2^w \\ b \end{array} \right)
\end{aligned}$$

□

Lemma 6.2. For every $k \leq n-1$, $\left(\begin{array}{c} . 2 0^{n+2} 0 2 \\ b \end{array} \right)^* \vdash \left(\begin{array}{c} . 2 2 0^k 0 2 0^{n-k} 2 \\ b \end{array} \right)$ and $\left(\begin{array}{c} . 2 0^{n+2} 0 2 \\ b \end{array} \right)^* \vdash \left(\begin{array}{c} . 2 0^{n-k} 2 0^k 0 2 2 \\ b \end{array} \right)$.

Proof.

$$\begin{aligned}
 & \left(.2 \ 0 \ 0 \ 0^k \ 0 \ 0^{n-k-2} \ 0 \ 0 \ 2 \right) \\
 & \qquad \qquad \qquad \text{Apply } B(n+2) \\
 & \left(.2 \ 0 \ 0 \ 0^k \ 0 \ 0^{n-k-2} \ 0 \ 0 \ 2 \right) \\
 & \qquad \qquad \qquad \text{Two steps} \\
 & \left(.2 \ 2 \ 0 \ 0^k \ 0 \ 0^{n-k-2} \ 0 \ 0 \ 2 \right) \\
 & \qquad \qquad \qquad \text{Apply } D(n+1) \\
 & \left(.2 \ 2 \ 0 \ 0^k \ 0 \ 0^{n-k-2} \ 0 \ 0 \ 2 \right) \\
 & \qquad \qquad \qquad \text{Two steps} \\
 & \left(.2 \ 2 \ 0 \ 0^k \ 0 \ 0^{n-k-2} \ 0 \ 2 \ 2 \right) \\
 & \qquad \qquad \qquad \text{Apply } B(n+2) \\
 & \left(.2 \ 2 \ 0 \ 0^k \ 0 \ 0^{n-k-2} \ 0 \ 2 \ 2 \right) \\
 & \qquad \qquad \qquad \text{One step} \\
 & \left(.2 \ 2 \ 0 \ 0^k \ 0 \ 0^{n-k-2} \ 0 \ 2 \ 2 \right) \\
 & \qquad \qquad \qquad \text{Apply } P(n+1) \\
 & \left(.2 \ 2 \ 0 \ 0^k \ 0 \ 0^{n-k-2} \ 0 \ 2 \ 2 \right) \\
 & \qquad \qquad \qquad \text{Two steps} \\
 & \left(.2 \ 2 \ 0 \ 0^k \ 0 \ 0^{n-k-2} \ 0 \ 2 \ 0 \ 2 \right) \\
 & \qquad \qquad \qquad \text{Repeat four last steps } n-k-1 \text{ times} \\
 & \left(.2 \ 2 \ 0^k \ 0 \ 2 \ 0 \ 0^{n-k-2} \ 0 \ 2 \right)
 \end{aligned} \tag{6.3}$$

Now we prove the second part.

$$\begin{aligned}
& \left(.2 \ 0 \ 0 \ 0 \ 0^{n-k-1} \ 0^{k-1} \ 0 \ 0 \ 2 \right) \\
& \qquad \text{Apply } B(n+2) \\
& \left(.2 \ 0 \ 0 \ 0 \ 0^{n-k-1} \ 0^{k-1} \ 0 \ 0 \ 2 \right) \\
& \qquad \text{Two steps} \\
& \left(.2 \ 2 \ 0 \ 0 \ 0^{n-k-1} \ 0^{k-1} \ 0 \ 0 \ 2 \right) \\
& \qquad \text{Apply } D(n+1) \\
& \left(.2 \ 2 \ 0 \ 0 \ 0^{n-k-1} \ 0^{k-1} \ 0 \ 0 \ 2 \right) \\
& \qquad \text{One step} \\
& \left(.2 \ 2 \ 0 \ 0 \ 0^{n-k-1} \ 0^{k-1} \ 0 \ 0 \ 2 \right) \\
& \qquad \text{Apply } Q(n+1) \tag{6.4} \\
& \left(.2 \ 2 \ 0 \ 0 \ 0^{n-k-1} \ 0^{k-1} \ 0 \ 0 \ 2 \right) \\
& \qquad \text{Two steps} \\
& \left(.2 \ 0 \ 2 \ 0 \ 0^{n-k-1} \ 0^{k-1} \ 0 \ 0 \ 2 \right) \\
& \qquad \text{Apply } D(n) \\
& \left(.2 \ 0 \ 2 \ 0 \ 0^{n-k-1} \ 0^{k-1} \ 0 \ 0 \ 2 \right) \\
& \qquad \text{Repeat last two steps } n-k-1 \text{ times} \\
& \left(.2 \ 0 \ 0^{n-k-1} \ 2 \ 0 \ 0^{k-1} \ 0 \ 0 \ 2 \right) \\
& \qquad \text{Two steps} \\
& \left(.2 \ 0 \ 0^{n-k-1} \ 2 \ 0 \ 0^{k-1} \ 0 \ 2 \ 2 \right)
\end{aligned}$$

□

Lemma 6.3 (5). For every finite word $v' \in \{0, 1, 2\}^*$ of length n , every state r and every $i \in \{1, \dots, n\}$, there exist $k_1, k_2 \in \mathbb{N}$ such that $\left(.2 \ 0^{k_1+k_2+n-3} \ 0 \ 2 \right) \vdash^* \left(.2^{k_1} \ v'_1 \ \dots \ v'_i \ \dots \ v'_n \ 2^{k_2} \right)$.

Theorem 6.1. *The SMART machine is topologically transitive in TMH.*

Proof. The idea of the proof is the following: Lemma 6.3 tell us that any possible finite configuration is reachable from a finite configuration x' with a certain amount of 0 in a certain position. Lemma 6.2 establishes that x' is reachable from a configuration x'' with the 0s in the center. Finally, lemma 6.1 asserts that x'' is always reachable from $\left({}^w 2 \ .2 \ 2 \ 2^w \right)$. Therefore, configuration $\left({}^w 2 \ .2 \ 2 \ 2^w \right)$ is a transitive point, and the TMH model of SMART is transitive. □

6.1.2 Machine of type b : Transitive on TMT, but not in TMH, and without blocking words

In order to have an example of this class, it is enough to take a machine of the precedent class and to multiply its movements two, *i. e.*, instead of moving one cell to the left or right, to move twice (see figure 6.3). The resulting machine will not be transitive in TMH because, if the head starts on a even cell, it will not be able to modify the content of any odd cell, thus many configurations will be unreachable.

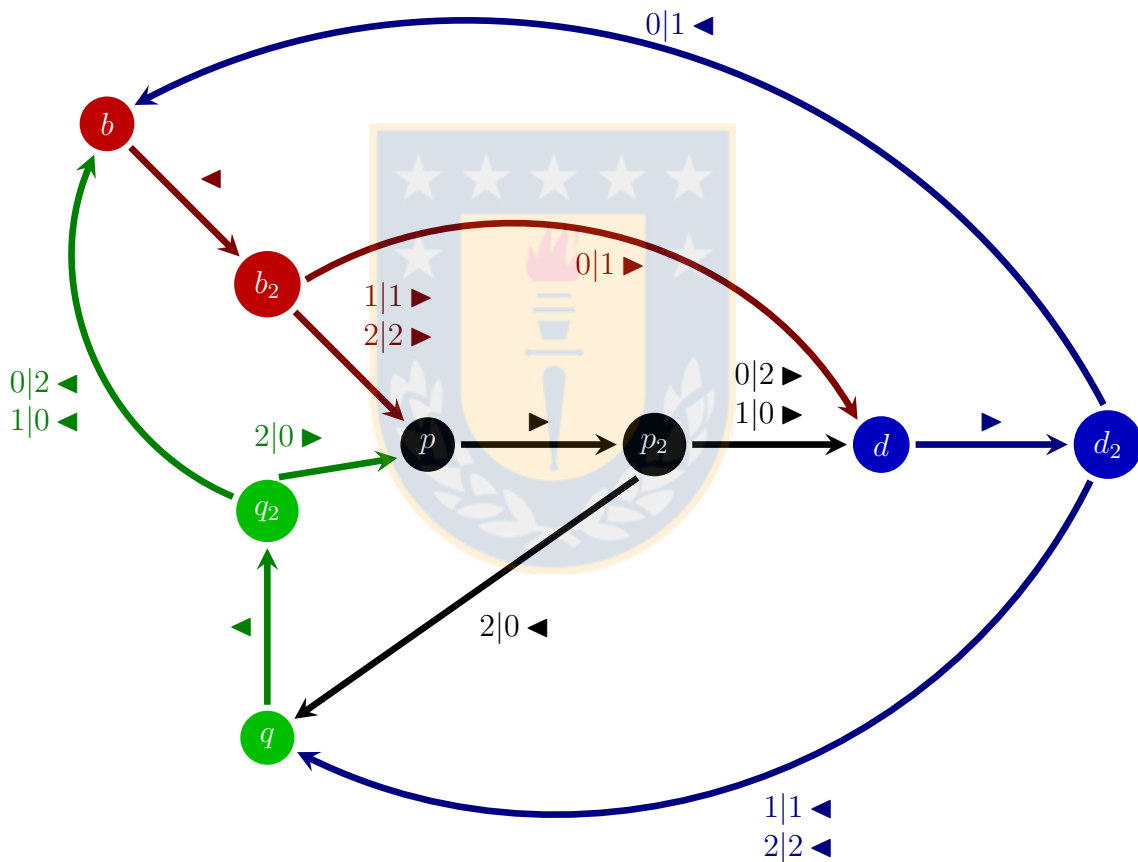


Figure 6.3: The SMART machine with twice its movement.

6.1.3 Machine of type c : Transitive on TMT, with a blocking word

The following is a machine that is transitive on TMT, but not on TMH, with the blocking word to the left: $(r_0, 1, \cdot \epsilon)$.



Figure 6.4: Shift machine

We call this machine the *shift machine*. Its trace-shift is the fullshift on $\{(r_0, 0), (r_0, 1)\}$, because the head just sequentially reads the tape, therefore its trace-shift is transitive. Now, as this machine goes forward in the tape, it can reach any finite configuration around the head, thus it is transitive on its TMT model.

6.1.4 Machine of type d : Transitive just for the trace-shift

Now we present a machine that we call *Lexicographical Ant* (LA) that has a transitive trace-shift, which is not transitive neither in TMT, nor in TMH, and that has a blocking word. Its transition function is depicted in figure 6.5. It has the particularity that it is always counting, when it starts from configuration $(\rightarrow, 0, {}^\omega 0.1^*{}^\omega)$, it will persistently comeback to the position 0 and we will see all the binary sequences appearing in increasing order. The configuration $(\rightarrow, 1, .1)$ is a blocking word to the right. It avoids transitivity in the TMT model because the finite configuration $(\overset{u}{\rightarrow} \overset{0}{\rightarrow} v)$ cannot reach the finite configuration $(\overset{u}{\rightarrow} \overset{0}{\rightarrow} w)$, if $v \neq w$, as we will prove in the following.

Given two words $v = v_0 \dots v_{n-1}$, $v' = v'_0 \dots v'_{n-1} \in \{0, 1\}^n$, let us define the lexicographical order by $v < v'$ if $\sum_{i=0}^{n-1} v_i 2^{n-i-1} < \sum_{i=0}^{n-1} v'_i 2^{n-i-1}$.

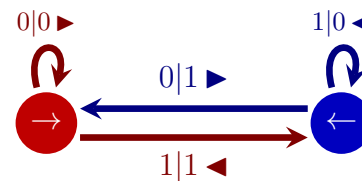


Figure 6.5: The lexicographical ant machine.

Lemma 6.4. The finite configuration $(\overset{0^n}{\rightarrow} \overset{1}{\rightarrow})$ will produce the sequence of finite configurations of the form $(\overset{v}{\rightarrow} \overset{1}{\rightarrow})$, in lexicographical order, for every $v \in \{0, 1\}^n$, without exiting the interval $[-n, 0]$.

Proof. Proof by induction on n .

Basis ($n = 1$): $(\begin{smallmatrix} 0 & 1 \\ \rightarrow & \end{smallmatrix}) \vdash (\begin{smallmatrix} 0 & 1 \\ \leftarrow & \end{smallmatrix}) \vdash (\begin{smallmatrix} 1 & 1 \\ \rightarrow & \end{smallmatrix})$

Induction hypothesis ($n = k$): $(\begin{smallmatrix} 0^k & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^* (\begin{smallmatrix} v & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^* (\begin{smallmatrix} v' & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^* (\begin{smallmatrix} 1^k & 1 \\ \rightarrow & \end{smallmatrix})$, for all $v, v' \in \{0, 1\}^k$ such that $v < v'$.

Induction thesis ($n = k + 1$): $(\begin{smallmatrix} 0^{k+1} & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^* (\begin{smallmatrix} v'' & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^* (\begin{smallmatrix} v''' & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^* (\begin{smallmatrix} 1^{k+1} & 1 \\ \rightarrow & \end{smallmatrix})$, for all $v'', v''' \in \{0, 1\}^{k+1}$ such that $v'' < v'''$.

Case 1. $v'' = 0u''$ and $v''' = 0u'''$

$$\begin{aligned} (\begin{smallmatrix} 0 & 0^k & 1 \\ \rightarrow & \end{smallmatrix}) &\vdash^{H.I.} (\begin{smallmatrix} 0 & u'' & 1 \\ \rightarrow & \end{smallmatrix}) \\ &\vdash^{H.I.} (\begin{smallmatrix} 0 & u''' & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^{H.I.} (\begin{smallmatrix} 0 & 1^k & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^* (\begin{smallmatrix} 0 & 0^k & 1 \\ \leftarrow & \end{smallmatrix}) \vdash (\begin{smallmatrix} 1 & 0^k & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^* (\begin{smallmatrix} 1 & 0^k & 1 \\ \rightarrow & \end{smallmatrix}) \\ &\vdash^{H.I.} (\begin{smallmatrix} 1 & 1^k & 1 \\ \rightarrow & \end{smallmatrix}) \end{aligned}$$

Case 2. $v'' = 0u''$ and $v''' = 1u'''$

$$\begin{aligned} (\begin{smallmatrix} 0 & 0^k & 1 \\ \rightarrow & \end{smallmatrix}) &\vdash^{H.I.} (\begin{smallmatrix} 0 & u'' & 1 \\ \rightarrow & \end{smallmatrix}) \\ &\vdash^{H.I.} (\begin{smallmatrix} 0 & 1^k & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^* (\begin{smallmatrix} 0 & 0^k & 1 \\ \leftarrow & \end{smallmatrix}) \vdash (\begin{smallmatrix} 1 & 0^k & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^* (\begin{smallmatrix} 1 & 0^k & 1 \\ \rightarrow & \end{smallmatrix}) \\ &\vdash^{H.I.} (\begin{smallmatrix} 1 & u''' & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^{H.I.} (\begin{smallmatrix} 1 & 1^k & 1 \\ \rightarrow & \end{smallmatrix}) \end{aligned}$$

Case 3. $v'' = 1u''$ and $v''' = 1u'''$

$$\begin{aligned} (\begin{smallmatrix} 0 & 0^k & 1 \\ \rightarrow & \end{smallmatrix}) &\vdash^{H.I.} (\begin{smallmatrix} 0 & 1^k & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^* (\begin{smallmatrix} 0 & 0^k & 1 \\ \leftarrow & \end{smallmatrix}) \vdash (\begin{smallmatrix} 1 & 0^k & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^* (\begin{smallmatrix} 1 & 0^k & 1 \\ \rightarrow & \end{smallmatrix}) \\ &\vdash^{H.I.} (\begin{smallmatrix} 1 & u'' & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^{H.I.} (\begin{smallmatrix} 1 & u''' & 1 \\ \rightarrow & \end{smallmatrix}) \vdash^{H.I.} (\begin{smallmatrix} 1 & 1^k & 1 \\ \rightarrow & \end{smallmatrix}) \end{aligned}$$

□

Corollary 6.1. *The finite configuration \rightarrow is a blocking word to the right.*

Proof. By lemma 6.4, we have that starting from any configuration of the form $v \xrightarrow{1}$, where $v \in \{0, 1\}^n$, the machine will arrive to $1^n \xrightarrow{1}$, without going to the right of cell 0. But there again it is on a configuration of this form, for a larger ' n ', thus it will never go to the right of cell 0. \square

Let us take the following function:

$$a(n) = \begin{cases} 1+a(\frac{n-1}{2}) & \text{if } n = \text{odd} \\ 0 & \text{if not} \end{cases}. \quad (6.5)$$

This function describes the amount of 1s at the beginning of each number represented in binary.

It is easy to see that, if we start counting from $x = (\overset{\omega 0}{\rightarrow} \xrightarrow{1})$, we have that the corresponding sequence in the trace-shift is the following.

$$\tau(x) = \prod_{i \in \mathbb{N}} \left(\begin{array}{cccc} 1 & 1^{a(i)} & 0 & 0^{a(i)} \\ \rightarrow & \leftarrow & \leftarrow & \rightarrow \end{array} \right) \quad (6.6)$$

The head of the machine will start at $\xrightarrow{1}$. As we already explained, it will switch all the 1s next to this position. The amount of 1s to convert is given by the a function.

Lemma 6.5. The trace-shift of the Lexicographical Ant is described by:

$$S_{LA} = \mathcal{S}_{\mathcal{L}(\{u\})} = \overline{\mathcal{O}(u)}, \text{ with } u = \prod_{i \in \mathbb{N}} \left(\begin{array}{cccc} 1 & 1^{a(i)} & 0 & 0^{a(i)} \\ \rightarrow & \leftarrow & \leftarrow & \rightarrow \end{array} \right) \quad (6.7)$$

Proof. :

$\overline{\mathcal{O}(u)} \subseteq S_{LA}$. It is clear from equation 6.6 and the fact that S_{LA} is closed.

$S_{LA} \subseteq \overline{\mathcal{O}(u)}$. It is enough to prove that $\mathcal{L}(S_{LA}) \subseteq \mathcal{L}(u)$. Let $w \in \mathcal{L}(S_{LA})$, and let (v, r, v') be its canonical pre-image by τ , i. e., $\tau(v, r, v') = w$ and every coordinate of (v, v') is visited during the evolution of LA on (v, r, v') . Let us suppose that $v = v_0 \dots v_m$ and $v' = v'_0 \dots v'_n$. We need to prove that $x^{(\omega 0, \rightarrow, 1)} \vdash^* (v, r, v')$. Four cases appear.

$r \Rightarrow \rightarrow$:

$v'_0 = 1$. By corollary 6.1, v' has length equal to 1: $v' = v'_0$, and by lemma 6.4 $(\overset{\omega 0}{\rightarrow} \xrightarrow{1}) \vdash^* (\overset{\omega 0}{\rightarrow} v \xrightarrow{1})$ which proves that w is a subword of u .

$v'_0 = 0$. The machine will move to the right until to find a 1.

If v' has a 1, then by corollary 6.1 it is the last symbol of v' . Now let us suppose that j is the last coordinate of v such that $v_j = 1$ (if $v = 0^{m+1}$ we are already in a pre-image of u). By lemma 6.4 $(\overset{\omega}{\leftarrow} 0 \overset{1}{\rightarrow})^* \vdash (\overset{\omega}{\leftarrow} 0 v_0 \dots v_{j-1} 0 1 \dots 1 \overset{1}{\rightarrow})^* \vdash (\overset{\omega}{\leftarrow} 0 v_0 \dots v_{j-1} \overset{0}{\leftarrow} 0 \dots 0 1)^* \vdash (\overset{\omega}{\leftarrow} 0 v_0 \dots v_{j-1} 1 0 \dots 0 1)^*$ which contains (v, r, v') , proving that w is a subword of u .

If v' is identically equal to 0, then the head exit (v, r, v') at iteration n , $v = \epsilon$, and $w = (\rightarrow, 0)^n$ which is clearly a subword of u .

$r = \leftarrow$:

$v'_0 = 0$. In this case, $(\overset{v}{\leftarrow} 0 v'_1 \dots v'_n)^* \vdash (\overset{v}{\rightarrow} 1 v'_1 \dots v'_n)^*$. This configuration fits in one of the former cases, thus it is attained by x . Since the machine is reversible, its predecessor is also attained by x .

$v'_0 = 1$. Let j be the last coordinate of v such that $v_j = 0$ (if $v = 1^{m+1}$, $v' = \epsilon$ and $w = (\leftarrow, 1)^m$, which is clearly a subword of u), then $(v_0 \dots v_{j-1} 0 1 \dots 1 \overset{1}{\leftarrow} v'_1 \dots v'_n)^* \vdash (v_0 \dots v_{j-1} \overset{0}{\leftarrow} 0 \dots 0 0 v'_1 \dots v'_n)^*$. Which is proved to be in the orbit of x in the last case, and again this implies, by reversibility, that the original configuration is reached by x .

□

We can note that, in fact, the Lexicographical Ant is not transitive in TMT. This is because from configuration $(\overset{1}{\rightarrow} 1) T_t$ can never get to $(\overset{1}{\rightarrow} 0)$, due to Corollary 6.1. Although, Lexicographical Ant has a transitive trace-shift, because this is described as the closure of the orbit of a unique infinite word.

Theorem 6.2. *The trace-shift of the Lexicographical Ant is transitive.*

6.2 Undecidability

The following definitions are already described in Chapter 5, but it will be discussed again in order to preserve the independence of each Chapter.

Reversing the time

This technique is used in [28]. It takes a deterministic and reversible Turing machine $M = (Q, \Sigma, \delta)$, and creates two new reversible machines $M+ = (Q \times \{+\}, \Sigma, \delta^+)$, and $M- =$

$(Q \times \{-\}, \Sigma, \delta^-)$, where $(r, +)$ and $(r, -)$ states represent M in state r running forwards or backwards in time, respectively.

Embedding

This technique serves to put a machine inside another in such a way that the new machine has a property(ies) that depends on some properties of the original machines. We distinguish the *host* machine $H = (Q, \Sigma, \delta)$ and the invited machine I .

The invited machine needs to be deterministic and reversible and we will assume that it has the gentle property of *innocuousness*.

Definition 6.1. *A machine is innocuous if:*

- every defective pair (r_d, β) is associated to a unique error pair (r_e, β) , in such a way that
- for every defective configuration (r_d, i, w) , its evolution is either infinite or it leads to a halting configuration of the form (r_e, i, w) .

In other words, if it halts, it does it at the initial position and with the initial tape contents.

Innocuous machines can be obtained by gluing together the error states of $M+$ with the defective states of $M-$. The defective states of M corresponds to the defective states of $M+$ and so the error states of $M-$. In the next section we will give two different constructions of innocuous machines, that will serve to different purposes.

The alphabet of the invited machine will be coded in such a way that H and I work with the same alphabet. This can be done, by modifying the transition rule of I , without losing its determinism, reversibility and innocuousness, but maybe augmenting the number of defective and error states.

The invited machine I will be embedded in the host machine in the following way:

- Every pair defective-error state (r, r') of machine I will be connected to a unique state p of machine H .
- In order to do that, state p is replaced by two states p' and p'' , called *splitted states*.
- Every instruction $\delta(q, \alpha) = (p, \beta)$ in H is now replaced by $\delta(q, \alpha) = (p', \beta)$ and every instruction $\delta(p, \alpha') = (q', \beta')$ is replaced by $\delta(p'', \alpha') = (q', \beta')$.

- Now, for every defective pair (β, r) , we add the instructions $\delta(p', \beta) = (r, \beta)$ and $\delta(r', \beta) = (p'', \beta)$.
- Finally, we add the instructions $\delta(p', \alpha) = (p'', \alpha)$, with α a no-defective symbol for r .

If necessary for the amount of pair defective-error states, we can iteratively replace one splitted state into two more.

In the resulting machine we will observe that, if we start at a state of H , we will see the evolution of H , interrupted by the action of I when the state p is attained. We can always suppose that we start at H , because evolving I backward will carry us to a defective state, and finally to H , except if we are over a backward infinite orbit of I .

The embedding will be particularly useful when H is minimal and I is mortal, because in this case, only one initial configuration of H will allow to test the behavior of I over all its defective configurations.

6.2.1 Undecidability results

(c-RtransTMH) Given a reversible and complete Turing machine M , decide whether its TMH system is transitive or not.

(c-RtransTMT) Given a reversible and complete Turing machine M , decide whether its TMT system is transitive or not.

(c-RtransTS) Given a reversible and complete Turing machine M , decide whether its trace-shift is transitive or not.

Theorem 6.3. **(c-RtransTMH)**, **(c-RtransTMT)** and **(c-RtransTS)** are undecidable.

Proof. We will do it by reduction from the **Reachability** problem for reversible and aperiodic machines, proved undecidable in [28].

(Reachability) Given a reversible and aperiodic Turing machine M and two states r_1 and r_2 , decide whether there exists a configuration $(r_1, 0, w)$ that reaches the state r_2 at some iteration.

Let us take a reversible and aperiodic machine $M = (Q, \Sigma, \delta)$ and two states $r_1, r_2 \in Q$. We start by eliminating every exiting transition from r_2 and every entering transition to r_1 . Let $\$$ be a symbol which is not in Σ , and let us add it to it. We obtain in this way a new machine that is not complete, because non instruction is defined for this new symbol. Moreover, all of its states are both error and defective states.

We will embed this machine into SMART; in order to do this, we recode its alphabet in base three, by adding new additional symbols $\$_1, \dots, \$_k$, if necessary, called error symbols. Let us call $M' = (Q', \Sigma', \delta')$ the so obtained machine.

Now, we create two copies of the machine M'_+ and other two of M'_- , by using the technique of “reversing the time”, and we connect them as shown in figure 6.6 to create an “invited” machine I to be embedded into SMART. The state $(r_2, +)$ of M'_+ is identified to the state $(r_2, -)$ of M'_- , the same happens with $(r_1, +)$ and $(r_1, -)$; in this way, these states are not defective nor error states of I . The arrows labeled by $q_1, \dots, q_{|Q|-2}$ are connected to each state of M'_+ , with exception of r_1 and r_2 . The arrows labeled by $p_1, \dots, p_{|Q|-2}$ are connected to each state of M'_- , with exception of r_1 and r_2 . The analogous happens with the arrows labeled by $q'_1, \dots, q'_{|Q|-2}$ and $p'_1, \dots, p'_{|Q|-2}$ with respect to machines M'_- and M'_+ , respectively.

The machine I defined in this way is innocuous, in fact, when starting at q_i (or p_i) we distinguish three cases.

1. The machine stays in M'_+ forever.
2. The machine attains r_2 , and enters M'_- . From there, the computation is “reversed” and it exits by $(q_i, -)$ to M'_+ , where the history is repeated, and the machine exits by q'_i , by leaving as it was at the beginning.
3. The machine attains an error state of M'_+ . In this case it enters M'_- , the computation is reversed and it exits by q'_i as we wanted.

If we embed I using *SMART* as the host machine, we obtain a machine $SMART^I$.

Now, we will prove that, if M machine can not reach r_2 from r_1 , then the new machine is transitive in TMH.

A machine is transitive in TMH if every two finite configurations on TMH can reach each other. We already know that every two configurations with states on SMART can reach each

other, because SMART is transitive on TMH and to enter through splitting states to I is innocuous.

Also, any finite configuration with an state on I can always reach any configuration with an state on SMART. This is because, as stated before, any finite configuration inside I can exit to SMART if it can not be reached r_2 from r_1 through M ; and on SMART we can reach any finite configuration.

We just need to prove that any configuration with a state on SMART can reach any finite configuration on M' . As SMART is transitive by itself, then we just need that finite configurations on M' can be reached from any splitted state.

Suppose without loss of generality that a finite configuration x on M'_1+ is unreachable from SMART. Next, we put error symbols on the context of the configuration. Here we have two possible paths:

M'_1+ reaches an error pair and it go to machine M'_2- . Here the computation is reversed to reach the same initial configuration, but with the state in M'_2- . Now, as machine M has no periodic points, the computation will evolve until reach an error pair or state r_2 . In any of the previous cases, the computation will reach a state of SMART, in the first case directly, and in the second one, computation in M'_2+ will reach an error pair because M'_1+ reached it. This path is forbidden, because if we exit to SMART, then, as machine I is innocuous, we should enter in some point and the configuration x would be reachable.

M'_1+ reaches state r_2 and it go to machine M'_1- . Here the computation is reversed to reach the same initial configuration, but with the state in M'_1- . Now, as machine M has no periodic points, the computation will evolve until reach an error pair. Here, we are in machine M'_2+ , when the computation is repeated to exit to SMART through M'_2- . This path is also forbidden for the same reasons that before.

As there not exist more possible paths, because M has no periodic points and r_1 is unreachable from r_1 , we have a contradiction.

Therefore, the new machine is transitive in TMH if the original M machine can not reach r_2 from r_1 . Also, the transitivity in TMT and t -shift are inherit from TMH, so if M can not reach r_2 from r_1 , then the new machine is transitive in both TMT and t -shift.

Now, if M machine can reach r_2 from r_1 , then machine M' has a periodic configuration

for TMH. As we remarked in section 6.1, this implies that $SMART^I$ cannot be transitive for its trace shift, neither for its TMT system nor for its TMH system [15].

Finally, we conclude that all these three problems are undecidable.

□

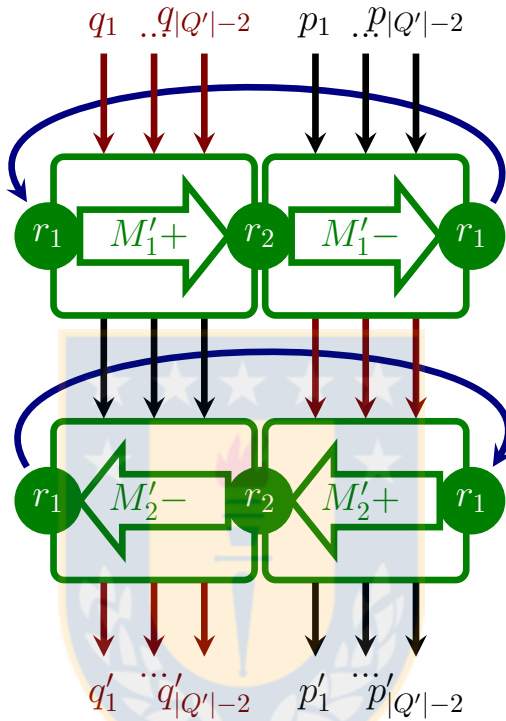


Figure 6.6: Invited machine for an embedding that is transitive if and only if r_1 cannot reach r_2 in the evolution of M , used in the proof of theorem 6.2.1.

(c-RminTMT) Given a reversible and complete Turing machine M , decide whether its TMT system is minimal or not.

(c-RminTS) Given a reversible and complete Turing machine M , decide whether its trace-shift is minimal or not.

Theorem 6.4. **(c-RminTMT)** and **(c-RminTS)** are undecidable.

Proof. We will do it by reduction from the **Mortality** problem for reversible and aperiodic machines, proved undecidable in [28]. Mortal machines are machines that, independently in the starting configuration, they always attain a defective pair that halts them.

(Mortality) Given a reversible and aperiodic Turing machine M , decide whether it is mortal or not.

Let us take a reversible and aperiodic Turing machine $M = (Q, \Sigma, \delta)$ and let us code its alphabet Σ in base three to embed it into SMART. We use the technique of reversing the time to define the machines $M+$ and $M-$ and produce the *invited* machine I described in figure 6.7.

This machine is clearly innocuous, and we embed it into $SMART$ to produce $SMART^I$.

First, if M is not mortal, there is a trace that starts in a state of $M+$ and never exits from this machine. Such a behavior is impossible in a machine with a minimal TMT or t -shift system, where all the trajectories need to be transitive, and so they must visit all the states of the machine.

Conversely, if M is mortal, so it its reverse and we will prove that any arbitrary TMT configuration x of $SMART^I$ is a transitive point. Every finite configuration with a state in $SMART$ can be reached from x , because $SMART$ is minimal, and the invited machine is innocuous and it has not an infinite computation. Now, every finite configuration with a state in the invited machine can be reached from a finite configuration y with a defective state, because M is mortal, and we already know that configuration y can be reached from x , because $SMART$ is minimal and y is reached directly from a splitted state. Then, $SMART^I$ is minimal in its TMT dynamical model. Also, the minimality is inherited from TMT to t -shift, therefore $SMART^I$ is minimal in its t -shift.

Finally, machine $SMART^I$ is minimal in its trace-shift and its TMT model if and only if M is mortal.

□

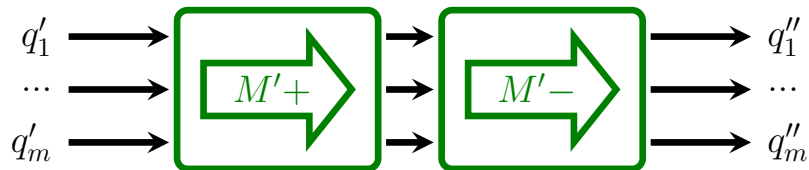


Figure 6.7: Invited machine for an embedding that is minimal if and only if M is mortal, used in the proof of theorem 6.4.

6.3 Complexity of Transitivity Problem and Minimality Problem

In this section, we will discuss how complex is the transitivity problem and minimality problem.

6.3.1 Transitivity Problem and Minimality Problem are Π_2^0

As we proved in the previous section, transitivity problem is Π_1^0 -hard and minimality problem is Σ_1^0 -hard, as the emptiness problem is reducible to the first, and mortality problem is reducible to the second. Both problems naturally arise from *Analytical Hierarchy*, as it can be defined by uncountable quantifiers (second order). We will prove that both problems can be classified in arithmetical hierarchy, more specifically in Π_2^0 .

Lemma 6.6. Transitivity Problem and Minimality Problem belong to Π_2^0 .

Proof. To prove this, we need to write the formulas that define them in first order arithmetic. The Transitivity Problem can be written as:

$$\begin{aligned} \text{(TMT)} \text{ A machine } M \text{ is transitive in TMT dynamical model } &\iff \\ &\forall x, y \in \Sigma^* \times Q \times \Sigma^*, \exists t \in \mathbb{N} : R(M, x, y, t) \end{aligned}$$

With $R(M, x, y, t)$ is the recursive function that computes machine M starting in finite configuration x , making t steps of computation (proving all the possible finite context if needed), and test if finite configuration y is part of the finite configuration reached.

It is easy to see that this definition is equivalent to the definition in section 3.5, because the uncountable part is replaced by finite context, as the rest of the tape can not be reached in time t . For TMH, we just need to change the countable quantifiers for finite configurations. The definition of transitivity in t -shift is already in first order arithmetic with a universal quantifier, followed by an existential quantifier.

Now, the Minimality problem can be written as:

(TMT) A machine M is minimal in TMT dynamical model $\iff \forall n \in \mathbb{N}, \exists t \in \mathbb{N} : Q(M, n, t)$

With $Q(M, n, t)$ the recursive function that computes finite orbits of all possible finite configurations of size t (computes until it reaches the bound of the configuration, or it loops), and test if all configurations of size n are present in all finite orbits.

It is easy to see that if the previous statement is true, then the machine is topologically minimal in TMT. Now for the other implications, let prove it by contradiction: Imagine that the previous statement is false:

Then, exists a size n , such that for all size t , the orbit of a configuration of size t does not have all finite configurations of size n . Therefore, for compactity, there exists an infinite configuration that does not includes all finite configurations of certain size, therefore, the machine is not minimal.

For the t -shift, we just need to change finite configurations for finite words, and everything will work of the same way. \square

6.4 Coded Systems associated to Turing machines

We have worked with several Turing machines with transitive no sofic t -shifts, but, can we have examples with every type of coded systems? Recall that we have four types of coded systems that have the following relation: sofic shifts \subset shifts with countable many follower sets of left-infinite words \subset synchronized systems \subset coded system. It is easy to see that a machine that erase symbols to the right has an irreducible sofic t -shift. On the other hand, the SMART machine (Figure 6.2) is not even a coded system, because its t -shift has no periodic points.

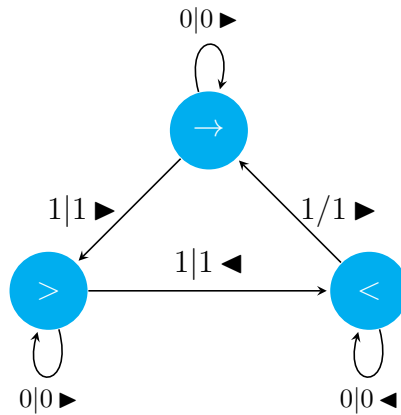


Figure 6.8: Machine J : Synchronized machine with countable follower sets in its t -shift.

Note: We will call to a Turing machine with a sofic t -shift as sofic machine. This will also be true with machines with synchronized t -shifts, therefore these machines will be called synchronized machines.

Now, we take into account the machine in Figure 6.8:

This machine is just the quintuple representation of an Embedding (see Section 5.3.1) inside the shift machine (see figure 6.4) of a machine that reads symbol 0 and goes to the right until it reads symbol 1, together with its reverse. This machine is, by definition, transitive in its t -shift, and then its representative labeled graph is irreducible. It has the synchronizing words $\frac{1}{\rightarrow}$, therefore its t -shift is a synchronized system.

Now, we have that $\frac{1}{\rightarrow} 0^n \frac{1}{\rightarrow}$ has as many follower sets as n . Therefore, the t -shift of machine J has infinite countable many follower set. Then, we have a non sofic synchronizing t -shift with countable many follower set.

Now, if we made a little modification, we can get a synchronized machine with uncountable follower sets, as can be seen in Figure 6.9.

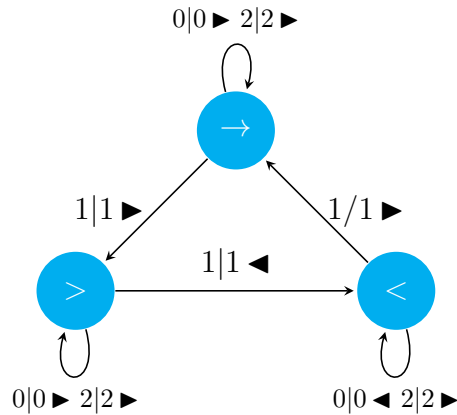


Figure 6.9: Machine J' : Synchronized machine with uncountable follower sets in its t -shift.

This machine has the same behavior that the previous one, taking symbols 2 and 0 as the same. Then, this machine is synchronized, with synchronized word $\frac{1}{\rightarrow}$. But, we have that $\frac{1}{\rightarrow} 0|2^n \frac{1}{\rightarrow}$ has as many follower set as $\{0, 2\}^n$. Therefore, the t -shift of machine J' has infinite uncountable many follower set. Then, we have a no sofic synchronizing t -shift with uncountable many follower set.

Now, we just need a machine without a synchronized word, but coded. Is it possible?

Conjecture 6.1. *If S is the t -shift of a Turing machine, then if S is a coded system, then it is a synchronized system.*

Chapter 7

Conclusions



This thesis was based on the study of dynamical properties of three dynamical systems related to Turing machines. Turing machines are not naturally translatable to topological dynamical systems, because the position of the head is measured in the set \mathbb{Z} , implying that the phase space is not compact. That is why in this thesis we consider three dynamical systems, Turing model with Moving Head (TMH), Turing model with Moving Tape (TMT) and the column factor of TMT model, the t -shift. While other dynamical models have been associated with Turing machines (including generalized shifts and cellular automata), they do not faithfully represent the mechanics of the machine, not entirely reflecting the original dynamics.

The questions addressed in this study were based on interesting dynamical properties within the dynamical and topological field that had not yet been fully addressed in the context of Turing machines. The considered properties are the surjectivity of the t -shift, the entropy of the TMT model, periodicity in reversible and complete machines, transitivity of all models and minimality of TMT and t -shift. We focus on proving the existence of machines with each of these properties and the decidability of them.

In general terms, consider quadruple or quintuple model of a Turing machine does not affect the existence or decidability of topological properties, but the surjectivity in t -shift and its dichotomy with the surjectivity in Turing machines depends of the selected model, as there does not exist a non-surjective quadruple model Turing machine with a surjective t -shift.

To address the problem of surjectivity in the t -shift of quintuple model Turing machine, the notion of *blocking word* was considered. This concept was born in cellular automata, as a sequence of states that do not allow information to pass through. In the universe of Turing machine, the information is transferred through the head, so that our translation considered a finite configuration that prevents the head from passing through a part of the tape. We prove that there exists non surjective quintuple model Turing machines with a surjective t -shift. But this is possible only when a blocking word exists. Performing a reduction from the problem of blocking words, it was possible to show that the problem of surjectivity in t -shift is undecidable.

Blocking words resulted to be fundamental, not only in the surjective context, but also in the transitive context. Non surjective machine can be transitive in its t -shift model. Also, the presence of blocking words prevents the machine to have a transitive TMH model, and the absence of blocking words implies the equivalence between transitivity in TMT and t -shift.

The topological entropy of Turing machines has been widely discussed in literature. Blondel et al. demonstrated that entropy is uncomputable for Turing machines with two or more tapes and Jeandel proved that it is computable for one-tape Turing machines, in the sense that it is

possible to approach the entropy value as well as one wishes. However, as suggested by the study and properties exposed by E. Jeandel himself, it is not possible to decide if a Turing machine has positive entropy. This demonstration was carried out by reduction from the halting problem for counter machines.

Most of our work concerns a particular Turing machine, called *SMART*, created by J. Cassaigne. Until the presentation of such a machine, there was no evidence of a complete and reversible machine with the property of aperiodicity, topological transitivity in TMH and topological minimality in TMT. Furthermore, SMART resulted to have a substitutive subshift and be time-symmetric. The latter concept is introduced from cellular automata, and allowed us to more easily demonstrate many lemmas about the machine.

In the study of transitivity, we have presented several Turing machines that have transitive dynamical systems. It was possible to hierarchize Turing machines in nested classes depending on which associated dynamical system had the topological transitivity property, including examples in each category that prove that the hierarchy is strict.

To prove the undecidability of aperiodicity in complete and reversible Turing machines, transitivity on all dynamical models, and minimality in the t -shift and the TMT model, we have used the same reduction technique, that we call *embedding*. This technique allows us to connect an Invited machine I with a Host machine H that has the property we want to study. Machine I must have the property of *innocuity* in order to keep machine H dynamics. Thus, the property to study depends on another property, which is known to be undecidable, that machine I should hold. In addition, this technique allows us to create families of Turing machines with properties such as minimality and transitivity. As an example, embedding mortal machines into SMART, allows us to have infinitely many Turing machines with a topologically minimal TMT model. Also, as the prove of Aperiodicity in complete and reversible Turing machine, and the prove of Transitivity in all dynamical models are the same (except for the error symbol in the second prove, which did not affect Periodicity), we can conclude that the set of Turing machine with periodic points and the set of Turing machine with transitive t -shift are *recursively inseparable sets*.

TMH is a cellular automaton based on a subshift. The study of cellular automata based on subshifts started with the *Curtis-Hedlund-Lyndon theorem*, which states the locality of maps from subshifts to subshifts that are continuous and shift-commuting [3]. The undecidability results in TMH, as a shift space cellular automaton, are applicable in that context. Thus, the transitivity problem is undecidable for shift space cellular automata.

In addition to the hierarchization by dynamical model, there exists a hierarchization within

the class of coded shifts. We also exhibit examples that proves that this hierarchization is proper, except that we could not find an example of a Turing machine with a coded t -shift without synchronizing words. Does there exists a Turing machine with a coded systems without a synchronized word?

It was possible to include the problems of transitivity and minimality in the arithmetical hierarchy although both problems come from the analytical hierarchy. We were not able to prove their completeness, nevertheless we proved that they are Π_1^0 -hard and Σ_1^0 -hard, respectively. Are these problems complete in any hierarchical set?

In addition with the conjecture given in chapter 6, also exists another classification on subshifts, called *almost sofic* shifts [32]. Those are shifts that can be entropically approximated through sofic shifts. In the general subshift universe, there are subshifts that are not almost sofic, those are shift with positive entropy but without periodic points. But all the examples known to be aperiodic on Turing machines have zero entropy. Are all t -shift almost sofic?

The Rice's theorem [45] states the undecidability of any non-trivial problem about the language that a given Turing machine recognizes. In fact, our problems are defined over infinite words in the tape, being, intrinsically, *harder* to solve. In this fashion, we could think that any non-trivial problem defined with infinite inputs would have the same fate, but there are no simple way to translate the finite input version to the infinite input version. An example of this could be seen in [22], where the undecidability of the infinite input version of totality problem, the immortal problem, is proved in a very clever way, passing through recursive-like calls in Turing machines. These two problems are complete in different levels of the arithmetical hierarchy.

On the other hand, some problems are decidable, even when defined for infinite inputs, as the reversibility problem. Also, it is not easy to determine the triviality of a problem, because the difficulty to prove the existence of a Turing machine with a certain property. As an example, the existence of an aperiodic Turing machine was questioned by Kůrka, and then years later proved by Blondel, Cassagne and Nichitiu. Also, the minimality for TMH is trivial, as there not exists a minimal TMH machine.

Nevertheless, the embedding technique has proved to be highly powerful and versatile. In this way, it may allow to prove a Rice-like theorem for infinite inputs. If there exists some common property between long term dynamical concepts, it would be possible to prove that this property depends of another undecidable property in an embedding. Does a Rice-like theorem exists in this context?

Even when computers, represented here by Turing machines, are used to study dynamical

Topic	Worked by and Results
Blocking Words	<i>This Work</i> (Characterization, Existence and Undecidability)
Entropy	Jeandel (Computability in One Head TM) [26] Blondel and Delvene (Undecidability in Two or more Heads TM) [4] <i>This Work</i> (Undecidability in One Head TM)
Equicontinuity	Gajardo and Guillon (Characterization and Existence) [14] <i>Open</i> (decidability)
Immortality	Hooper (Characterization, Existence and Undecidability) [22] Jeandel (Immortal Configurations) [25] Kari and Ollinger (Undecidability in Reversible TM) [28]
Periodic Orbit	Blondel, Cassaigne and Nichitiu (Existence) [5] Kari and Ollinger (Undecidability in RTM and c-TM) [28] <i>This Work</i> (Undecidability in c-RTM)
Relation between Topological Properties	<i>This work</i> (Periodicity with Transitivity) <i>Open</i> (Relation between Transitivity, Minimality and Mixing)
Rice-Like Theorem	<i>Open</i>
Sofic on t -shift	Gajardo (Characterization and Existence) [13] <i>Open</i> (Decidability, Almost Sofic)
Subshift based in TMH	<i>Open</i>
Surjectivity on t -shift	<i>This Work</i> (Characterization, Existence and Undecidability)
Topological Transitivity	<i>This Work</i> (Characterization, Existence and Undecidability)
Topological Minimality	<i>This Work</i> (Existence and Undecidability) <i>Open</i> (Characterization)
Topological Mixing	<i>Open</i>

Table 7.1: Research on Turing machine dynamical systems.

systems, much of our results are not translatable directly to the simulated dynamical systems. The topological properties use the information of all possible contents of the tape, and Universal Turing machines, when simulating other systems, use just a subset of configurations sets.

It was possible to study dynamical properties associated to one-tape Turing machines and prove its undecidability. We hope that the inclusion of this new understanding about periodicity, entropy, transitivity and minimality in Turing machines encourages other researchers to fill the gaps inside one-tape Turing machines dynamics.

Finally, let us give a summary chart of the present state of Turing Machine dynamic.

Bibliography

- [1] Berlekamp, E., J. Conway, and R. Guy: *Winning ways for your mathematical plays*. Academic Press, 2:334–343, 2012.
- [2] Blanchard, F. and G. Hansel: *Coded systems*. *Theor. Comput. Sci.*, 44(1):17–49, August 1986, ISSN 0304-3975.
- [3] Blanchard, F., A. Maass, and A. Nogueira: *Topics in Symbolic Dynamics and Applications*. Cambridge University Press, 2000, ISBN 9780521796606.
- [4] Blondel, V. and J. Delvenne: *Quasi-periodic configurations and undecidable dynamics for tilings, infinite words and Turing machines*. *Theoret. Comput. Sci.*, 319:127–143, 2004.
- [5] Blondel, V. D., J. Cassaigne, and C. Nitchiu: *On the presence of periodic configurations in Turing machines and in counter machines*. *Theoret. Comput. Sci.*, 289:573–590, 2002.
- [6] Börger, E.: *Computability, complexity, logic*. Elsevier Science Publishers, Holland, 1989, ISBN 978-0-444-87406-1.
- [7] Brudno, A.: *Entropy and the complexity of the trajectories of a dynamical system*. *Transactions of the Moscow Mathematical Society*, 44(2):127–151, 1983.
- [8] Church, A.: *An unsolvable problem of elementary number theory*. *American Journal of Math*, 58:345–363, 1936.
- [9] Davis, M.: *The undecidable, basic papers on undecidable propositions, unsolvable problems and computable functions*. Raven Press, New York, 1965, ISBN 0-911216-01-4.
- [10] Devaney, R.: *An introduction to chaotic dynamical systems*. Addison-Wesley, 1989, ISBN 978-081-334-085-2.
- [11] Downey, R. and D Hirschfeldt: *Algorithmic randomness and complexity*. Springer, 2010.
- [12] Fiebig, D. and U. Fiebig: *Covers for coded systems, in symbolic dynamics and its applications*. *Contemporary Mathematics*, 135:139–180, 1992.
- [13] Gajardo, A.: *Sofic one head machines*. In Durand, B. (editor): *Journées Automates Cellulaires*, pages 54–64, 2008.
- [14] Gajardo, A. and P Guillon: *Zigzags in Turing machines*. In Ablayev, Farid M. and Ernst W. Mayr (editors): *CSR*, volume 6072 of *Lecture Notes in Computer Science*, pages 109–119. Springer, 2010, ISBN 978-3-642-13181-3.

- [15] Gajardo, A. and J. Mazoyer: *One head machines from a symbolic approach*. Theor. Comput. Sci., 370:34–47, 2007.
- [16] Gajardo, A., Kari J. and A. Moreira: *On time-symmetry in cellular automata*. Journal of Computer and System Sciences, (78):1115–1126, 2012.
- [17] Garby, M. and S. Johnson: *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, 1979, ISBN 0-716-71045-5.
- [18] Gödel, K. and B. Meltzer: *On formally undecidable propositions of principia mathematica and related systems*. Dover Publications, New York, 1992, ISBN 0-486-66980-7.
- [19] Hájek, P.: *Arithmetical hierarchy and complexity of computation*. Theoretical Computer Science, 8:227–237, 1979.
- [20] Hennie, F.: *One-tape, off-line Turing machine computations*. Information and Control, 8:553–578, 1965.
- [21] Hilbert, D. and W. Ackermann: *Grundzüge der theoretischen Logik (Principles of Mathematical Logic)*. Springer-Verlag, 1928, ISBN 0-8218-2024-9.
- [22] Hooper, Philip K.: *The undecidability of the Turing machine immortality problem*. Journal of Symbolic Logic, 31(2):219–234, 1966.
- [23] Hopcroft, J. and J. Ullman: *Introduction to automata theory, languages, and computation*. Addison-Wesley, 1979, ISBN 0-321-45536-3.
- [24] Ilachinski, A.: *Cellular automata: A discrete universe*. World Scientific, 2001, ISBN 978-981-238-183-5.
- [25] Jeandel, E.: *On immortal configurations in Turing machines*. Lecture Notes in Computer Science, 7318:Ch. 25, 1982.
- [26] Jeandel, E.: *Computability of the entropy of one-tape Turing machines*. STACS, 25:421–432, 2014.
- [27] Kari, J.: *Reversibility and surjectivity problems of cellular automata*. Journal of Computer and System Sciences, 48(1):149–182, 1994.
- [28] Kari, J. and N. Ollinger: *Periodicity and immortality in reversible computing*. In Ochmanski, Edward and Jerzy Tyszkiewicz (editors): *MFCSS*, volume 5162 of *Lecture Notes in Computer Science*, pages 419–430. Springer, 2008, ISBN 978-3-540-85237-7.
- [29] Kůrka, P.: *Language complexity of unimodal systems*. Complex Systems, 10:283–300, 1996.
- [30] Kůrka, P.: *On topological dynamics of Turing machines*. Theoret. Comput. Sci., 174(1-2):203–216, 1997.
- [31] Kůrka, P.: *Topological and Symbolic Dynamics*. Société Mathématique de France, Paris, France, 2003.

- [32] Lind, D. and B. Marcus: *An introduction to symbolic dynamics and coding*. Cambridge University Press, New York, NY, USA, 1995, ISBN 0-521-55900-6.
- [33] Lukkarila, V.: *Sensitivity and topological mixing are undecidable for reversible one-dimensional cellular automata*. TUCS, (927), 2009.
- [34] Martin, G.: *Mathematical games: The fantastic combinations of john conway's new solitaire game "life"*. Scientific American, 223:120–123, 1970.
- [35] Melanie, M.: *Is the universe a universal computer?* Science, 298:65–68, 2002.
- [36] Moore, C.: *Unpredictability and undecidability in dynamical systems*. Phys. Rev. Lett., (64), 1990.
- [37] Morita, K.: *A simple construction method of a reversible finite automaton out of Fredkin gates, and its related problem*. The Trans. of the IEICE, E73(6):978–984, 1990.
- [38] Morita, K.: *Universality of a reversible two-counter machine*. Theor. Comput. Sci., 168(2):303–320, 1996.
- [39] Morita, K., A. Shirasaki, and Y. Gono: *A 1-tape 2-symbol reversible Turing machine*. IEICE TRANSACTIONS, E72-E(3):223–228, 1989.
- [40] Morita, K. and Y. Yamaguchi: *An universal reversible Turing machine*. LNCS, 4664:90–98, 2007.
- [41] Neumann, J. von: *Theory of self-reproducing automata*. Univ. Illinois Press, 1970.
- [42] Nielsen, M. and I. Chuang: *Quantum computation and quantum information*. Cambridge University Press, Cambridge, UK, 2000, ISBN 0-521-63503-9.
- [43] Oprocha, P.: *On entropy and Turing machine with moving tape dynamical model*. Nonlinearity, 19:2475–2487, October 2006.
- [44] Papadimitriou, M.: *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994, ISBN 0201530821.
- [45] Rice, H.: *Classes of recursively enumerable sets and their decision problems*. Trans. Amer. Math. Soc., 74:358–366, 1953.
- [46] Shoenfield, J.: *Mathematical logic*. Association for Symbolic Logic, 2001.
- [47] Torres, R., N. Ollinger, and A. Gajardo: *Undecidability of the surjectivity of the subshift associated to a Turing machine*. LNCS, (7581):44–56, 2013.
- [48] Torres, R., N. Ollinger, and A. Gajardo: *The transitivity problem of Turing machine*. LNCS, (9234):231–242, 2015.
- [49] Turing, A.: *On computable numbers, with an application to the entscheidungsproblem*. Proceedings of the London Mathematical Society, 2(42):230–265, 1936.

- [50] Ulam, S.: *Random processes and transformations*. Proc. Int. Congress of Math, 2:264–275, 1952.
- [51] Weiss, B.: *Subshifts of finite type and sofic systems*. Monats. Math., 77:462–474, 1973.
- [52] Willard, S.: *General topology*. Addison-Wesley, Reading, Massachusetts, 1968, ISBN 0486434796.
- [53] Wolfram, S.: *Statistical mechanics of cellular automata*. Reviews of Modern Physics, 55:601–644, 1983.

