MÁQUINAS DE TURING PARACONSISTENTES:
algunas posibles definiciones y consecuencias

Juan Carlos Agudelo Agudelo

Monografía presentada como requisito parcial para optar
al título de especialista en Lógica y Filosofía

Asesor
Andrés Sicard Ramírez
Profesor Departamento de Ciencias Básicas
Universidad EAFIT

Departamento de Humanidades
Escuela de Ciencias y Humanidades
Universidad EAFIT
Medellín, Colombia, S.A.
2003

Nota de aceptación

____

____

____

____

____

____

Presidente del Jurado

____

Jurado

____

Jurado

Medellín, noviembre de 2003.

A mi mamá y a la memoria de mi papá,
por permitirme elegir.

Prefacio

Soñar es quizás una de las características más distintivas del ser humano. En nuestra imaginación todo es posible. Construimos teorías, situaciones y objetos ideales y empezamos a creer que pueden ser reales, lo que nos lleva, en algunas situaciones, a emprender una aventura de realización de lo ideal. Los resultados son diversos.

Pienso que una monografía, desde que sea tomada de forma seria, es un proceso de soñar y de tratar de realizar lo soñado. Sueño que en ciertas ocaciones parece convertirse en pesadilla. Cuando logramos aclarar de forma total o parcial los objetivos propuestos, luego de intensas jornadas de trabajo, donde suelen haber momentos de alegría y también de frustración, damos por terminada la monografía y hacemos un balance de los resultados obtenidos. Desde que el proceso haya sido realizado de forma juiciosa y conciente, nos sentimos satisfechos por haber logrado realizar de forma total o parcial lo propuesto, o simplemente por haber aclarado que los objetivos propuestos no pueden ser realizados. Es común que, después de todo este proceso, uno termine tan agotado que piense que nunca va a volver a emprender una de estas aventuras, sin embargo, volvemos a soñar.

Soy conciente del contenido puramente técnico del presente trabajo, por lo tanto, quiero utilizar este espacio para contar de una forma más humana el trabajo realizado: la monografía comienza al relacionar algunos conocimientos e interrogantes adquiridos en el marco de la Especialización en Lógica y Filosofía, con algunos conocimientos e interrogantes que venían de tiempo atrás. Específicamente, al relacionar el conocimiento de algunos fundamentos y sistemas de lógica paraconsistente con el modelo de máquina de Turing, con el objetivo de construir un modelo de hipercomputación (intuitivamente, un modelo de computación más potente que las máquinas de Turing). Por intermedio del profesor Andrés Sicard, quién fue mi asesor, conocí dos trabajos que hacen referencia a la posibilidad de construir modelos de hipercomputación utilizando lógicas paraconsistentes [21, 25], uno de ellos de su autoría. Al compartir intereses con el profesor Sicard, decidimos emprender la aventura de construir una teoría de máquinas de Turing paraconsistentes. Concientes desde el comienzo de los riesgos que corriamos y de que, así no se obtuviera un modelo de hipercomputación, el trabajo podría resultar interesante. Esto fue precisamente lo que sucedió, el modelo obtenido no resultó ser un modelo de hipercomputación, sin embargo, se logró construir un nuevo modelo de máquina de Turing, con unas características muy particulares, que puede ser el punto de partida de nuevas investigaciones. Quizás el camino elegido no haya sido el indicado o quizas el objetivo no sea alcanzable, lo único cierto es que en esta aventura aun quedan muchos caminos por explorar.

Al ver finalizada esta monografía (o por lo menos esta etapa del sueño), vienen a mi cabeza muchas personas que hicieron que esto fuera posible, a ellos deseo expresar mis agradecimientos: al profesor Andrés Sicard, por su entrega como asesor y por sus valiosos aportes en la realización del presente trabajo; a los profesores Manuel Sierra, Raúl Gomez, Andrés Sicard (de nuevo) y demás profesores que, compartiendo generosamente sus conocimientos, nos ayudaron a descubir, o por lo menos palpar más de cerca, el inmenso y facinante mundo de la lógica con sus implicaciones y fundamentaciones filosóficas; al profesor Walter Carnielli, director del CLE (Centro de Lógica, Epistemología e História da Ciência, Campinas - Brasil), por mostrar interés por el presente trabajo y por facilitarme algunos de sus trabajos; a Hugo, Carmenza, Marta Elena, María Elena, Jorge Hernán y Javier por ser amigos, más que compañeros, que hicieron de la especialización un disfrute; a mi mamá, hermanas, amigos y amigas por su comprensión y apoyo; a Jaqueline, por soportarme; a la universidad EAFIT por patrocinar mi estudio y a todas las personas que de una u otra forma colaboraron en la realización del presente trabajo y en el momento se me escapan.

El autor, noviembre de 2003.

Índice general

Introducción

Con el interés de crear un modelo de hipercomputación, mediante la construcción de un modelo de máquina de Turing bajo una lógica paraconsistente, dentro de la propuesta de monografía, se plantea en la formulación del problema:

> "[mediante la axiomatización de las máquinas de Turing]...aventurar ciertas generalizaciones a la definición de máquina de Turing. Generalizaciones que den lugar a la generación de inconsistencias, las cuales deberán ser manejadas seleccionando como lógica subyacente una lógica paraconsistente, para evitar así la trivialización. De este modo, teniendo en cuenta que varias lógicas paraconsistentes actuan igual a la lógica clásica ante la ausencia de contradicciones y no trivializan ante la presencia de ellas, se puede pensar en la posibilidad de ampliar la noción de computabilidad".

Además, se propone como objetivo general:

> "mediante la axiomatización y sustitución de la lógica subyacente a la noción de máquina de Turing, construir algunas definiciones de MTP y determinar si los modelos obtenidos son modelos de hipercomputación"

y como objetivos específicos:

1. Definir un procedimiento para construir teorías axiomáticas que den cuenta del comportamiento de máquinas de Turing deterministas.
2. Definir un procedimiento para construir teorías axiomáticas que den cuenta del comportamiento de máquinas de Turing no deterministas.
3. Mediante la generalización o modificación de la noción de máquina de Turing, producir teorías de máquinas de Turing contradictorias.

4. Definir un procedimiento para construir teorías axiomáticas que den cuenta del comportamiento de MTPs.

5. Establecer el poder computacional de los modelos de MTPs obtenidos, demostrando, si es posible, equivalencias computacionales con otros modelos de computación o hipercomputación existentes.

Lo planteado como propuesta de monografía, puede ser enmarcado dentro de los principios pragmáticos de la razón, propuestos por Newton da Costa [19, p. 370], de la siguiente manera: el primer principio pragmático de la razón, denominado 'principio de la sistematización', propone:

> "*La razón siempre se expresa por medio de una lógica.* Nótese que este principio, incluso si la razón en su ejercicio lícito y principal sólamente se expresase a través de una única lógica, permanecería válido. Además, tal vez fuese mejor formularlo aseverando que en los contextos racionales se encuentra siempre, de modo explícito o implícito, un sistema lógico." [19, p. 370].

En los capítulos 1 y 2 precisamente se expresa, por medio de un sistema lógico[1], las nociones de máquina de Turing determinista (MTD) y máquina de Turing no determinista (MTND). Se busca que el sistema lógico sea válido y completo con respecto a una estructura matemática que representa una computación específica de la máquina de Turing en cuestión, persiguiendo de esta manera el tercer principio pragmático de la razón, denominado 'principio de adecuación':

> "*La lógica subyacente a un contexto dado debe ser la que mejor se le adapte.* Lo que significa el principio precedente se resume así: al estudiar determinado dominio de objetos, reales o ideales, se debe escoger el sistema de categorías racionales y de leyes universales que la reglamenten que mejor se ajuste a esos objetos." [19, p. 371].

El segundo principio pragmático de la razón, denominado 'principio de la unicidad', plantea:

> "*En un contexto dado, la lógica subyacente es única.* Hablando metafóricamente, este segundo principio nos asegura que, una vez fijadas las reglas de juego, ellas no deben ser alteradas. Una alteración modificaría inmediatamente el juego inicial,

[1] Una teoría en lógica de predicados de primer orden clásica.

> transformándolo en otro. De modo más exacto, las modificaciones de la lógica subyacente en un contexto racional lo convierten en un contexto distinto.” [19, p. 371].

En el capítulo 3 se modifica la lógica subyacente al ‘juego’ de las máquinas de Turing, por una lógica paraconsistente, se reinterpreta a la luz de la nueva lógica las reglas del ‘juego’, modificando de esta manera el contexto de la noción de máquina de Turing, esperando obtener así un modelo de hipercomputación.

Para dar cumplimiento a los objetivos propuestos, los capítulos fueron estructurados de la siguiente manera: en el capítulo 1 se define el proceso de axiomatización de las MTDs. Inicialmente, se presenta el método de axiomatización definido por Jorge Boolos y Richard Jeffrey (B-J), por medio del cual se obtienen teorías $\Delta_{LC}(\mathcal{M}(n))$ que axiomatizan el comportamiento de la máquina de Turing $\mathcal{M}$, para el dato de entrada n (sección 1.1). Luego se demuestra que dicho método de axiomatización produce teorías incompletas de MTDs, por lo tanto, se modifica el método de axiomatización adicionando nuevos esquemas axiomáticos, obteniendo teorías $\Delta'_{LC}(\mathcal{M}(n))$, y se demuestra que para las MTDs, las teorías $\Delta'_{LC}(\mathcal{M}(n))$ son completas y válidas con respecto a una estructura matemática que se define para $\mathcal{M}(n)$, la cual se simboliza por $\mu(\mathcal{M}(n))$ (sección 1.2). Tanto $\Delta_{LC}(\mathcal{M}(n))$ como $\Delta'_{LC}(\mathcal{M}(n))$ tienen como lógica subyacente la lógica de predicados de primer orden clásica.

En el capítulo 2 se demuestra que el método de axiomatización definido en el capítulo 1, al ser aplicado a MTNDs, produce en algunos casos teorías $\Delta'_{LC}(\mathcal{M}(n))$ contradictorias y por lo tanto triviales, teniendo en cuenta la lógica subyacente. Se muestra que estas contradicciones se presentan debido a que las teorías $\Delta'_{LC}(\mathcal{M}(n))$ no axiomatizan de forma adecuada el comportamiento de una MTND, al encontrarse en una situación en la cual existen varias instrucciones que pueden ser ejecutadas, caso en el cual la MTND elige y ejecuta sólo una de ellas. Se modifica el método de axiomatización para corregir el problema, obteniendose teorías $\Delta''_{LC}(\mathcal{M}(n))$. Además, se demuestra que para las MTNDs, las teorías $\Delta''_{LC}(\mathcal{M}(n))$ son completas y válidas con respecto a la estructura matemática que se define para $\mathcal{M}(n)$.

En el capítulo 3 en lugar de modificar las teorías $\Delta'_{LC}(\mathcal{M}(n))$ (imponiendo la restricción de que $\mathcal{M}(n)$, al encontrarse en una situación en la cual puede ejecutar varias instrucciones, elige y ejecuta sólo una de ellas) para solucionar el problema de la trivialización al axiomatizar MTNDs, se sustituye la lógica subyacente a $\Delta'_{LC}(\mathcal{M}(n))$ por una lógica paraconsistente ($C_1^=$), obteniendose teorías $\Delta'_{C_1^=}(\mathcal{M}(n))$ (sección 3.2). A partir de las teorías

$\Delta'_{C_1^=}(\mathcal{M}(n))$ se redefine el comportamiento de una máquina de Turing, al modelo de máquina de Turing obtenido se le denomina 'máquina de Turing paraconsistente' (MTP) (sección 3.3). En la sección 3.4 se abre la posibilidad de definir condiciones en las instrucciones de las MTPs. La definición de las MTPs busca obtener un modelo de hipercomputación, sin embargo, el resultado obtenido es negativo (sección 3.5). La figura 1 equematiza el trabajo realizado.

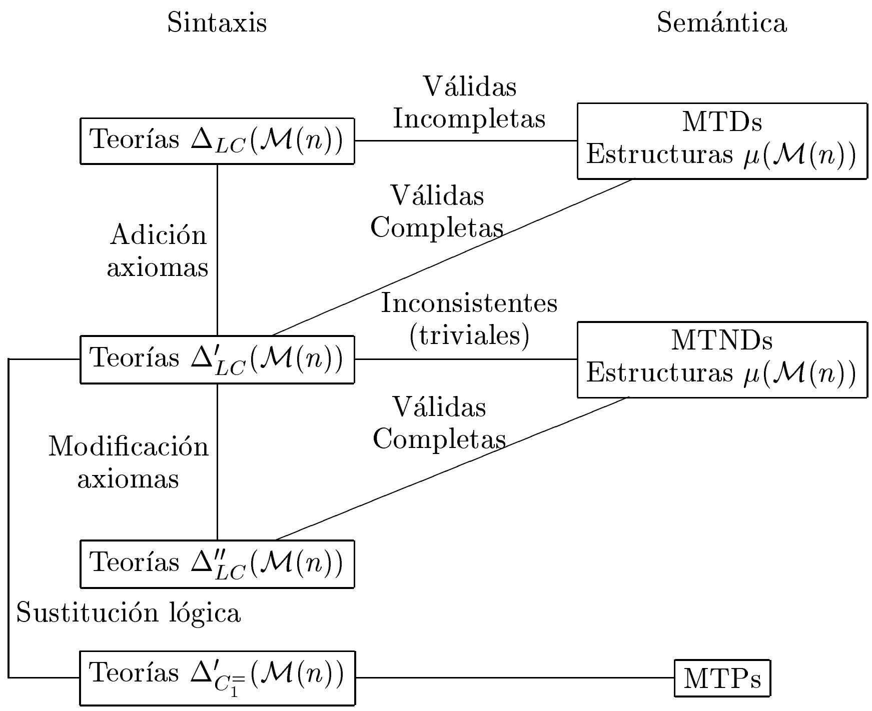

Figura 1: Esquema del trabajo realizado

Finalmente, es necesario aclarar que el presente trabajo supone un conocimiento previo de los conceptos de máquina de Turing, hipercomputación y lógica paraconsistente. Sin embargo, en los anexos A, B y C, en forma respectiva, se presenta una breve descripción de cada uno de estos conceptos. En el anexo C que trata sobre lógica paraconsistente, se presenta también una breve descripción de los sistemas C_n, C_n^* y $C_n^=$ de da Costa, y en forma particular $C_1^=$, que será el sistema utilizado en la construcción del modelo de MTP.

Capítulo 1

Axiomatización de máquinas de Turing deterministas

Como se indicó en la introducción, el presente trabajo supone un conocimiento previo de los conceptos de máquina de Turing, hipercomputación y lógica paraconsistente. Sin embargo, en los anexos A, B y C, en forma respectiva, se presenta una breve descripción de cada uno de estos conceptos. Bajo este supuesto, se comienza con la definición de un procedimiento de axiomatización de máquinas de Turing deterministas.

Con el objetivo de demostrar la indecidibilidad de la lógica de predicados de primer orden clásica, George Boolos y Richard Jeffrey [2] definen un procedimiento para construir una teoría axiomática Δ y una fórmula H (se seguirá haciendo referencia a dicho procedimiento utilizando la sigla B-J), a partir de la descripción[1] de una máquina de Turing $\mathcal{M}$ y un dato de entrada n (lo que se seguirá simbolizando por $\mathcal{M}(n)$). De forma tal que, bajo una interpretación $\mathcal{I}$, la teoría Δ describe la operación de $\mathcal{M}(n)$ y H indica que la máquina eventualmente se detiene. De este modo, $\Delta \vdash H$ si y sólo si $\mathcal{M}(n)$ se detiene. Por lo tanto, si se supone que existe un procediemiento efectivo para determinar si $\Delta \vdash H$, se tendría como consecuencia que el problema de la parada de una máquina de Turing es decidible. Como se conoce que el problema de la parada de una máquina de Turing es indecidible [2, 16, 26] se concluye que determinar si $\Delta \vdash H$ también es indecidible. Por lo tanto, la lógica de predicados de primer orden clásica es indecidible[2].

[1] Definición de las instrucciones de la máquina en forma de quádruplas, grafos, tablas o cualquier otra representación adecuada.

[2] Como Δ es una teoría finita, $\Delta \vdash H$ si y sólo si $\vdash \left(\bigwedge_{i=1}^{n} \delta_i\right) \to H$, donde $\delta_i \in \Delta$ y n es el cardinal de Δ. Por lo tanto, si se supone que la lógica de predicados de primer orden

Para el objetivo perseguido por Boolos y Jeffrey, el procedimiento de axiomatización que definen resulta ser adecuado, sin embargo, si se desea tener un procedimiento de axiomatización para máquinas de Turing deterministas, de forma tal que la teoría resultante pueda dar cuenta exacta del comportamiento de la máquina de Turing en cuestión, se tiene que el método de axiomatización B-J produce, por lo menos en algunos casos, teorías que no cumplen con este objetivo y son, en este sentido, incompletas.

En las siguientes sesiones se describe el método de axiomatización B-J, se demuestra por que dicho procedimiento en algunos casos produce teorías incompletas de máquinas de Turing deterministas y se realizan algunos ajustes al procedimiento con el fin de producir teorías completas.

1.1. Método de axiomatización B-J

Para definir el método de axiomatización de máquinas de Turing, Boolos y Jeffrey [2] imaginan que las casillas de la cinta están numeradas por medio de los números enteros y que el tiempo está dividido en series de momentos t en los cuales la máquina ejecuta exactamente una de sus operaciones, existiendo un momento 0 en el cual la máquina comienza su ejecución leyendo la casilla 0. Los momentos de tiempo se supone que se extienden infinitamente hacia el futuro y el pasado, al igual que la cinta se extiende infinitamente hacia la derecha y la izquierda. Además, asumen que la máquina es 'prendida' en el momento 0 y 'apagada' en el primer momento (si existe alguno) despúes de que la máquina se detiene y que en todos los tiempos negativos y todos los tiempos después del primero (si existe alguno) en el que la máquina se detiene, la máquina no está en ningún estado, no está leyendo ninguna de las casillas y no tiene ningún símbolo (ni siquiera el vacío) en ninguna de las casillas de la cinta[3] (figura 1.1).

Una vez establecidos los supuestos descritos anteriormente, Boolos y Jeffrey definen el lenguaje de primer orden $\mathcal{L} = \{Q_1, Q_2, \ldots, Q_n, S_0, S_1, \ldots, S_m, <, ', 0\}$, el cual utilizan para construir la teoría[4] $\Delta_{LC}(\mathcal{M}(n))$ que da cuenta del comportamiento de $\mathcal{M}(n)$. En $\mathcal{L}$, los símbolos Q_i, S_j y $<$ son símbolos de predicado de aridad 2, $'$ es un

clásica es decidible, se tiene que $\vdash \left(\bigwedge_{i=1}^{n} \delta_i\right) \rightarrow H$ sería decidible y por lo tanto $\Delta \vdash H$ también sería decidible.

[3]Estos últimos supuestos no se ven reflejados en el proceso de axiomatización B-J, como se verá en la siguiente sección.

[4]Se adiciona el subíndice LC a Δ para resaltar el hecho de que la lógica subyacente a la teoría Δ es la lógica clásica y se adiciona el parámetro $\mathcal{M}(n)$ para hacer claridad sobre la máquina de Turing y el dato de entrada que está axiomatizando la teoría.

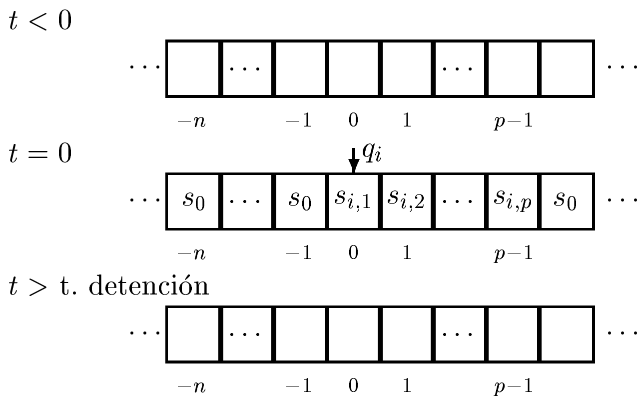

Figura 1.1: Supuestos axiomatizacion B-J ($s_0, s_{i,j} \in \Sigma$)

símbolo de función de aridad 1 y el símbolo 0 es un símbolo de constante.

Bajo la interpretación intensional[5] $\mathcal{I}$ de las sentencias presentes en $\Delta_{LC}(\mathcal{M}(n))$, las variables se interpretan como números enteros y los símbolos de $\mathcal{L}$ se interpretan así:

- $Q_i(t, x)$ se interpreta como el hecho de que la máquina de Turing $\mathcal{M}$ se encuentra en el estado q_i, en el tiempo t, en la posición x.
- $S_j(t, x)$ se interpreta como el hecho de que la máquina de Turing $\mathcal{M}$ se encuentra leyendo el símbolo s_j, en el tiempo t, en la posición x.
- $<$ se interpreta como la relación menor que en los números enteros.
- $'$ se interpreta como la función sucesor.
- 0 se interpreta como el número 0.

En la construcción de $\Delta_{LC}(\mathcal{M}(n))$ se incluyen sentencias para definir las propiedades de los símbolos $<$ y $'$, se incluye una nueva sentencia para definir cada una de las instrucciones de $\mathcal{M}$ y una sentencia para definir la situación inicial de $\mathcal{M}$ (estado inicial, posición sobre la cinta y dato de entrada n). En beneficio de la claridad, inicialmente se definirá una teoría Δ_{LC}^{-1} por medio de la cual se establecerán las propiedades de los símbolos

[5] "Al considerar a los predicados desde el punto de vista *extensional* se dice que aluden, o mejor, que denotan *clases* o *conjuntos*. Pero cuando se les contempla desde al ángulo *intensional* se dice que designan *propiedades* o *notas* de los objetos" [15, p. 163].

$<$ y $'$ y se demostrarán algunos teoremas. Luego, se indicará como se puede obtener $\Delta_{LC}(\mathcal{M}(n))$ extendiendo Δ_{LC}^{-1}.

Definición 1.1 (Teoría Δ_{LC}^{-1}). Δ_{LC}^{-1} *está constituida por los siguientes axiomas:*

$$\forall z \exists x (z = x'), \tag{A1}$$
$$\forall z \forall x \forall y (((z = x') \wedge (z = y')) \rightarrow (x = y)), \tag{A2}$$
$$\forall x \forall y \forall z (((x < y) \wedge (y < z)) \rightarrow (x < z)), \tag{A3}$$
$$\forall x (x < x'), \tag{A4}$$
$$\forall x \forall y ((x < y) \rightarrow (x \neq y)), \tag{A5}$$

El axioma (A1) *establece la existencia de sucesor para todo número. El axioma* (A2) *establece la inyectividad de la función sucesor. El axioma* (A3) *establece la propiedad de transitividad de la relación $<$. El axioma* (A4) *establece la relación entre la función $'$ y la relación $<$, especificando que el sucesor de un número siempre es mayor que él. El axioma* (A5) *establece la propiedad antirreflexiva de la relación $<$.*

Teorema 1.1. $\Delta_{LC}^{-1} \vdash \forall x (x' \neq x)$.

Demostración. [6]

1.	$x < x'$.	IU(x/x) en (A4).
2.	$x \neq x'$.	IU$(x/x, y/x')$ en (A5) y MP con 1.
3.	$\forall x (x' \neq x)$.	GU en 2.

□

Teorema 1.2. $\Delta_{LC}^{-1} \vdash \forall x \forall y ((x \neq y) \rightarrow (x' \neq y'))$.

Demostración.

1.	$x' = y'$.	Supesto (SUP).
2.	$((x' = x') \wedge (x' = y')) \rightarrow (x = y)$.	IU$(z/x', x/x, y/y)$ en (A2).

[6] Las abreviaturas utilizadas en las explicaciones de las demostraciones están especificadas en el apéndice D. En las reglas de instanciación universal (IU) y de instanciación existencial (IE), para mayor claridad, se especifican entre paréntesis los elementos sustituidos.

3.	$x = y.$	Identidad, CONJ con 1 y MP con 2.
4.	$(x' = y') \rightarrow (x = y).$	TD 1-3.
5.	$(x \neq y) \rightarrow (x' \neq y').$	CTRP en 4.
6.	$\forall x \forall y((x \neq y) \rightarrow (x' \neq y')).$	GU en 5.

$\square$

Definición 1.2 (Teoría $\Delta_{LC}(\mathcal{M}(n))$). *Sea $\mathcal{M} =< Q, \Sigma, M, I >$ una máquina de Turing y $n = s_{i,1}s_{i,2}\ldots s_{i,p}$ con $s_{i,j} \in \Sigma$ la sucesión o dato de entrada para $\mathcal{M}$. La teoría $\Delta_{LC}(\mathcal{M}(n))$ se construye adicionando a Δ_{LC}^{-1} un axioma por cada instrucción $i_j \in I$ y un axioma especificando la situación inicial de $\mathcal{M}$ (estado inicial, posición sobre la cinta y dato de entrada n).*

Sea[7] $m = \overline{\overline{\Sigma}}$, los axiomas de las instrucciones $i_j \in I$ se construyen de acuerdo con los esquemas:

- *Si i_j es de la forma[8] (I), entonces:*

$$\forall t \forall x \Bigg(\Big(Q_i(t,x) \wedge S_j(t,x) \Big) \rightarrow \Big(Q_l(t',x) \wedge S_k(t',x) \wedge \forall y \Big((y \neq x) \rightarrow \Big(\bigwedge_{i=0}^{m} \big(S_i(t,y) \rightarrow S_i(t',y) \big) \Big) \Big) \Big) \Bigg), \quad \text{(A5+j (I))}$$

este esquema axiomático indica que si la máquina de Turing $\mathcal{M}$, en el instante de tiempo t, se encuentra en la posición x, en el estado q_i, leyendo el símbolo s_j, en el siguiente instante de tiempo t', la máquina estará en la misma posición x, en el estado q_l y leyendo el símbolo s_k. Además, en todas las posiciones $y \neq x$ se tiene que el símbolo en el instante de tiempo t' será el mismo del instante anterior t.

[7] $\overline{\overline{A}}$ denota el cardinal del conjunto A.

[8] Las formas o tipos de instrucciones están especificadas en el apéndice A, sección A.1.

- *Si i_j es de la forma* (II), *entonces:*

$$\forall t\forall x\Bigg(\Big(Q_i(t,x)\wedge S_j(t,x)\Big)\rightarrow\Big(Q_l(t',x')\wedge \forall y\Big(\bigwedge_{i=0}^{m}\big(S_i(t,y)\rightarrow S_i(t',y)\big)\Big)\Big)\Bigg), \quad \text{(A5+j (II))}$$

  este esquema axiomático indica que si la máquina de Turing $\mathcal{M}$, en el instante de tiempo t, se encuentra en la posición x, en el estado q_i, leyendo el símbolo s_j, en el siguiente instante de tiempo t', la máquina estará en la siguiente posición hacia la derecha x', en el estado q_l. Además, en todas las posiciones de la cinta, los símbolos en el instante de tiempo t' serán los mismo del instante anterior t.

- *Si i_j es de la forma* (III), *entonces:*

$$\forall t\forall x\Bigg(\Big(Q_i(t,x')\wedge S_j(t,x')\Big)\rightarrow\Big(Q_l(t',x)\wedge \forall y\Big(\bigwedge_{i=0}^{m}\big(S_i(t,y)\rightarrow S_i(t',y)\big)\Big)\Big)\Bigg), \quad \text{(A5+j (III))}$$

  la interpretación de este esquema axiomático es similar al caso anterior.

El axioma para especificar la situación inicial de la máquina, suponiendo que la máquina comienza en el estado q_1, sobre la posición 0 *de la cinta, con la entrada $n = s_{i,1}s_{i,2}\dots s_{i,p}$, se construye de acuerdo con el esquema:*

$$Q_1(0,0)\ \wedge\ \left(\bigwedge_{j=1}^{p} S_{i,j}(0,0^{j-1})\right)\ \wedge\ \forall y\left(\left(\bigwedge_{j=1}^{p} y\neq 0^{j-1}\right)\rightarrow S_0(0,y)\right), \quad (\text{A5}+\overline{\overline{I}}+1)$$

donde 0^{j-1} indica la aplicación de $j-1$ veces la función sucesor a la constante 0.

La fórmula H que indica que $\mathcal{M}$ se detiene, se construye haciendo la disyunción de todas las fórmulas que indican que existe un tiempo t y una posición x, en los cuales la máquina se encuentra en el estado q_i leyendo el símbolo s_j y no existe ninguna instrucción $i_k \in I$ que comience por *los símbolos $q_i s_i$ (lo que se simboliza con el subíndice '$q_i s_j$ no inic.' en la disyuntoria). El esquema para la construcción de H es:*

$$\bigvee_{q_i s_j \text{ no inic.}} (\exists t \exists x (Q_i(t,x) \wedge S_j(t,x))) \,. \tag{H}$$

Ejemplo 1.1. *Sea $\mathcal{M}_1 =< Q, \Sigma, M, I >$ una máquina de Turing con $Q = \{q_1, q_2\}$, $\Sigma = \{s_0, s_1\}$, $I = \{i_1, i_2\}$, donde $i_1 = q_1 s_1 R q_1$ y $i_2 = q_1 s_0 s_1 q_2$ (figura 1.2) y sea $n_1 = s_1$ la secuencia de entrada para $\mathcal{M}_1$.*

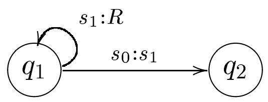

Figura 1.2: Grafo máquina $\mathcal{M}_1$

La teoría $\Delta_{LC}(\mathcal{M}_1(n_1)) = \Delta_{LC}^{-1} \cup \{(\mathrm{A6}), (\mathrm{A7}), (\mathrm{A8})\}$, donde:

$$\forall t \forall x \Bigg(\Big(Q_1(t,x) \wedge S_1(t,x) \Big) \to \Big(Q_1(t',x') \wedge \forall y \Big(\big(S_0(t,y) \to S_0(t',y) \big) \wedge \big(S_1(t,y) \to S_1(t',y) \big) \Big) \Big) \Bigg), \tag{A6}$$

$$\forall t \forall x \Bigg(\Big(Q_1(t,x) \wedge S_0(t,x) \Big) \to \Big(Q_2(t',x) \wedge S_1(t',x) \wedge \forall y \Big((y \neq x) \to \big((S_0(t,y) \to S_0(t',y)) \wedge (S_1(t,y) \to S_1(t',y)) \big) \Big) \Big) \Bigg), \tag{A7}$$

$$Q_1(0,0) \wedge S_1(0,0) \wedge \forall y \left((y \neq 0) \to S_0(0,y) \right) . \tag{A8}$$

Con la teoría $\Delta_{LC}(\mathcal{M}_1(n_1))$ se puede demostrar, entre muchas otras cosas, que $\mathcal{M}_1$ con la entrada n_1 se detiene (teorema 1.3). Puesto que no hay instrucciones en $\mathcal{M}_1$ que comiencen por $q_2 s_0$ ni por $q_2 s_1$, la fórmula H que indica que $\mathcal{M}_1$ se detiene se define así:

$$H \equiv \exists t \exists x \left(Q_2(t,x) \wedge S_0(t,x) \right) \vee \exists t \exists x \left(Q_2(t,x) \wedge S_1(t,x) \right) . \tag{H}$$

Teorema 1.3. $\Delta_{LC}(\mathcal{M}_1(n_1)) \vdash H$.

Demostración.

1.	$Q_1(0,0) \wedge S_1(0,0)$.	SIMP en (A8).
2.	$\forall y((y \neq 0) \rightarrow S_0(0,y))$.	SIMP en (A8).
3.	$Q_1(0',0')$.	IU($t/0, x/0$) en (A6), MP con 1 y SIMP.
4.	$\forall y((S_0(0,y) \rightarrow S_0(0',y)) \wedge (S_1(0,y) \rightarrow S_1(0',y))))$.	IU($t/0, x/0$) en (A6), MP con 1 y SIMP.
5.	$(0' \neq 0) \rightarrow S_0(0,0')$.	IU($y/0'$) en 2.
6.	$S_0(0,0')$.	IU($x/0$) en teorema (1.1) y MP con 5.
7.	$S_0(0,0') \rightarrow S_0(0',0')$.	IU($y/0'$) y SIMP en 4.
8.	$Q_1(0',0') \wedge S_0(0',0')$.	MP en 6 y 7 y CONJ con 3.
9.	$Q_2(0'',0') \wedge S_1(0'',0')$.	IU($t/0', x/0'$) en (A7), MP con 8 y SIMP.
10.	H.	GE en 9 y ADIC.

□

1.2. Ajustes al método de axiomatización B-J para máquinas de Turing deterministas

Por medio de un ejemplo, se demuestra la necesidad de incluir nuevos axiomas en el método de axiomatización B-J, para poder así demostrar fórmulas, que bajo la interpretación intensional $\mathcal{I}$ son verdaderas, pero no son demostrables a partir de $\Delta_{LC}(\mathcal{M}(n))$. Al adicionar dichos axiomas a $\Delta_{LC}(\mathcal{M}(n))$ se obtiene una nueva teoría $\Delta'_{LC}(\mathcal{M}(n))$, que como se verá en el siguiente capítulo, no es adecuada para axiomatizar máquinas de Turing no deterministas.

Ejemplo 1.2. *Sea* $\mathcal{M}_2 =< Q, \Sigma, M, I >$ *una máquina de Turing con* $Q = \{q_1, q_2\}$, $\Sigma = \{s_0, s_1\}$, $I = \{i_1, i_2\}$, *donde* $i_1 = q_1 s_1 s_0 q_2$ *y* $i_2 = q_2 s_0 s_0 q_2$ *(figura 1.3) y sea* $n_2 = s_1$ *la secuencia de entrada para* $\mathcal{M}_2$.

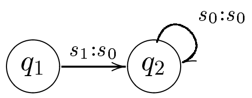

Figura 1.3: Grafo máquina $\mathcal{M}_2$

Entonces $\Delta_{LC}(\mathcal{M}_2(n_2)) = \Delta_{LC}^{-1} \cup \{(\mathrm{A6}), (\mathrm{A7}), (\mathrm{A8})\}$, *donde:*

$$\forall t \forall x \Bigg(\Big(Q_1(t,x) \wedge S_1(t,x) \Big) \rightarrow \Big(Q_2(t',x) \wedge S_0(t',x) \wedge \\ \forall y \Big((y \neq x) \rightarrow \big((S_0(t,y) \rightarrow S_0(t',y)) \wedge (S_1(t,y) \rightarrow S_1(t',y)) \big) \Big) \Big) \Bigg), \tag{A6}$$

$$\forall t \forall x \Bigg(\Big(Q_2(t,x) \wedge S_0(t,x) \Big) \rightarrow \Big(Q_2(t',x) \wedge S_0(t',x) \wedge \\ \forall y \Big((y \neq x) \rightarrow \big((S_0(t,y) \rightarrow S_0(t',y)) \wedge (S_1(t,y) \rightarrow S_1(t',y)) \big) \Big) \Big) \Bigg), \tag{A7}$$

$$Q_1(0,0) \wedge S_1(0,0) \wedge \forall y \left((y \neq 0) \rightarrow S_0(0,y) \right). \tag{A8}$$

Realizando un seguimiento de $\mathcal{M}_2(n_2)$, *se tiene que* $\mathcal{M}_2$ *en el instante de tiempo* $t = 0$ *ejecuta la instrucción* i_1, *borrando el símbolo (escribiendo* s_0*) que hay en la casilla* 0 *y pasando al estado* q_2. *A partir del tiempo* $t = 1$, $\mathcal{M}_2$ *ejecutará la instrucción* i_2, *quedandose en un ciclo infinito. Por lo tanto, la teoría* $\Delta_{LC}(\mathcal{M}_2(n_2))$ *debe poder demostrar que a partir del tiempo* $t = 1$ *la máquina siempre va a estar en la posición* 0, *en el estado* q_2, *leyendo el símbolo* s_0, *como en efecto lo demuestra (teorema 1.4). Además, por el hecho anterior,* $\Delta_{LC}(\mathcal{M}_2(n_2))$ *debería poder demostrar que* $\mathcal{M}_2(n_2)$ *nunca va a estar en el estado* q_1 *leyendo el símbolo* s_0, *ni tampo va a estar en el estado* q_2 *leyendo el símbolo* s_1, *por lo tanto, nunca se va a detener, lo cual se expresa con la fórmula:*

$$\neg H \equiv \neg \big(\exists t \exists x \left(Q_1(t,x) \wedge S_0(t,x) \right) \vee \exists t \exists x \left(Q_2(t,x) \wedge S_1(t,x) \right) \big), \tag{$\neg$H}$$

sin embargo, debido a que en $\Delta_{LC}(\mathcal{M}_2(n_2))$ no existen axiomas para asegurar la unicidad de símbolo en cada casilla, estado y posición sobre la cinta, del hecho de que $\Delta_{LC}(\mathcal{M}_2(n_2)) \vdash Q_2(t,0)$, para $t \geq 1$, no se puede deducir que $\Delta_{LC}(\mathcal{M}_2(n_2)) \vdash \neg Q_1(t,0)$, para $t \geq 1$, ni tampoco que $\Delta_{LC}(\mathcal{M}_2(n_2)) \vdash \neg Q_1(t,x) \wedge \neg Q_2(t,x)$, para $t \geq 1$ y $x \neq 0$. Para los símbolos sucede algo similar, además, para los momentos en que la máquina está 'apagada' ($t < 0$ para este caso), no se puede demostrar lo establecido en los supuestos del método de axiomatización B-J. Por lo tanto, es necesario adicionar a $\Delta_{LC}(\mathcal{M}_2(n_2))$ nuevos axiomas para asegurar dichas unicidades y supuestos y poder así demostrar $\neg H$, como se muestra en el teorema 1.5.

Teorema 1.4. $\Delta_{LC}(\mathcal{M}_2(n_2)) \vdash (Q_2(t,0) \wedge S_0(t,0))$, *para todo* $t \geqslant 1$.

Demostración. Utilizando inducción matemática se tiene:

1. Paso base $t = 1$:

1.1.	$Q_2(0',0) \wedge S_0(0',0)$.	SIMP en (A8), IU$(t/0, x/0)$ en (A6), MP y de nuevo SIMP.

2. Paso inductivo:

2.1.	$Q_2(0^k,0) \wedge S_0(0^k,0)$.	Hipótesis inductiva (HI).
2.2.	$Q_2(0^{k+1},0) \wedge S_0(0^{k+1},0)$.	IU$(t/k{+}1, x/0)$ en (A7), MP con a y SIMP.

3. $\Delta_{LC}(\mathcal{M}_2(n_2)) \vdash (Q_2(t,0) \wedge S_0(t,0))$, para todo $t \geqslant 1$. — Inducción matemática (IM).

□

Teorema 1.5. $\Delta'_{LC}(\mathcal{M}_2(n_2)) \vdash \neg H$.

Demostración. Demostrar $\neg H$ es equivalente a demostrar $\forall t \forall x(\neg Q_1(t,x) \vee \neg S_0(t,x)) \wedge \forall t \forall x(\neg Q_2(t,x) \vee \neg S_1(t,x))$. Se demuestra sólo el primer elemento de la conjunción, la demostración del segundo elemento es similar. Sean $n = \overline{\overline{Q}}$ y $m = \overline{\overline{\Sigma}}$:

1. Para $t < 0$: En los supuestos dados por Boolos y Jeffrey antes de definir el método de axiomatización, se asume que la máquina es 'prendida' en el momento 0 y que, por lo tanto, en todos los tiempos negativos la máquina no está en ningún estado, leyendo ninguna de las casillas

y sin ningún símbolo (ni siquiera el vacío) en ninguna de las casillas de la cinta. Este supuesto no está reflejado en ninguno de los axiomas de $\Delta_{LC}(\mathcal{M}_2(n_2))$, por lo tanto es necesario incluir un nuevo axioma que establezca este supuesto en la teoría para poder continuar con la demostración. Dicho axioma se construye de acuerdo con el esquema:

$$\forall t \forall x \left((t < 0) \rightarrow \left(\left(\bigwedge_{i=1}^{n} \neg Q_i(t,x) \right) \wedge \left(\bigwedge_{j=0}^{m} \neg S_j(t,x) \right) \right) \right). \tag{A5+$\overline{\overline{I}}$+2}$$

Para el caso de $\mathcal{M}_2$ se tiene:

$$\forall t \forall x \Big((t < 0) \rightarrow \big(\neg Q_1(t,x) \wedge \neg Q_2(t,x) \wedge \neg S_0(t,x) \wedge \neg S_1(t,x) \big) \Big). \tag{A9}$$

Con este nuevo axioma, con $t < 0$, aplicando MP y SIMP se pueden obtener $\neg Q_1(x,t)$ y $\neg S_0(x,t)$. Aplicando ADIC en cualquiera de los dos se tiene $\neg Q_1(t,x) \vee \neg S_0(t,x)$.

2. Para $t = 0$:

   2.1. Para $x = 0$: Aplicando SIMP en (A8) se tiene $S_1(0,0)$. De acuerdo con la definición de máquina de Turing, sobre cada casilla de la cinta, en cada instante de tiempo t sólo puede existir un símbolo, por lo tanto, teniendo $S_1(0,0)$ se debería poder demostrar $\neg S_0(0,0)$. Sin embargo, $\Delta_{LC}(\mathcal{M}_2(n_2)) \cup \{S_1(0,0)\}$ no permite deducir $\neg S_0(0,0)$. Se hace necesario entonces incluir un nuevo axioma, por cada símbolo s_i, en $\Delta_{LC}(\mathcal{M}_2(n_2))$ para poder asegurar dicha unicidad y poder continuar con la demostración. Dichos axiomas se construyen de acuerdo con el esquema:

$$\forall t \forall x \left(S_i(t,x) \rightarrow \bigwedge_{i \neq j} \neg S_j(t,x) \right). \tag{A5+$\overline{\overline{I}}$+i+3}$$

   Para el caso de $\mathcal{M}_2$ se tiene:

$$\forall t \forall x (S_0(t,x) \rightarrow \neg S_1(t,x)), \tag{A10}$$

$$\forall t \forall x (S_1(t,x) \rightarrow \neg S_0(t,x)). \tag{A11}$$

Con $S_1(0,0)$, aplicando IU$(t/0,x/0)$ en (A11), MP y SIMP se tiene $\neg S_0(0,0)$. Aplicando ADIC se tiene $\neg Q_1(0,0) \vee \neg S_0(0,0)$.

2.2. Para $x \neq 0$: Aplicando SIMP en (A8) se tiene $Q_1(0,0)$. De acuerdo con la definición de máquina de Turing, en cada instante de tiempo t la máquina sólo puede estar en un estado y en una posición (casilla) de la cinta, por lo tanto, teniendo $Q_1(0,0)$ se debería poder demostrar $\neg Q_1(0,x)$ (para todo $x \neq 0$). Sin embargo, $\Delta_{LC}(\mathcal{M}_2(n_2)) \cup Q_1(0,0)$ no permite deducir $\neg Q_1(0,x)$ (para todo $x \neq 0$). Se hace necesario entonces incluir un nuevo axioma en $\Delta_{LC}(\mathcal{M}_2(n_2))$ para poder asegurar dicha unicidad y poder continuar con la demostración. Dicho axioma se construye de acuerdo con el esquema:

$$\forall t \forall x \left(\bigvee_{i=1}^{n} Q_i(t,x) \to \forall y \left(x \neq y \to \bigwedge_{j=1}^{n} \neg Q_j(t,y) \right) \right). \tag{A5+$\overline{\overline{I}}$+$\overline{\overline{\Sigma}}$+3}$$

Para el caso de $\mathcal{M}_2$ se tiene:

$$\begin{aligned} \forall t \forall x \Big(\big(Q_1(t,x) \vee Q_2(t,x)\big) \to \\ \forall y \Big(x \neq y \to \big(\neg Q_1(t,y) \wedge \neg Q_2(t,y)\big)\Big)\Big). \end{aligned} \tag{A12}$$

Aplicando ADIC a $Q_1(0,0)$ se tiene $Q_1(0,0) \vee Q_2(0,0)$. Aplicando IU$(t/0,x/0)$ en (A12) y MP se tiene $\forall y(0 \neq y \to (\neg Q_1(0,y) \wedge \neg Q_2(0,y)))$. Aplicando IU$(y/x)$, conmutando $x \neq 0$ y aplicando MP se tiene $\neg Q_1(0,x) \wedge \neg Q_2(0,x)$. Aplicando SIMP y ADIC se tiene $\neg Q_1(0,x) \vee \neg S_0(0,x)$.

3. Para $t \geqslant 1$:

3.1. Para $x = 0$: Por teorema (1.4) y SIMP se tiene $Q_2(t,0)$ (para todo $t \geqslant 1$). De acuerdo con la definición de máquina de Turing, en cada instante de tiempo t la máquina sólo puede estar en un estado q_i, por lo tanto, teniendo $Q_2(t,0)$ se debería poder demostrar $\neg Q_1(t,0)$. Sin embargo, $\Delta_{LC}(\mathcal{M}_2(n_2)) \cup \{Q_2(t,0)\}$ no permite deducir $\neg Q_1(t,0)$. Se hace necesario entonces incluir un nuevo axioma, por cada estado q_i, en $\Delta_{LC}(\mathcal{M}_2(n_2))$ para poder asegurar dicha unicidad y poder continuar con la demostración. Dichos axiomas se construyen de acuerdo con el esquema:

$$\forall t \forall x \left(Q_i(t,x) \rightarrow \bigwedge_{i \neq j} \neg Q_j(t,x) \right). \qquad (\text{A5}+\overline{\overline{I}}+\overline{\overline{\Sigma}}+\text{i}+3)$$

Para el caso de $\mathcal{M}_2$ se tiene:

$$\forall t \forall x (Q_1(t,x) \rightarrow \neg Q_2(t,x)), \qquad (\text{A13})$$

$$\forall t \forall x (Q_2(t,x) \rightarrow \neg Q_1(t,x)), \qquad (\text{A14})$$

Con $Q_2(t,0)$, aplicando IU$(t/t, x/0)$ en (A14), MP y SIMP se tiene $\neg Q_1(t,0)$. Aplicando ADIC se tiene $\neg Q_1(t,0) \vee \neg S_0(t,0)$.

3.2. Para $x \neq 0$: Por teorema 1.4, SIMP y ADIC se tiene $Q_1(t,0) \vee Q_2(t,0)$ (para todo $t \geqslant 1$). Aplicando IU$(t/t, x/0)$ en (A12) y MP se tiene $\forall y(0 \neq y \rightarrow (\neg Q_1(0,y) \wedge \neg Q_2(0,y)))$. Aplicando IU$(y/x)$, conmunando $x \neq 0$ y aplicando MP se tiene $\neg Q_1(0,x) \wedge \neg Q_2(0,x)$. Aplicando SIMP y ADIC se tiene $\neg Q_1(0,x) \vee \neg S_0(0,x)$.

4. $\forall t \forall x(\neg Q_1(t,x) \vee \neg S_0(t,x))$. Demostración por casos.

□

Adicional a los nuevos axiomas que surgen del ejemplo anterior, se debe incluir en la nueva teoría un axioma para asegurar el supuesto de que, si $\mathcal{M}$ se detiene en el tiempo t, en todo tiempo superior a t, la máquina $\mathcal{M}$ no se encuentra en ningún estado ni con ningún símbolo en las casillas de la cinta (ni siquiera el vacío). Dicho axioma se construye de acuerdo con el esquema:

$$\forall t \forall x \Bigg(\bigg(\bigvee_{q_i s_j \text{ no inic.}} \Big(Q_i(t,x) \wedge S_j(t,x) \Big) \bigg) \rightarrow \forall u \forall y \bigg(t < u \rightarrow \Big(\Big(\bigwedge_{i=1}^{n} \neg Q_i(u,y) \Big) \wedge \Big(\bigwedge_{j=1}^{m} \neg S_j(u,y) \Big) \Big) \bigg) \Bigg). \qquad (\text{A5}+\overline{\overline{I}}+\overline{\overline{\Sigma}}+\overline{\overline{Q}}+4)$$

Para el caso de $\mathcal{M}_2$ del ejemplo anterior, se tiene:

$$\forall t \forall x \Bigg(\bigg(\Big(Q_1(t,x) \wedge S_0(t,x)\Big) \vee \Big(Q_2(t,x) \wedge S_1(t,x)\Big)\bigg) \rightarrow$$
$$\forall u \forall y \bigg(t < u \rightarrow \Big(\neg Q_1(u,y) \wedge \neg Q_2(u,y) \wedge \neg S_0(u,y) \wedge S_1(u,y)\Big)\bigg)\Bigg). \quad \text{(A15)}$$

Definición 1.3 (Teoría $\Delta'_{LC}(\mathcal{M}(n))$). *Sea $\mathcal{M} =< Q, \Sigma, M, I >$ una máquina de Turing y $n = s_{i,1}s_{i,2}\ldots s_{i,p}$ con $s_{i,j} \in \Sigma$ la sucesión o dato de entrada para $\mathcal{M}$. La teoría $\Delta'_{LC}(\mathcal{M}(n))$ se construye adicionando a $\Delta_{LC}(\mathcal{M}(n))$ un axioma de acuerdo con el esquema* (A5+$\overline{\overline{I}}$+2)*, un axioma por cada símbolo $s_i \in \Sigma$ de acuerdo con el esquema* (A5+$\overline{\overline{I}}$+i+3)*, un axioma de acuerdo con el esquema* (A5+$\overline{\overline{I}}$+$\overline{\overline{\Sigma}}$+3) *y un axioma por cada estado $q_i \in Q$ de acuerdo con el esquema* (A5+$\overline{\overline{I}}$+$\overline{\overline{\Sigma}}$+i+3)*.*

Para demostrar la completud de $\Delta'_{LC}(\mathcal{M}(n))$ para MTDs, se define una estructura como semántica para las computaciones de MTDs y se establecen algunos teoremas.

Definición 1.4 (Estructura para $\mathcal{M}(n)$). *Sea $\alpha_0\alpha_1\alpha_2\ldots$ la secuencia (posiblemente infinita) de descripciones instantáneas de $\mathcal{M}(n)$, donde $\alpha_i \xrightarrow{\mathcal{M}} \alpha_{i+1}$ (para $i = 0, 1, 2\ldots$). Se define la estructura $\mu(\mathcal{M}(n)) =< A, Q_1^\mu, Q_2^\mu, \ldots, Q_n^\mu, S_0^\mu, S_1^\mu, \ldots, S_m^\mu, <^\mu, '^\mu, 0^\mu >$ donde*[9]*:*

- $A = \mathbb{Z}$,
- $Q_i^\mu = \{(t,x) \mid q_i$ *está en* α_t *y* $\mathcal{M}$ *se encuentra sobre la posición* $x\}$,
- $S_j^\mu = \{(t,x) \mid s_j$ *está en* α_t *en la posición* $x\} \cup \{(t,y) \mid j = 0,$ *existe* α_t *y* y *es una posición no descrita en* $\alpha_t\}$,
- $<^\mu =$ *relación menor que en* $\mathbb{Z}$,
- $'^\mu =$ *función sucesor en* $\mathbb{Z}$,
- $0^\mu = 0$.

Bajo la función de interpretación δ para $\Delta'_{LC}(\mathcal{M}(n))$, donde $\delta(Q_i(t,x)) = Q_i^\mu$, $\delta(S_j(t,x)) = S_j^\mu$, $\delta(<) =<^\mu$, $\delta(') ='^\mu$ y $\delta(0) = 0^\mu$ se prueba:

[9]Definición adaptada de [28, p. 13]

Teorema 1.6. *Si $\mathcal{M}$ es MTD entonces $\mu(\mathcal{M}(n))$ es modelo de $\Delta'_{LC}(\mathcal{M}(n))$.*

Demostración.

1. La relación menor que ($<$) y la función sucesor ($'$) en los números enteros cumplen con las propiedades establecidas por los axiomas (A1) - (A5) de la definición 1.1, por lo tanto, $\mu(\mathcal{M}(n)) \models$ (A1) - (A5).

2. Si $\mathcal{M}$ contiene una instrucción de la forma (I) entonces en $\Delta'_{LC}(\mathcal{M}(n))$ existe un axioma construido bajo el esquema (A5+j (I)), en este caso:

   2.1. Para toda descripción instantánea $\alpha_t = Pq_is_jQ$, con P y Q posiblemente vacíos, por definición de cambio de descripción instantánea y por ser $\mathcal{M}$ una MTD, se tiene que la descripción instantánea siguiente será obligatoriamente $\alpha_{t+1} = Pq_ls_kQ$.

   2.2. Por definición de Q_i^μ y S_j^μ en 2.1, se tiene que $(t, x) \in Q_i^\mu$ y $(t, x) \in S_j^\mu$ en $\mu(\mathcal{M}(n))$.

   2.3. Por definición de Q_i^μ y S_j^μ en 2.1, se tiene que $(t + 1, x) \in Q_l^\mu$ y $(t + 1, x) \in S_k^\mu$ en $\mu(\mathcal{M}(n))$.

   2.4. P y Q permanecen iguales, por lo tanto, por definición de S_j^μ en 2.1, se tiene que para todo $y \neq x$, si $(t, y) \in S_i^\mu$ entonces $(t + 1, y) \in S_i^\mu$ en $\mu(\mathcal{M}(n))$.

   2.5. Por definición de δ en 2.2, 2.3 y 2.4, se tiene que $\mu(\mathcal{M}(n)) \models$ (A5+j (I)).

3. Si $\mathcal{M}$ contiene una instrucción de la forma (II) entonces en $\Delta'_{LC}(\mathcal{M}(n))$ existe un axioma construido bajo el esquema (A5+j (II)), en este caso $\mu(\mathcal{M}(n)) \models$ (A5+j (II)). La demostración es similar a la realizada en el paso 2.

4. Si $\mathcal{M}$ contiene una instrucción de la forma (III) entonces en $\Delta'_{LC}(\mathcal{M}(n))$ existe un axioma construido bajo el esquema (A5+j (III)), en este caso $\mu(\mathcal{M}(n)) \models$ (A5+j (III)). La demostración es similar a la realizada en el paso 2.

5. Para el tiempo $t = 0$ la descripción instantánea de $\mathcal{M}(n)$ es $\alpha_0 = q_1S_{i,1}S_{i,2}\ldots S_{i,p}$. Por la definición de Q_i^μ y S_j^μ se tiene, en $\mu(\mathcal{M}(n))$, que $(0, 0) \in Q_1^\mu$, $(0, 0) \in S_{i,1}^\mu$, $(0, 1) \in S_{i,2}^\mu$, $\ldots$, $(0, p - 1) \in S_{i,p}^\mu$ y $(0, y) \in S_0^\mu$ para todo $y < 0$ ó $y > p - 1$. Por lo tanto, por definición de δ, se tiene que $\mu(\mathcal{M}(n)) \models$ (A5+$\overline{\overline{I}}$+1).

6. La situación inicial de $\mathcal{M}(n)$ es representada por la descripción instantánea α_0 (no existe descripción instantánea anterior). Por la definición de Q_i^μ y S_j^μ, para todo $t < 0$ y para todo x, se tiene que $(t,x) \notin Q_i^\mu$ (con $i = 1, 2, \ldots, n$) y $(t,x) \notin S_j^\mu$ (con $j = 0, 1, 2, \ldots, m$) en $\mu(\mathcal{M}(n))$. Por lo tanto, por definición de δ, se tiene que $\mu(\mathcal{M}(n)) \models$ (A5+$\overline{\overline{I}}$+2).

7. Como $\mathcal{M}$ es MTD, sólo existe, para la entrada n, una única secuencia $\alpha_0\alpha_1\alpha_2\ldots$ (posiblemente infinita) de descripciones instantáneas, tales que $\alpha_i \xrightarrow{\mathcal{M}} \alpha_{i+1}$ (para $i = 0, 1, 2\ldots$). Además, en cada α_t sólo un símbolo s_j se encuentra inmediatamente enseguida del símbolo q_i, en la posición x. Por definición de S_j^μ se tiene, en $\mu(\mathcal{M}(n))$, que $(t,x) \in S_j^\mu$ y que $(t,x) \notin S_k^\mu$ para todo $k \neq j$. Por lo tanto, por definición de δ, se tiene que $\mu(\mathcal{M}(n)) \models$ (A5+$\overline{\overline{I}}$+i+3).

8. Como $\mathcal{M}$ es MTD, sólo existe, para la entrada n, una única secuencia $\alpha_0\alpha_1\alpha_2\ldots$ (posiblemente infinita) de descripciones instantáneas, tales que $\alpha_i \xrightarrow{\mathcal{M}} \alpha_{i+1}$ (para $i = 0, 1, 2\ldots$). Además, en cada α_t existe exáctamente un símbolo q_i y $\mathcal{M}$ se encuentra en la posición x. Por definición de Q_i^μ se tiene, en $\mu(\mathcal{M}(n))$, que $(t,x) \in Q_i^\mu$ para algún $i = 1, 2, \ldots, n$ y para todas las demás posiciones $y \neq x$, se tiene que $(t,y) \notin Q_i^\mu$ para todo $i = 1, 2, \ldots, n$. Por lo tanto, por definición de δ, se tiene que $\mu(\mathcal{M}(n)) \models$ (A5+$\overline{\overline{I}}$+$\overline{\overline{\Sigma}}$+3).

9. Como $\mathcal{M}$ es MTD, sólo existe, para la entrada n, una única secuencia $\alpha_0\alpha_1\alpha_2\ldots$ (posiblemente infinita) de descripciones instantáneas, tales que $\alpha_i \xrightarrow{\mathcal{M}} \alpha_{i+1}$ (para $i = 0, 1, 2\ldots$). Además, en cada α_t existe exactamente un símbolo q_i y $\mathcal{M}$ se encuentra en la posición x. Por definición de Q_i^μ se tiene, en $\mu(\mathcal{M}(n))$, que $(t,x) \in Q_i^\mu$ y que $(t,x) \notin Q_k^\mu$ para todo $k \neq i$. Por lo tanto, por definición de δ, se tiene que $\mu(\mathcal{M}(n)) \models$ (A5+$\overline{\overline{I}}$+$\overline{\overline{\Sigma}}$+i+3).

10. Si la secuencia $\alpha_0\alpha_1\alpha_2\ldots\alpha_n$ de descripciones instantáneas, tales que $\alpha_i \xrightarrow{\mathcal{M}} \alpha_{i+1}$ (para $i = 0, 1, 2, \ldots, k$) es finita, entonces en α_k existe una subcadena q_is_j tal que $\mathcal{M}$ no contiene ninguna instrucción que comience por q_is_j. Por definición de Q_i^μ y S_j^μ se tiene, en $\mu(\mathcal{M}(n))$, que $(t,x) \in Q_i^\mu$, $(t,x) \in S_j^\mu$ y para todo tiempo $t > k$, para toda posición y, para todo $i = 1, 2, \ldots, n$ y para todo $j = 0, 1, \ldots, m$ se tiene que $(t,y) \notin Q_i^\mu$ y $(t,y) \notin S_j^\mu$. Por lo tanto, por definición de δ, se tiene que $\mu(\mathcal{M}(n)) \models$ (A5+$\overline{\overline{I}}$+$\overline{\overline{\Sigma}}$+$\overline{\overline{Q}}$+4).

□

Corolario 1.1. *Si $\mathcal{M}$ es MTD, para todo tiempo t y posición x, si $\Delta'_{LC}(\mathcal{M}(n)) \vdash Q_i(t,x)$ entonces $(t,x) \in Q_i^\mu$ en $\mu(\mathcal{M}(n))$.*

Corolario 1.2. *Si $\mathcal{M}$ es MTD, para todo tiempo t y posición x, si $\Delta'_{LC}(\mathcal{M}(n)) \vdash \neg Q_i(t,x)$ entonces $(t,x) \notin Q_i^\mu$ en $\mu(\mathcal{M}(n))$.*

Corolario 1.3. *Si $\mathcal{M}$ es MTD, para todo tiempo t y posición x, si $\Delta'_{LC}(\mathcal{M}(n)) \vdash S_j(t,x)$ entonces $(t,x) \in S_j^\mu$ en $\mu(\mathcal{M}(n))$.*

Corolario 1.4. *Si $\mathcal{M}$ es MTD, para todo tiempo t y posición x, si $\Delta'_{LC}(\mathcal{M}(n)) \vdash \neg S_j(t,x)$ entonces $(t,x) \notin S_j^\mu$ en $\mu(\mathcal{M}(n))$.*

Definición 1.5 (Fórmula asociada a una descripción instantánea[10]). *Sea $\alpha_t \equiv s_{j,1}\dots s_{j,n}q_i s_j \dots s_{j,p}$ la descripción instantánea de $\mathcal{M}(n)$ en el tiempo t, con $\mathcal{M}$ ubicada en la posición x. Utilizando para los valores negativos de p, para abreviar las fórmulas, las convenciones:*

$$\begin{aligned} Q_i(t,0^p) &\equiv Q_i(t,s) \wedge s^p = 0, \\ S_j(t,0^p) &\equiv S_j(t,s) \wedge s^p = 0, \\ y \neq 0^p &\equiv y \neq s \wedge s^p = 0; \end{aligned}$$

se define la fórmula:

$$\begin{aligned} D_t \equiv Q_i(t,0^x) \wedge S_{j,1}(t,0^{x-n}) \wedge \dots \wedge S_{j,n}(t,0^{x-1}) \wedge S_j(t,0^x) \wedge \dots \\ \wedge S_{j,p}(t,0^{(x+p)-(n+1)}) \wedge \forall y \Big(\big(y \neq 0^{x-n} \wedge \dots \wedge y \neq 0^{x-1} \\ \wedge y \neq 0^x \wedge \dots \wedge y \neq 0^{(x+p)-(n+1)} \big) \to S_0(t,y) \Big) \end{aligned} \qquad (D_t)$$

como la fórmula asociada a α_t.

Teorema 1.7. *Sea $\alpha_0\alpha_1\alpha_2\dots$ la secuencia (posiblemente infinita) de descripciones instantáneas de $\mathcal{M}(n)$, entonces $\Delta'_{LC}(\mathcal{M}(n)) \vdash D_t$, para $t = 0,1,2\dots$*

Demostración. Utilizando inducción matemática se tiene:

1. Paso base $t = 0$:

[10] Definición adaptada de [2, p. 117]

1.1. El axioma (A5+$\overline{\overline{\overline{I}}}$+1), el cual codifica la situación inicial de $\mathcal{M}(n)$, es precisamente la fórmula asociada a α_0.

2. Paso inductivo:

2.1. Sea $\alpha_t \equiv s_{j,1} \ldots s_{j,n} q_i s_j \ldots s_{j,p}$ la descripción instantánea de $\mathcal{M}(n)$ en el tiempo t, con $\mathcal{M}$ ubicada en la posición x.

2.2. Por HI se tiene que $\Delta'_{LC}(\mathcal{M}(n)) \vdash D_t$.

2.3. Si α_t no es la última en la secuencia ($\mathcal{M}(n)$ no se ha detenido en el tiempo t), entonces existe en $\mathcal{M}$ una instrucción que provoca el cambio de descripción instantánea $\alpha_t \xrightarrow{\mathcal{M}} \alpha_{t+1}$.

2.4. Si la instucción que provoca el cambio de descripción instantánea $\alpha_t \xrightarrow{\mathcal{M}} \alpha_{t+1}$ es de la forma (I):

2.4.1. Por 2.1, 2.4 y definición de cambio de descripción instantánea se tiene que $\alpha_{t+1} \equiv s_{j,1} \ldots s_{j,n} q_l s_k \ldots s_{j,p}$.

2.4.2. Por 2.2 y SIMP se tiene que $\Delta'_{LC}(\mathcal{M}(n)) \vdash Q_i(t, 0^x) \wedge S_j(t, 0^x)$.

2.4.3. Por 2.4 y definición de $\Delta'_{LC}(\mathcal{M}(n))$, se tiene que en $\Delta'_{LC}(\mathcal{M}(n))$ existe un axioma construido bajo el esquema (A5+j (I)).

2.4.4. Aplicando IU($t/t, x/0^x$) en (A5+j (I)), MP con (2.4.2) y SIMP se tiene que $\Delta'_{LC}(\mathcal{M}(n)) \vdash Q_l(t', 0^x) \wedge S_k(t', 0^x)$.

2.4.5. Aplicando IU($t/t, x/0^x$) en (A5+j (I)), se tiene que $\forall y\Big((y \neq 0^x) \rightarrow \Big(\bigwedge_{i=0}^{m}\big(S_i(t, y) \rightarrow S_i(t', y)\big)\Big)\Big)$.

2.4.6. Aplicando SIMP en 2.2 para cada uno de los símbolos $S_{j,i}(t, 0^y)$, luego IU($t/t, y/0^y$) en 2.4.5, MP y CONJ se obtiene $S_{j,1}(t', 0^{x-n}) \wedge \cdots \wedge S_{j,n}(t', 0^{x-1}) \wedge S_{j,n+2}(t', 0^{x+1}) \wedge \cdots \wedge S_{j,p}(t', 0^{(x+p)-(n+1)})$.

2.4.7. Por SIMP en 2.2 y 2.4.5 se obtiene $\forall y\Big(\big(y \neq 0^{x-n} \wedge \cdots \wedge y \neq 0^{x-1} \wedge y \neq 0^x \wedge \cdots \wedge y \neq 0^{(x+p)-(n+1)}\big) \rightarrow S_0(t', y)\Big)$.

2.4.8. Por CONJ de 2.4.4, 2.4.6 y 2.4.7, se tiene que $\Delta'_{LC}(\mathcal{M}(n)) \vdash D_{t+1}$.

2.5. Para los cambios de descripción instantánea provocados por instrucciones de la forma (II) y (III), la demostración es similar a la del paso 2.4.

3. Por IM queda demostrado el teorema para $t = 0, 1, 2 \ldots$

$\square$

Teorema 1.8. *Si $\mathcal{M}$ es MTD, para todo tiempo t y posición x, si $(t, x) \in Q_i^\mu$ en $\mu(\mathcal{M}(n))$ entonces $\Delta'_{LC}(\mathcal{M}(n)) \vdash Q_i(t, x)$.*

Demostración.

1.	$(t, x) \in Q_i^\mu$ en $\mu(\mathcal{M}(n))$.	SUP.
2.	q_i está en la descripción instantánea α_t de $\mathcal{M}(n)$ y $\mathcal{M}$ se encuentra en la posición x.	Definición de Q_i^μ en 1.
3.	$\Delta'_{LC}(\mathcal{M}(n)) \vdash D_t$.	Teorema 1.7 en 2.
4.	$\Delta'_{LC}(\mathcal{M}(n)) \vdash Q_i(t, x)$.	SIMP en 3.

$\square$

Teorema 1.9. *Si $\mathcal{M}$ es MTD, para todo tiempo t y posición x, si $(t, x) \notin Q_i^\mu$ en $\mu(\mathcal{M}(n))$ entonces $\Delta'_{LC}(\mathcal{M}(n)) \vdash \neg Q_i(t, x)$.*

Demostración.

1. $(t, x) \notin Q_i^\mu$ en $\mu(\mathcal{M}(n))$. SUP.
2. Si $t < 0$ ó $t > td$ (si existe td) entonces, por los esquemas (A5+$\overline{\overline{I}}$+2) y (A5+$\overline{\overline{I}}$+$\overline{\overline{\Sigma}}$+$\overline{\overline{Q}}$+4), con IU$(t/t, x/x)$, MP y SIMP se obtiene para ambos casos que $\Delta'_{LC}(\mathcal{M}(n)) \vdash \neg Q_i(t, x)$.
3. Si $0 \leq t \leq td$ (si existe td), por definición de Q_i^μ en 1, se tiene que la descripción instantánea $\alpha_t \equiv Pq_j s_l Q$, donde $q_i \neq q_j$ o que $\mathcal{M}$ no se encuentra en la posición x.
4. Si $\alpha_t \equiv Pq_j s_l Q$, donde $q_i \neq q_j$, por teorema 1.7 se tiene que $\Delta'_{LC}(\mathcal{M}(n)) \vdash D_t$ y por SIMP se tiene que $\Delta'_{LC}(\mathcal{M}(n)) \vdash Q_j(t, x)$. Por lo tanto, aplicando IU$(t/t, x/x)$ en (A5+$\overline{\overline{I}}$+$\overline{\overline{\Sigma}}$+i+3), MP y SIMP se tiene que $\Delta'_{LC}(\mathcal{M}(n)) \vdash \neg Q_i(t, x)$.
5. Si $\mathcal{M}$ no está en la posición x entonces debe estar en otra posición z $(z \neq x)$. En este caso, por teorema 1.7 se tiene que $\Delta'_{LC}(\mathcal{M}(n)) \vdash D_t$ y por SIMP se tiene que $\Delta'_{LC}(\mathcal{M}(n)) \vdash Q_j(t, z)$. Por lo tanto, aplicando IU$(t/t, x/z, y/x)$ en (A5+$\overline{\overline{I}}$+$\overline{\overline{\overline{\Sigma}}}$+3), MP y SIMP se tiene que $\Delta'_{LC}(\mathcal{M}(n)) \vdash \neg Q_i(t, x)$.

□

Teorema 1.10. *Si $\mathcal{M}$ es MTD, para todo tiempo t y posición x, si $(t,x) \in S_j^{\mu}$ en $\mu(\mathcal{M}(n))$ entonces $\Delta'_{LC}(\mathcal{M}(n)) \vdash S_j(t,x)$.*

Demostración. La demostración es similar a la del teorema 1.8. □

Teorema 1.11. *Si $\mathcal{M}$ es MTD, para todo tiempo t y posición x, si $(t,x) \notin S_i^{\mu}$ en $\mu(\mathcal{M}(n))$ entonces $\Delta'_{LC}(\mathcal{M}(n)) \vdash \neg S_j(t,x)$.*

Demostración. La demostración es similar a la del teorema 1.9. □

Capítulo 2

Axiomatización de máquinas de Turing no deterministas

Al utilizar el método de axiomatización definido en el capitulo anterior para máquinas de Turing no deterministas, en algunos casos se obtienen teorías contradictorias (como lo demuestra el ejemplo 2.1) y por lo tanto triviales, teniendo en cuenta la lógica subyacente. Para solucionar dicho problema, se hacen algunas modificaciones al método de axiomatización para obtener teorías $\Delta''_{LC}(\mathcal{M}(n))$ que den cuenta del comportamiento de máquinas de Turing no deterministas.

Ejemplo 2.1. *Sea* $\mathcal{M}_3 =< Q, \Sigma, M, I >$ *la MTND que se obtiene al adicionar a la MTD* $\mathcal{M}_2$ *del ejemplo 1.2 una nueva instrucción* $i_3 = q_2 s_0 s_0 q_1$ *(Figura 2.1), y sea* $n_3 = n_2$ *(ejemplo 1.2) la secuencia de entrada para* $\mathcal{M}_3$*.*

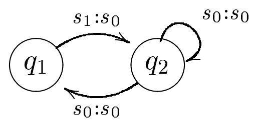

Figura 2.1: Grafo máquina $\mathcal{M}_3$

Utilizando el método de axiomatización definido en el capítulo 1, se tiene *que* $\Delta'_{LC}(\mathcal{M}_3(n_3)) = \Delta'_{LC}(\mathcal{M}_2(n_2)) \cup \{(\text{A16})\}$, *donde:*

$$\forall t \forall x \Bigg(\Big(Q_2(t,x) \wedge S_0(t,x) \Big) \rightarrow \Big(Q_1(t',x) \wedge S_0(t',x) \wedge \\ \forall y \Big((y \neq x) \rightarrow \big((S_0(t,y) \rightarrow S_0(t',y)) \wedge (S_1(t,y) \rightarrow S_1(t',y)) \big) \Big) \Big) \Bigg). \quad \text{(A16)}$$

Para $\mathcal{M}_3$, *la fórmula* H *que indica que la máquina se detiene es:*

$$H \equiv \exists t \exists x (Q_1(t,x) \wedge S_0(t,x)). \quad \text{(H)}$$

Con $\Delta'_{LC}(\mathcal{M}_3(n_3))$ *se puede demostrar* H *(teorema 2.1) y también* $\neg H$ *(teorema 2.2). Por lo tanto* $\Delta'_{LC}(\mathcal{M}_3(n_3))$ *es inconsistente*[1] *y trivial, teniendo en cuenta la lógica subyacente.*

Teorema 2.1. $\Delta'_{LC}(\mathcal{M}_3(n_3)) \vdash H$.

Demostración.

1.	$Q_1(0,0) \wedge S_1(0,0)$.	SIMP en A8 (ejemplo 1.2).
2.	$Q_2(0',0) \wedge S_0(0',0)$.	IU$(t/0, x/0)$ en A6 (ejemplo 1.2), MP con 1 y SIMP.
3.	$Q_1(0'',0) \wedge S_0(0'',0)$.	IU$(t/0', x/0)$ en A16, MP con 2 y SIMP.
4.	H.	GE en 3.

□

Teorema 2.2. $\Delta'_{LC}(\mathcal{M}_3(n_3)) \vdash \neg H$.

Demostración. La demostración de este teorema es similar a la del teorema 1.5. □

Las inconsistencias en el ejemplo 2.1 surgen debido a que en la teoría $\Delta'_{LC}(\mathcal{M}_3(n_3))$, no se establece la condición de que sólo una de las instrucciones (i_2, i_3) sea ejecutada cuando la máquina se encuentre en el estado q_2 leyendo el símbolo s_0. Reemplazando los consecuentes de los axiomas (A7)

[1] Con $\Delta'_{LC}(\mathcal{M}_3(n_3))$ también se pueden demostrar muchas otras fórmulas contradictorias como por ejemplo $Q_2(0',0)$ y $\neg Q_2(0',0)$.

(ejemplo 1.2) y (A16) por α y β respectivamente, con el objetivo de abreviar las fórmulas. Se obtiene:

$$\forall t \forall x ((Q_2(t,x) \wedge S_0(t,x)) \to (\alpha)), \tag{A7'}$$

$$\forall t \forall x ((Q_2(t,x) \wedge S_0(t,x)) \to (\beta)). \tag{A16'}$$

De donde se deduce:

$$\forall t \forall x ((Q_2(t,x) \wedge S_0(t,x)) \to (\alpha \wedge \beta)), \tag{A7'-A16'}$$

que interpretado, bajo la interpretación intencional $\mathcal{I}$, indica que la máquina ejecuta ambas intrucciones.

Definición 2.1 (Teoría $\Delta''_{LC}(\mathcal{M}(n))$). *Para construir teorías $\Delta''_{LC}(\mathcal{M}(n))$ en las que se establezca la condición de que la máquina de Turing $\mathcal{M}$, al encontrarse en una situación en la cual puede ejecutar más de una instrucción, sólo ejecuta una de dichas intrucciones, se debe modificar en $\Delta'_{LC}(\mathcal{M}(n))$ (definición 1.3) la forma de construir los axiomas para las instrucciones, incluyendo sólo un axioma en $\Delta''_{LC}(\mathcal{M}(n))$ para las instrucciones que comienzan con los mismos dos símbolos iniciales. Sean $i_1, i_2, \ldots, i_n$ las instrucciones cuyos dos símbolos iniciales son los mismos (q_i, s_j). Para estas instrucciones se debe adicionar en $\Delta''_{LC}(\mathcal{M}(n))$ el axioma:*

$$\forall t \forall x \left((Q_i(t,x) \wedge S_j(t,x)) \to \left(\left(\bigvee_{k=1}^{n} \alpha_k \right) \wedge \left(\bigwedge_{l \neq m} \neg(\alpha_l \wedge \alpha_m) \right) \right) \right), \tag{A5+j-5+j+n}$$

donde α_j representa el consecuente del axioma correspondiente a la instrucción i_j, de acuerdo con la definición 1.2.

Ejemplo 2.2. *Para el ejemplo* (2.1) *se tiene que el axioma correspondiente a las instrucciones i_2, i_3 es:*

$$\forall t \forall x \Big(\big(Q_2(t,x) \wedge S_0(t,x)\big) \to \big((\alpha \vee \beta) \wedge \neg(\alpha \wedge \beta)\big) \Big). \tag{A7-16}$$

Por lo tanto, $\Delta''_{LC}(\mathcal{M}_3(n_3))$ es el resultado de reemplazar los axiomas A7 y A16 en $\Delta'_{LC}(\mathcal{M}_3(n_3))$ por el axioma A7-16.

Básicamente existen dos modelos de MTND, los cuales se distinguen de acuerdo con el comportamiento que la máquina tiene en el momento en que se encuentra en una 'configuración ambigua'[2]: las máquinas de Turing randómicas (MTR), las cuales eligen la instrucción a ejecutar de forma randómica (aleatoria) y las máquinas de Turing probabilistas (MTP), en las cuales se asigna una probabilidad a las instrucciones que entran en conflicto, por lo tanto, no tienen el mismo chance de ser ejecutadas [20, p. 50]. En la construcción de la teoría $\Delta''_{LC}(\mathcal{M}(n))$, no se esquematiza ninguna regla que controle el comportamiento de la máquina en el momento de hacer una elección de la instrucción a ejecutar, es decir, se dejan estas reglas como algo externo a la teoría. Lo que concuerda, en cierto sentido, con la necesidad de que un operador externo realice dicha elección, como lo expresa Turing en [26, p. 232].

Para cada computación $c_i \equiv \alpha_0\alpha_1\alpha_2\ldots$ (posiblemente infinita) de $\mathcal{M}(n)$, donde $\mathcal{M}$ es MTND, se puede construir una estructura $\mu(\mathcal{M}(n))$ (definición 1.4) relativa a c_i, la cual se denotará por $\mu(\mathcal{M}(n))_{c_i}$. También se puede definir una teoría $\Delta''_{LC}(\mathcal{M}(n))$ relativa a c_i, la cual se denotará por $\Delta''_{LC}(\mathcal{M}(n))_{c_i}$, que se construye adicionando a $\Delta''_{LC}(\mathcal{M}(n))$ la instanciación de los consecuentes de las instrucciones elegidas en los pasos de computación de c_i donde hubo necesidad de hacer elecciones. Se ilustra esto con un ejemplo:

Ejemplo 2.3. *Sea $\mathcal{M}_3(n_3)$ la MTND y la secuencia de entrada definidas en el ejemplo 2.1 y sea $c_1 \equiv \alpha_0\alpha_1\alpha_2\alpha_3$ una posible computación de $\mathcal{M}_3(n_3)$ (figura 2.2), donde $\alpha_0 \equiv q_1s_1$, $\alpha_1 \equiv s_0q_2s_0$, $\alpha_2 \equiv \alpha_1$ (se eligió la instrucción i_2) y $\alpha_3 \equiv s_0q_1s_0$ (se eligió la instrucción i_3). Entonces $\Delta''_{LC}(\mathcal{M}(n))_{c_1} = \Delta''_{LC}(\mathcal{M}(n)) \cup \{E_1, E_2\}$, donde:*

$$\begin{aligned}E_1 \equiv\ & Q_2(2,1) \wedge S_0(2,1) \wedge \\ & \forall y\Big((y \neq 1) \rightarrow \Big(\big(S_0(1,y) \rightarrow S_0(2,y)\big) \wedge \big(S_1(1,y) \rightarrow S_1(2,y)\big)\Big)\Big),\end{aligned}$$

$$\begin{aligned}E_2 \equiv\ & Q_1(3,1) \wedge S_0(3,1) \wedge \\ & \forall y\Big((y \neq 1) \rightarrow \Big(\big(S_0(2,y) \rightarrow S_0(3,y)\big) \wedge \big(S_1(2,y) \rightarrow S_1(3,y)\big)\Big)\Big).\end{aligned}$$

Bajo las definiciones anteriores, para la teoría $\Delta''_{LC}(\mathcal{M}(n))$ se establecen teoremas equivalentes a los establecidos para $\Delta'_{LC}(\mathcal{M}(n))$ en el capítulo anterior:

[2]Término utilizado por Turing, en su artículo fundacional [26, p. 232], para referirse a que la máquina se encuentra en una situación en la cual tiene varias instrucciones posibles a ser ejecutadas.

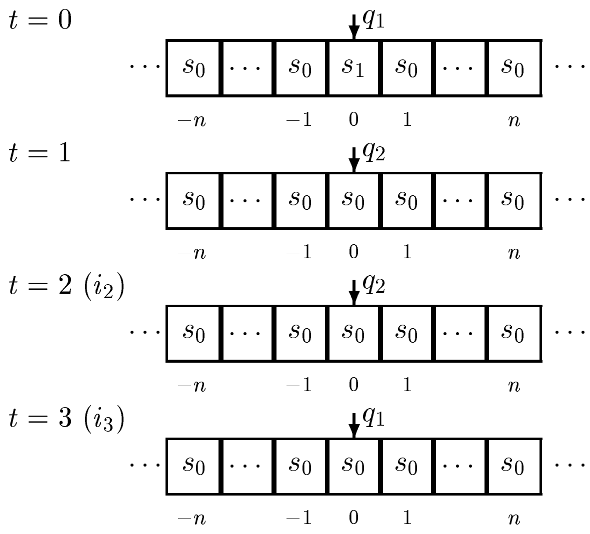

Figura 2.2: Computación c_i

Teorema 2.3. *Para toda computación posible c_i de $\mathcal{M}(n)$, donde $\mathcal{M}$ es MTND, se tiene que $\mu(\mathcal{M}(n))_{c_i}$ es modelo de $\Delta''_{LC}(\mathcal{M}(n))_{c_i}$.*

Corolario 2.1. *Para toda computación posible c_i de $\mathcal{M}(n)$, donde $\mathcal{M}$ es MTND, para todo tiempo t y posición x, si $\Delta'_{LC}(\mathcal{M}(n))_{c_i} \vdash Q_i(t,x)$ entonces $(t,x) \in Q_i^{\mu}$ en $\mu(\mathcal{M}(n))_{c_i}$.*

Corolario 2.2. *Para toda computación posible c_i de $\mathcal{M}(n)$, donde $\mathcal{M}$ es MTND, para todo tiempo t y posición x, si $\Delta'_{LC}(\mathcal{M}(n))_{c_i} \vdash \neg Q_i(t,x)$ entonces $(t,x) \notin Q_i^{\mu}$ en $\mu(\mathcal{M}(n))_{c_i}$.*

Corolario 2.3. *Para toda computación posible c_i de $\mathcal{M}(n)$, donde $\mathcal{M}$ es MTND, para todo tiempo t y posición x, si $\Delta'_{LC}(\mathcal{M}(n))_{c_i} \vdash S_j(t,x)$ entonces $(t,x) \in S_j^{\mu}$ en $\mu(\mathcal{M}(n))_{c_i}$.*

Corolario 2.4. *Para toda computación posible c_i de $\mathcal{M}(n)$, donde $\mathcal{M}$ es MTND, para todo tiempo t y posición x, si $\Delta'_{LC}(\mathcal{M}(n))_{c_i} \vdash \neg S_j(t,x)$ entonces $(t,x) \notin S_j^{\mu}$ en $\mu(\mathcal{M}(n))_{c_i}$.*

Teorema 2.4. *Para toda computación posible c_i de $\mathcal{M}(n)$, donde $\mathcal{M}$ es MTND, para todo tiempo t y posición x, si $(t,x) \in Q_i^{\mu}$ en $\mu(\mathcal{M}(n))_{c_i}$ entonces $\Delta'_{LC}(\mathcal{M}(n))_{c_i} \vdash Q_i(t,x)$.*

Teorema 2.5. *Para toda computación posible c_i de $\mathcal{M}(n)$, donde $\mathcal{M}$ es MTND, para todo tiempo t y posición x, si $(t,x) \notin Q_i^\mu$ en $\mu(\mathcal{M}(n))_{c_i}$ entonces $\Delta'_{LC}(\mathcal{M}(n))_{c_i} \vdash \neg Q_i(t,x)$.*

Teorema 2.6. *Para toda computación posible c_i de $\mathcal{M}(n)$, donde $\mathcal{M}$ es MTND, para todo tiempo t y posición x, si $(t,x) \in S_j^\mu$ en $\mu(\mathcal{M}(n))_{c_i}$ entonces $\Delta'_{LC}(\mathcal{M}(n))_{c_i} \vdash S_j(t,x)$.*

Teorema 2.7. *Para toda computación posible c_i de $\mathcal{M}(n)$, donde $\mathcal{M}$ es MTND, para todo tiempo t y posición x, si $(t,x) \notin S_i^\mu$ en $\mu(\mathcal{M}(n))_{c_i}$ entonces $\Delta'_{LC}(\mathcal{M}(n))_{c_i} \vdash \neg S_j(t,x)$.*

Capítulo 3

Teorías paraconsistentes de máquinas de Turing

En el contexto de la discusión de las relaciones entre sistematización cognocitiva y lógica, Nicolas Rescher considera que "la consistencia es un requisito muy importante de la sistematización cognocitiva, aunque entendida no tanto como 'principio constitutivo' a nivel de la descripción ontológica del mundo, sino como un 'principio regulativo' a nivel epistemológico" [19, p. 379]. Afirma que "tolerar inconsistencias dentro de la esfera de la sistematización racional no sólo es permisible, sino que en situaciones apropiadas puede resultar ventajoso e incluso inevitable" [19, p. 381]. Entiende las inconsistencias como una 'sobredeterminación de información', que puede surgir por diferentes causas como lo puede ser la existencia de diferentes fuentes de información, diferentes recopilaciones de la misma información o porque la información hace parte de 'masas' diferentes de información, y "plantea que si bien las inconsistencias no son un resultado deseable, son diferentes al error, y no siempre tiene sentido aplicar todos los mecanismos para lograr evitarlas, pues esta actitud puede llevar a perder información valiosa" [19, p. 384].

Los planteamientos de Rescher abren un camino para construir lo que puede ser una teoría paraconsistente de máquinas de Turing. Las instrucciones 'ambiguas' pueden ser pensadas como 'sobredeterminaciones de información', la máquina cuenta con múltiples acciones a ser realizadas en una situación determinada, acciones que pueden dar lugar a contradicciones, que como se verá en la sección 3.1, se presentan en cuanto a los símbolos en la cinta, estados y posición de la máquina en un instante de tiempo determinado. Estas contradicciones son debidas a la existencia de instrucciones 'ambiguas' y a las exigencias de unicidad establecidas sobre los elementos mencionados. Puede entonces pensarse en la unicidad de símbolos en las casillas y en la unicidad de estado y posición de la máquina, en cada instante de tiempo, como situaciones deseables o 'principios regulativos' dentro del proceso de computación de la máquina, que sin embargo, pueden ser violados en ciertos instantes de tiempo dentro del proceso de computación, instantes en los que la máquina se encuentra en estado de 'inconsistencia' o 'sobredeterminación', que deberán ser superados en instantes posteriores de la computación.

Para definir entonces un modelo de máquina de Turing paraconsistente, inicialmente, se analizarán algunas consecuencias de sustituir la lógica subyacente a $\Delta'_{LC}(\mathcal{M}(n))$ (definición 1.3) por la lógica $C_1^=$ (sección C.0.1), y a partir de estas consecuencias se redefinirá lo que es una computación en una máquina de Turing.

Se ha elegido la lógica $C_1^=$ por tratarse de una lógica paraconsistente a nivel de predicados de primer orden con igualdad, que cuenta con algunas propiedades que harán que el cambio de lógica preserve muchas características del comportamiento de la máquina. En principio, se puede realizar un trabajo similar con las lógicas de predicados de primer orden de los *C-Systems*[1] o con cualquier otra lógica de predicados de primer orden paraconsistente[2].

Antes de sustituir la lógica en $\Delta'_{LC}(\mathcal{M}(n))$ por $C_1^=$, se presenta un análisis de las posibles contradicciones que pueden surgir en $\Delta'_{LC}(\mathcal{M}(n))$ al axiomatizar MTNDs.

[1]En [5] Walter Carnielli y João Marcos presentan una jerarquía de lógicas proposicionales paraconsistentes, en las cuales internalizan la noción de 'consistencia', definiendola por medio de un operador y los axiomas correspondientes en el lenguaje objeto. A estas lógicas las denominan *logics of formal inconsistency* (LFIs). Los *C-Systems* los definen como LFIs que se obtienen de extender las bases positivas de lógicas consistentes. A partir de la lógica positiva clásica construyen una jerarquía de *C-Systems* buscando que el operador de 'consistencia' cumpla con ciertas propiedades. El cálculo C_n de da Costa resulta ser un caso particular dentro de la jerarquía.

En [6] Walter Carnielli, João Marcos y Sandra de Amo definen dos systemas de lógica de predicados de primer orden para LFIs (LFI1* y LFI2*).

En [3], Walter Carnielli, Marcelo Coniglio y João Marcos presentan los *C-Systems*, las extensiones al cálculo de predicados de primer orden y las semánticas asociadas.

[2]Aunque puede resultar complicado realizar este procedimiento con lógicas paraconsistentes que no preserven, por lo menos, ciertas reglas de la lógica positiva clásica.

3.1. Posibles contradicciones que surgen en $\Delta'_{LC}(\mathcal{M}(n))$ al axiomatizar MTNDs

Teniendo en cuenta las tres posibles formas de instrucciones con las que puede contar una máquina de Turing (definición A.1), las posibles inconsistencias que pueden surgir en $\Delta'_{LC}(\mathcal{M}(n))$ de acuerdo con las diferentes combinaciones de instrucciones 'ambiguas' son:

1. Las dos instrucciones 'ambiguas' son de la forma (I): en este caso, la teoría $\Delta'_{LC}(\mathcal{M}(n))$ cuenta con dos axiomas construidos bajo el esquema (A5+j (I)). Como en LC, por TD, SIMP y CONJ, es válido pasar de $\alpha \to (\beta \wedge \gamma)$ a $(\alpha \to \beta) \wedge (\alpha \to \gamma)$, aplicando IU$(t/t, x/x)$, la regla anterior y SIMP en (A5+j (I)) se obtiene:

$$\big(Q_i(t,x) \wedge S_j(t,x)\big) \to \big(Q_l(t',x) \wedge S_k(t',x)\big), \tag{3.1}$$
$$\big(Q_i(t,x) \wedge S_j(t,x)\big) \to \big(Q_n(t',x) \wedge S_m(t',x)\big). \tag{3.2}$$

Además, como en $\Delta'_{LC}(\mathcal{M}(n))$, para cada símbolo $s_j \in \Sigma$, existe un axioma construido bajo el esquema (A5+$\overline{\overline{I}}$+i+3), en el caso en que $k \neq m$, aplicando IU$(t/t', x/x)$ en (A5+$\overline{\overline{I}}$+i+3) se obtiene:

$$S_k(t',x) \to \neg S_m(t',x), \tag{3.3}$$
$$S_m(t',x) \to \neg S_k(t',x). \tag{3.4}$$

De forma similar, como en $\Delta'_{LC}(\mathcal{M}(n))$, para cada símbolo $q_i \in Q$, existe un axioma construido bajo el esquema (A5+$\overline{\overline{I}}$+$\overline{\overline{\Sigma}}$+i+3), en el caso en que $l \neq n$, aplicando IU$(t/t', x/x)$ en (A5+$\overline{\overline{I}}$+$\overline{\overline{\Sigma}}$+i+3) se obtiene:

$$Q_l(t',x) \to \neg Q_n(t',x), \tag{3.5}$$
$$Q_n(t',x) \to \neg Q_l(t',x). \tag{3.6}$$

De (3.1), (3.2), (3.3) y (3.4), utilizando TD, MP, SIMP y CONJ se obtiene:

$$\begin{aligned}&\big(Q_i(t,x) \wedge S_j(t,x)\big) \to \\ &\qquad \Big(\big(S_k(t',x) \wedge \neg S_k(t',x)\big) \wedge \big(S_m(t',x) \wedge \neg S_m(t',x)\big)\Big),\end{aligned} \tag{3.7}$$

lo que, bajo la interpretación intensional $\mathcal{I}$ (sección 1.1), indica que si $\mathcal{M}(n)$, en cualquier instante de tiempo t y posición x, llega a estar en el estado q_i leyendo el símbolo s_j, entonces, en el siguiente instante de tiempo t', en la posición x se presentarán inconsistencias con respecto al símbolo que esta casilla contiene. En $\mathcal{M}(n)$ los símbolos s_k y s_m estarán y no estarán en la casilla x en el tiempo t'.

De forma similar, de (3.1), (3.2), (3.5) y (3.6), utilizando TD, MP, SIMP y CONJ se obtiene:

$$\begin{aligned}&\big(Q_i(t,x) \wedge S_j(t,x)\big) \to \\ &\qquad\Big(\big(Q_l(t',x) \wedge \neg Q_l(t',x)\big) \wedge \big(Q_n(t',x) \wedge \neg Q_n(t',x)\big)\Big), \qquad (3.8)\end{aligned}$$

lo que, bajo la interpretación intensional $\mathcal{I}$, indica que si $\mathcal{M}(n)$, en cualquier instante de tiempo t y posición x, llega a estar en el estado q_i leyendo el símbolo s_j, entonces, en el siguiente instante de tiempo t', en la posición x se presentarán inconsistencias con respecto al estado en el que $\mathcal{M}(n)$ se encuentra. La máquina $\mathcal{M}(n)$ estará y no estará en el estado q_l y q_n en el tiempo t'.

2. Una de las instrucciones 'ambiguas' es de la forma (I) y la otra es de la forma (II): en este caso, de forma similar al caso 1, se puede obtener:

$$\begin{aligned}&\big(Q_i(t,x) \wedge S_j(t,x)\big) \to \\ &\qquad\Big(\big(S_k(t',x) \wedge \neg S_k(t',x)\big) \wedge \big(S_j(t',x) \wedge \neg S_j(t',x)\big)\Big),\end{aligned}$$

que es similar y tiene la misma interpretación de (3.7). Además, se puede obtener:

$$\begin{aligned}&\big(Q_i(t,x) \wedge S_j(t,x)\big) \to \\ &\qquad\Big(\big(Q_l(t',x) \wedge \neg Q_l(t',x)\big) \wedge \big(Q_m(t',x') \wedge \neg Q_m(t',x')\big)\Big),\end{aligned}$$

que se diferencia de (3.8) en cuanto, en este caso, las inconsistencias en los estados se presentan en posiciones diferentes de la cinta (x y x').

3. Una de las instrucciones 'ambiguas' es de la forma (I) y la otra es de la forma (III): este caso es similar al caso 2.

4. Las dos instrucciones 'ambiguas' son de la forma (II): en este caso sólo se puede obtener:

$$\begin{aligned}&\big(Q_i(t,x) \wedge S_j(t,x)\big) \\ &\qquad\to \Big(\big(Q_l(t',x') \wedge \neg Q_l(t',x')\big) \wedge \big(Q_n(t',x') \wedge \neg Q_n(t',x')\big)\Big),\end{aligned}$$

que tiene una interpretación similar a la de (3.8), con la diferencia de que las inconsistencias se presentan en la posición x' y no en la posición x.

5. Una de las instrucciones 'ambiguas' es de la forma (II) y la otra es de la forma (III): en este caso, se hace necesario recurrir a la convención:

$$Q_m(t', x^{-1}) \equiv Q_m(t', y) \wedge y' = x,$$

para obtener:

$$\begin{aligned}&\big(Q_i(t,x) \wedge S_j(t,x)\big) \to \\ &\qquad \Big(\big(Q_l(t',x') \wedge \neg Q_l(t',x')\big) \wedge \big(Q_m(t',x^{-1}) \wedge \neg Q_m(t',x^{-1})\big)\Big),\end{aligned}$$

que tiene una interpretación similar a (3.8), con la diferencia de que las inconsistencias se presentan en las posiciones x^{-1} y x' y no en la posición x.

6. Las dos instrucciones 'ambiguas' son de la forma (III): este caso es similar al caso 4.

En resumen, al axiomatizarse MTNDs por medio de teorías $\Delta'_{LC}(\mathcal{M}(n))$ se pueden generar inconsistencias en cuanto al símbolo presente en una casilla e instante de tiempo determinado, y en cuanto al estado y posición de la máquina en un instante de tiempo determinado. La tabla 3.1 resume las inconsistencias generadas de acuerdo con las diferentes combinaciones de instrucciones 'ambiguas'.

Instrucciones 'ambiguas'	Contradicciones
Dos instrucciones de la forma (I)	$S_k(t',x) \wedge \neg S_k(t',x), S_m(t',x) \wedge \neg S_m(t',x),$ $Q_l(t',x) \wedge \neg Q_l(t',x), Q_n(t',x) \wedge \neg Q_n(t',x).$
Una instrucción de la forma (I) y una instrucción de la forma (II)	$S_k(t',x) \wedge \neg S_k(t',x), S_j(t',x) \wedge \neg S_j(t',x),$ $Q_l(t',x) \wedge \neg Q_l(t',x), Q_m(t',x') \wedge \neg Q_m(t',x').$
Una instrucción de la forma (I) y una instrucción de la forma (III)	$S_k(t',x) \wedge \neg S_k(t',x), S_j(t',x) \wedge \neg S_j(t',x),$ $Q_l(t',x) \wedge \neg Q_l(t',x), Q_m(t',x^{-1}) \wedge \neg Q_m(t',x^{-1}).$
Dos instrucciones de la forma (II)	$Q_l(t',x') \wedge \neg Q_l(t',x'), Q_n(t',x') \wedge \neg Q_n(t',x').$
Una instrucción de la forma (II) y una instrucción de la forma (III)	$Q_l(t',x') \wedge \neg Q_l(t',x'), Q_m(t',x^{-1}) \wedge \neg Q_m(t',x^{-1}).$
Dos instrucciones de la forma (III)	$Q_l(t',x^{-1}) \wedge \neg Q_l(t',x^{-1}), Q_n(t',x^{-1}) \wedge \neg Q_n(t',x^{-1}).$

Tabla 3.1: Contradicciones en $\Delta'_{LC}(\mathcal{M}(n))$ por instrucciones 'ambiguas'

3.2. Sustitución de la lógica subyacente a $\Delta'_{LC}(\mathcal{M}(n))$ por una lógica paraconsistente

Sea $\Delta'_{C_1^=}(\mathcal{M}(n))$ el resultado de sustituir la lógica subyacente a $\Delta'_{LC}(\mathcal{M}(n))$ por la lógica $C_1^=$, sin modificar ninguno de los axiomas. Como primer consecuencia se tiene que las posibles contradicciones analizadas en la sección anterior también son demostrables en $\Delta'_{C_1^=}(\mathcal{M}(n))$, puesto que todos los axiomas y reglas de inferencia utilizados o bien son instanciaciones universales (IU) o son pertenecientes al cálculo proposicional positivo clásico, por lo tanto, también son válidos en $C_1^=$. Sin embargo, como en ninguna parte se exige el buen comportamiento de $S_j(t,x)$ ni de $Q_i(t,x)$, la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$, a diferencia de la teoría $\Delta'_{LC}(\mathcal{M}(n))$, no se trivializa ante el surgimiento de estas contradicciones, permitiendo seguir el análisis del comportamiento de $\mathcal{M}(n)$ y definir una nueva noción de computación.

Para entender el proceso de computación en $\Delta'_{C_1^=}(\mathcal{M}(n))$ se presentan algunos ejemplos, en los cuales se deducen las descripciones instantáneas D_t, para $t = 0, 0', 0'', \ldots$, tomando los hechos positivos, con respecto a símbolos y estados, demostrables en $\Delta'_{C_1^=}(\mathcal{M}(n))$ para cada tiempo t. Mostrando de esta manera como evoluciona $\mathcal{M}(n)$ bajo la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$.

Elegir sólo los hechos positivos para formar la fórmula asociada a la descripción instantánea sigue, de cierta forma, el argumento de Jean Lukasiewicz en el cual plantea que "al hablarse de contradicciones, hay que tener muy presente que estas no se pueden 'percibir' en la realidad, porque no es posible percibir la negación que le es inherente, pues toda percepción es de hechos dados, es decir de hechos afirmativos, y sólo a partir de ellos se llega a aseverar negaciones, y de ahí se pueden llegar a inferir contradicciones" [19, p. 20].

Los múltiples ejemplos pretenden ilustrar diferentes situaciones que se pueden presentar en el proceso de computación, bajo la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$, conviviendo con las contradicciones en la teoría y aceptando la simultaneidad de símbolos sobre una celda de la cinta, de estados y de posición sobre la cinta, en un instante de tiempo determinado como un estado de 'inconsistencia' temporal dentro del proceso de computación. En el primer ejemplo se presenta todo el proceso de deducción de las descripciones instantáneas D_t. En los siguientes ejemplos sólo se presentan las descripciones instantáneas y las contradicciones obtenidas sin presentar el proceso de deducción, se hacen sólo las aclaraciones y demostraciones que se consideran convenientes.

Ejemplo 3.1. *Sea* $\mathcal{M}_1 =< Q, \Sigma, M, I >$ *una máquina de Turing con* $Q = \{q_1, q_2\}$, $\Sigma = \{s_0, s_1\}$, $I = \{i_1, i_2, i_3\}$, *donde* $i_1 = q_1 s_1 s_0 q_2$, $i_2 = q_1 s_1 s_1 q_2$ *y*

$i_3 = q_2 s_1 s_0 q_2$ (figura 3.1) y sea $n_1 = s_1$ la secuencia de entrada para $\mathcal{M}_1$.

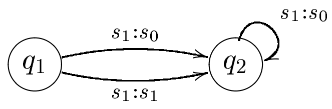

Figura 3.1: Grafo máquina $\mathcal{M}_1$

La teoría $\Delta'_{C_1^{=}}(\mathcal{M}_1(n_1)) = \Delta^{-1}_{C_1^{=}} \cup \{(\mathrm{A6}), (\mathrm{A7}), (\mathrm{A8}), (\mathrm{A9}), (\mathrm{A10}), (\mathrm{A11}), (\mathrm{A12}), (\mathrm{A13}), (\mathrm{A14}), (\mathrm{A15}), (\mathrm{A16})\}$, *donde:*

$$\forall t \forall x \Bigg(\Big(Q_1(t,x) \wedge S_1(t,x) \Big) \rightarrow \Big(Q_2(t',x) \wedge S_0(t',x) \wedge \forall y \Big((y \neq x) \rightarrow \Big(\big(S_0(t,y) \rightarrow S_0(t',y)\big) \wedge \big(S_1(t,y) \rightarrow S_1(t',y)\big) \Big) \Big) \Big) \Bigg), \tag{A6}$$

$$\forall t \forall x \Bigg(\Big(Q_1(t,x) \wedge S_1(t,x) \Big) \rightarrow \Big(Q_2(t',x) \wedge S_1(t',x) \wedge \forall y \Big((y \neq x) \rightarrow \Big(\big(S_0(t,y) \rightarrow S_0(t',y)\big) \wedge \big(S_1(t,y) \rightarrow S_1(t',y)\big) \Big) \Big) \Big) \Bigg), \tag{A7}$$

$$\forall t \forall x \Bigg(\Big(Q_2(t,x) \wedge S_1(t,x) \Big) \rightarrow \Big(Q_2(t',x) \wedge S_0(t',x) \wedge \forall y \Big((y \neq x) \rightarrow \Big(\big(S_0(t,y) \rightarrow S_0(t',y)\big) \wedge \big(S_1(t,y) \rightarrow S_1(t',y)\big) \Big) \Big) \Big) \Bigg), \tag{A8}$$

$$Q_1(0,0) \wedge S_1(0,0) \wedge \forall y \big((y \neq 0) \rightarrow S_0(0,y) \big), \tag{A9}$$

$$\forall t \forall x \Big((t < 0) \rightarrow \big(\neg Q_1(t,x) \wedge \neg Q_2(t,x) \wedge \neg S_0(t,x) \wedge \neg S_1(t,x) \big) \Big), \tag{A10}$$

$$\forall t \forall x (S_0(t,x) \rightarrow \neg S_1(t,x)), \tag{A11}$$

$$\forall t \forall x (S_1(t,x) \rightarrow \neg S_0(t,x)), \tag{A12}$$

$$\forall t \forall x \Big(\big(Q_1(t,x) \vee Q_2(t,x)\big) \rightarrow \forall y \Big(x \neq y \rightarrow \big(\neg Q_1(t,y) \wedge \neg Q_2(t,y)\big)\Big)\Big), \tag{A13}$$

$$\forall t \forall x (Q_1(t,x) \rightarrow \neg Q_2(t,x)), \tag{A14}$$

$$\forall t \forall x (Q_2(t,x) \rightarrow \neg Q_1(t,x)), \tag{A15}$$

$$\forall t \forall x \Bigg(\bigg(\Big(Q_1(t,x) \wedge S_0(t,x) \Big) \vee \Big(Q_2(t,x) \wedge S_0(t,x) \Big) \bigg) \rightarrow$$
$$\forall u \forall y \bigg(t < u \rightarrow \Big(\neg Q_1(u,y) \wedge \neg Q_2(u,y) \wedge \neg S_0(u,y) \wedge S_1(u,y) \Big) \bigg) \Bigg). \tag{A16}$$

La fórmula asociada a la descripción instantánea inicial ($t = 0$), *que se denota por* D_0 *(definición 1.5), está dada por* (A9).

Aplicando IU($t/0, x/0$*) en* (A6), *SIMP, en* (A9), *MP y de nuevo SIMP se obtiene:*

$$Q_2(0',0) \wedge S_0(0',0). \tag{3.9}$$

Aplicando IU($t/0, x/0$*) en* (A7), *SIMP, en* (A9), *MP y de nuevo SIMP se obtiene:*

$$Q_2(0',0) \wedge S_1(0',0). \tag{3.10}$$

De (3.9) *y* (3.10), *bajo la interpretación intensional* $\mathcal{I}$, *se tiene que en el tiempo* $t = 0' = 1$ *y en la posición* $x = 0$, $\mathcal{M}_1(n_1)$ *va a estar leyendo tanto el símbolo* s_0 *como el símbolo* s_1. *Aplicando IU(*$t/0', x/0$*) en* (A11) *y* (A12) *se obtiene:*

$$S_0(0',0) \rightarrow \neg S_1(0',0), \tag{3.11}$$

$$S_1(0',0) \rightarrow \neg S_0(0',0). \tag{3.12}$$

Aplicando SIMP en (3.9), *MP con* (3.11), *SIMP en* (3.10) *y CONJ se obtiene:*

$$S_1(0',0) \wedge \neg S_1(0',0).$$

De forma similar se obtiene:

$$S_0(0',0) \wedge \neg S_0(0',0).$$

Lo que, bajo la interpretación intencional $\mathcal{I}$, *indica que existe una contradicción con respecto al símbolo que hay en la posición* $x = 0$ *en el tiempo* $t = 0'$. *En esta posición y tiempo están y no están tanto* s_0 *como* s_1.

Para las demás posiciones $y \neq 0$, para el tiempo $t = 0'$, de (A9)*, aplicando IU($t/0, x/0$) bien sea en* (A6) *o en* (A7)*, MP y SIMP se obtiene:*

$$\big(S_0(0', y) \to \neg S_0(0', y)\big) \wedge \big(S_1(0', y) \to \neg S_1(0', y)\big). \tag{3.13}$$

Por SIMP en (A9)*, IU(y/y) para cada valor $y \neq 0$, aplicando SIMP en* (3.13)*, MP y CONJ se obtiene:*

$$\bigwedge_{y \neq 0} S_0(0', y).$$

Puede entonces obtenerse la fórmula asociada a la descripción instantánea de $\mathcal{M}_1(n_1)$, para el tiempo $t = 0'$, denotada por $D_{0'}$, como la conjunción de los hechos positivos obtenidos anteriormente, por lo tanto:

$$D_{0'} \equiv Q_2(0', 0) \wedge S_0(0', 0) \wedge S_1(0', 0) \wedge \left(\bigwedge_{y \neq 0} S_0(0', y)\right).$$

De forma similar a como se obtuvo $D_{0'}$, de $D_{0'}$ y (A8) *se puede obtener $D_{0''}$, donde:*

$$D_{0''} \equiv Q_2(0'', 0) \wedge S_0(0'', 0) \wedge \left(\bigwedge_{y \neq 0} S_0(0'', y)\right).$$

En $\mathcal{M}_1$ existe sólo una instrucción (i_3) que puede ser ejecutada cuando la máquina se encuentra en el estado q_2 leyendo el símbolo s_1, y no existe ninguna instrucción a ser ejecutada cuando $\mathcal{M}_1$ se encuentra en el estado q_2 leyendo el símbolo s_0, por lo tanto, en $\Delta'_{C_1^{=}}(\mathcal{M}_1(n_1))$ no existen axiomas que puedan ser utilizados para obtener nuevos hechos positivos que den cuenta de estados, posición de la máquina y símbolos sobre la cinta. Sin embargo, aunque no se puede demostrar una situación de simultaneidad, aplicando IU($t/0', x/0$) en (A16)*, SIMP en $D_{0'}$ y MP se obtiene:*

$$\forall u \forall y \Bigg(0' < u \to \Big(\neg Q_1(u, y) \wedge \neg Q_2(u, y) \wedge \neg S_0(u, y) \wedge S_1(u, y)\Big)\Bigg). \tag{3.14}$$

Aplicando IU($u/0'', y/0$) en (3.14)*, IU($x/0'$) en* (A4)*, MP y SIMP se obtiene:*

$$\neg S_0(0'', 0), \tag{3.15}$$

$$\neg Q_2(0'', 0). \tag{3.16}$$

Aplicando SIMP en $D_{0''}$ y CONJ con (3.15) *y* (3.16) *respectivamente, se obtiene:*

$$S_0(0'', 0) \wedge \neg S_0(0'', 0), \tag{3.17}$$

$$Q_2(0'', 0) \wedge \neg Q_2(0'', 0). \tag{3.18}$$

Para las posiciones $y \neq 0$, que son infinitas, de forma similar a como se obtuvo (3.17), *se obtiene:*

$$S_0(0'', y) \wedge \neg S_0(0'', y).$$

Lo que indica que el axioma (A16) *puede generar contradicciones aún en situaciones donde no se presenta simultaneidad. Todas estas contradicciones surgen debido a que el axioma* (A16) *asegura el supuesto (sección 1.1) de que $\mathcal{M}_1(n_1)$ es 'apagada' en el primer momento (si existe alguno) despúes de que la máquina se detiene, por lo tanto la máquina no está en ningún estado y posición sobre la cinta y no contiene símbolos en las casillas de la cinta (ni siquiera el vacío). Como $\mathcal{M}_1(n_1)$ en el tiempo $t = 0'$ se encuentra en el estado q_2 leyendo tanto el símbolo s_0 como el símbolo s_1, es contradictoria la fórmula utilizada para determinar si $\mathcal{M}_1(n_1)$ se detiene o no.*

Para los tiempos $t > 0''$ no se pueden deducir más hechos positivos relacionados con símbolos sobre la cinta, estado y posición de la máquina.

Tomando las fórmulas asociadas a las descripciones instantáneas D_0, $D_{0'}$ y $D_{0''}$, representando los símbolos sobre la cinta como subconjuntos de Σ y los estados en una posición como subconjuntos de Q, para poder simbolizar las situaciones de simultaneidad, y utilizando el símbolo $^\bullet$ para indicar la presencia de contradicciones (como ejemplo: $S_0^\bullet \equiv S_0 \wedge \neg S_0$), se representa la secuencia de descripciones instantáneas de $\mathcal{M}_1(n_1)$ por la figura 3.2.

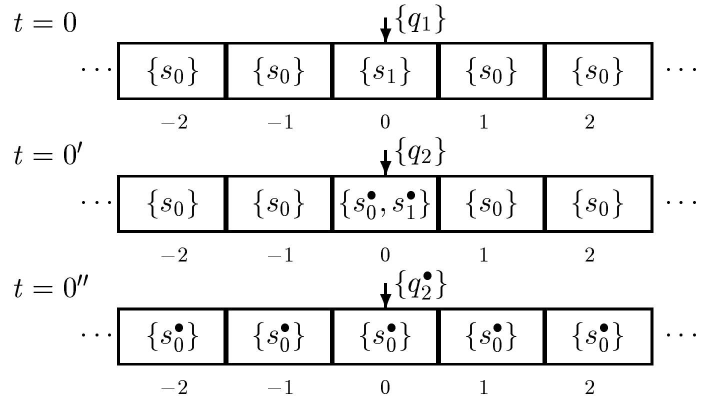

Figura 3.2: Descripciones instantáneas de $\mathcal{M}_1(n_1)$

Ejemplo 3.2. *Sea* $\mathcal{M}_2 =< Q, \Sigma, M, I >$ *una máquina de Turing con* $Q = \{q_1, q_2\}$, $\Sigma = \{s_0, s_1\}$, $I = \{i_1, i_2, i_3\}$, *donde* $i_1 = q_1 s_1 R q_1$, $i_2 = q_1 s_1 R q_2$ *y* $i_3 = q_1 s_0 s_1 q_2$ *(figura 3.3) y sea* $n_2 = s_1$ *la secuencia de entrada para* $\mathcal{M}_2$.

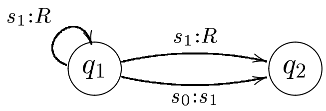

Figura 3.3: Grafo máquina $\mathcal{M}_2$

Para este caso la secuencia de descripciones instantáneas obtenida se representa por la figura 3.4.

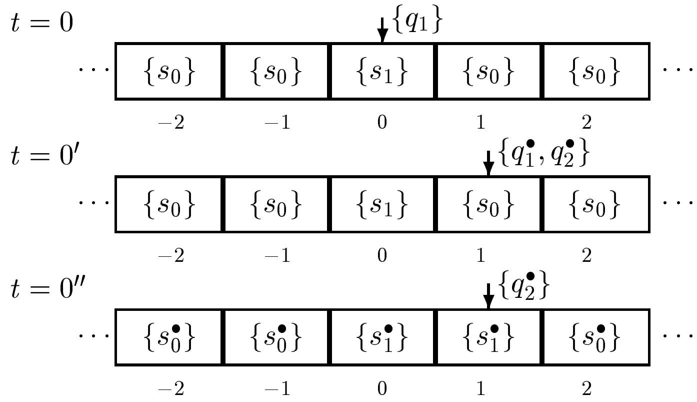

Figura 3.4: Descripciones instantáneas de $\mathcal{M}_2(n_2)$

Ejemplo 3.3. *Sea* $\mathcal{M}_3 =< Q, \Sigma, M, I >$ *una máquina de Turing con* $Q = \{q_1, q_2\}$, $\Sigma = \{s_0, s_1\}$, $I = \{i_1, i_2, i_3\}$, *donde* $i_1 = q_1 s_1 s_1 q_2$, $i_2 = q_1 s_1 R q_2$ *y* $i_3 = q_2 s_0 s_1 q_2$ *(figura 3.5) y sea* $n_3 = s_1$ *la secuencia de entrada para* $\mathcal{M}_3$.

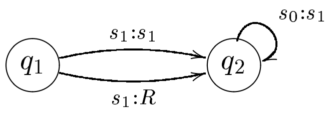

Figura 3.5: Grafo máquina $\mathcal{M}_3$

Para este caso la secuencia de descripciones instantáneas obtenida se representa por la figura 3.6.

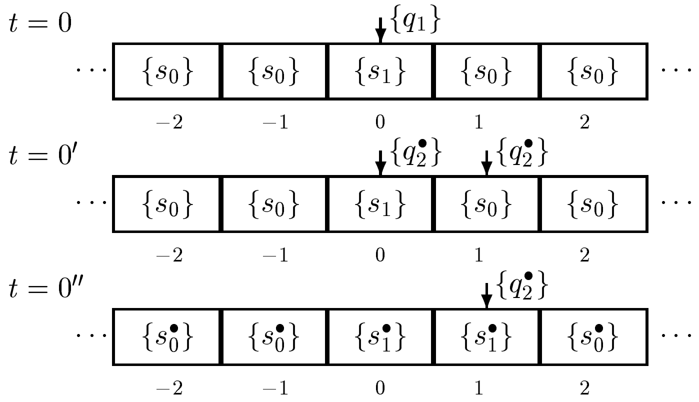

Figura 3.6: Descripciones instantáneas de $\mathcal{M}_3(n_3)$

Ejemplo 3.4. *Sea $\mathcal{M}_4 =< Q, \Sigma, M, I >$ una máquina de Turing con $Q = \{q_1, q_2\}$, $\Sigma = \{s_0, s_1\}$, $I = \{i_1, i_2\}$, donde $i_1 = q_1 s_1 s_1 q_2$ y $i_2 = q_1 s_1 s_0 q_2$ (figura 3.7) y sea $n_4 = s_1$ la secuencia de entrada para $\mathcal{M}_4$.*

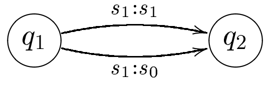

Figura 3.7: Grafo máquina $\mathcal{M}_4$

Para este caso la secuencia de descripciones instantáneas obtenida se representa por la figura 3.8.

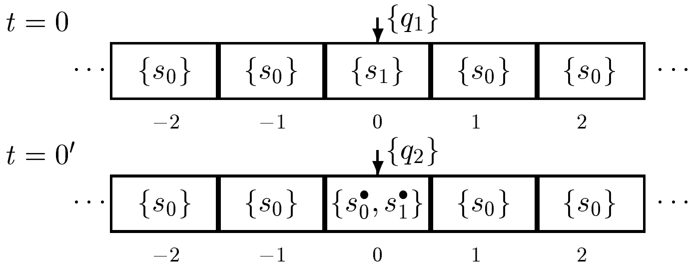

Figura 3.8: Descripciones instantáneas de $\mathcal{M}_4(n_4)$

Ejemplo 3.5. *Sea* $\mathcal{M}_5 =< Q, \Sigma, M, I >$ *una máquina de Turing con* $Q = \{q_1, q_2, q_3\}$*,* $\Sigma = \{s_0, s_1\}$*,* $I = \{i_1, i_2, i_3\}$*, donde* $i_1 = q_1 s_1 R q_2$*,* $i_2 = q_2 s_1 s_0 q_2$ *y* $i_3 = q_2 s_1 s_0 q_3$ *(figura 3.9) y sea* $n_5 = s_1 s_1$ *la secuencia de entrada para* $\mathcal{M}_5$*.*

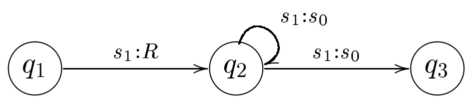

Figura 3.9: Grafo máquina $\mathcal{M}_5$

Para este caso la secuencia de descripciones instantáneas obtenida se representa por la figura 3.10.

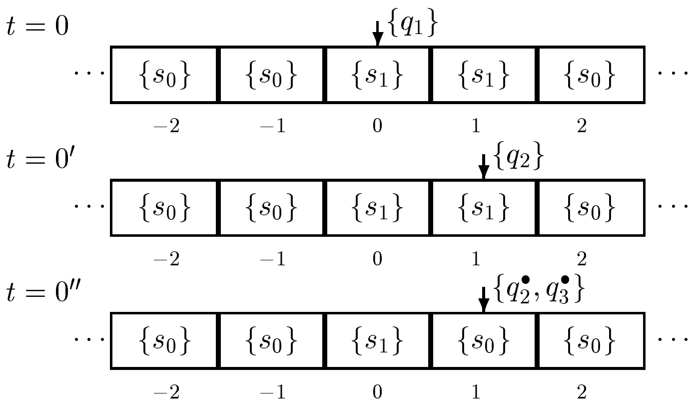

Figura 3.10: Descripciones instantáneas de $\mathcal{M}_5(n_5)$

Ejemplo 3.6. *Sea* $\mathcal{M}_6 =< Q, \Sigma, M, I >$ *una máquina de Turing con* $Q = \{q_1, q_2\}$*,* $\Sigma = \{s_0, s_1\}$*,* $I = \{i_1, i_2, i_3\}$*, donde* $i_1 = q_1 s_1 s_0 q_2$*,* $i_2 = q_1 s_1 R q_2$ *y* $i_3 = q_2 s_0 s_1 q_2$ *(figura 3.11) y sea* $n_6 = s_1$ *la secuencia de entrada para* $\mathcal{M}_6$*.*

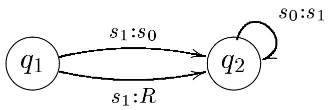

Figura 3.11: Grafo máquina $\mathcal{M}_6$

Para este caso la secuencia de descripciones instantáneas obtenida se representa por la figura 3.12.

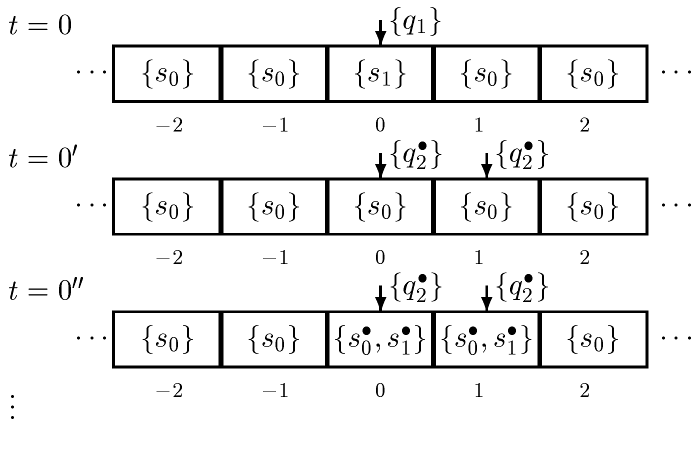

Figura 3.12: Descripciones instantáneas de $\mathcal{M}_6(n_6)$

Para todo tiempo $t > 0''$*, La descripción instantánea es igual a la del tiempo* $t = 0''$*. La máquina* $\mathcal{M}_6(n_6)$ *no se detiene.*

Las contradicciones con respecto a los símbolos s_0 *y* s_1 *en el tiempo* $t = 0''$*, en las posiciones* $x = 0$ *y* $x = 0'$*, se deducen de la situación de* $\mathcal{M}_6(n_6)$ *en el tiempo* $t = 0'$ *así:*

$$Q_2(0', 0) \wedge S_0(0', 0), \tag{3.19}$$

$$Q_2(0', 0') \wedge S_0(0', 0'). \tag{3.20}$$

En $\Delta'_{C_1^=}(\mathcal{M}_6(n_6))$ *existe un axioma* $(A8)$ *para la instrucción* i_3 *construido bajo el esquema* (A5+j (I))*. Aplicando* $IU(t/0', x/0)$ *en dicho axioma, MP* *con* (3.19) *y SIMP se obtiene:*

$$Q_2(0'',0) \wedge S_1(0'',0), \tag{3.21}$$

$$\forall y\Big((y \neq 0) \rightarrow \Big(\big(S_0(0',y) \rightarrow S_0(0'',y)\big) \wedge \big(S_1(0',y) \rightarrow S_1(0'',y)\big)\Big)\Big). \tag{3.22}$$

De forma similar, de (3.20) *y* (*A*8) *se obtiene:*

$$Q_2(0'',0') \wedge S_1(0'',0'), \tag{3.23}$$

$$\forall y\Big((y \neq 0') \rightarrow \Big(\big(S_0(0',y) \rightarrow S_0(0'',y)\big) \wedge \big(S_1(0',y) \rightarrow S_1(0'',y)\big)\Big)\Big). \tag{3.24}$$

Aplicando IU($y/0'$) en (3.22), *IU($x/0$) en* (1.1), *MP y SIMP se obtiene:*

$$S_0(0',0') \rightarrow S_0(0'',0'). \tag{3.25}$$

Aplicando SIMP en (3.20) *y MP con* (3.25) *se obtiene:*

$$S_0(0'',0'). \tag{3.26}$$

De forma similar se obtiene:

$$S_0(0'',0). \tag{3.27}$$

De (3.21), (3.23), (3.26) *y* (3.27), *aplicando SIMP y CONJ se obtiene:*

$$S_0(0'',0) \wedge S_1(0'',0), \tag{3.28}$$

$$S_0(0'',0') \wedge S_1(0'',0'). \tag{3.29}$$

De (3.28) *y* (3.29), *teniendo en cuenta que en* $\mathcal{M}_6(n_6)$ *existe un axioma igual al axioma* (A11) *del ejemplo 3.1 y otro igual al axioma* (A12) *del ejemplo 3.1, para asegurar la unicidad de los símbolos, se deducen las contradicciones:*

$$S_0(0'',0) \wedge \neg S_0(0'',0),$$
$$S_1(0'',0) \wedge \neg S_1(0'',0),$$
$$S_0(0'',0') \wedge \neg S_0(0'',0'),$$
$$S_1(0'',0') \wedge \neg S_1(0'',0').$$

La simultaneidad de símbolos para el tiempo $t = 0''$, *en las posiciones* $x = 0$ *y* $x = 0'$, *surgen por la multiplicidad de posiciones de* $\mathcal{M}_6(n_6)$ *en el tiempo* $t = 0'$ *y por suponer, en la codificación de las instrucciones, que en cada instante de tiempo la máquina sólo puede modificar el símbolo sobre una casilla de la cinta y que por lo tanto las demás casillas conservan el símbolo del instante de tiempo anterior.*

3.3. Máquina de Turing paraconsistente

Teniendo ya suficientes ilustraciones sobre las diferentes situaciones que se pueden presentar en $\Delta'_{C_1^=}(\mathcal{M}(n))$ al axiomatizar MTNDs, cabe preguntarse cuales de las secuencias de descripciones instantáneas obtenidas en los ejemplos anteriores pueden ser tomadas como computaciones y cuales son las reglas que definen el comportamiento de $\mathcal{M}(n)$ bajo la nueva lógica.

De acuerdo con la definición de computación ofrecida en el apéndice A (definición A.9), las condiciones necesarias y suficientes para que una secuencia de descripciones instantáneas sea una computación son: que dicha secuencia sea finita, que las descripciones instantáneas se impliquen la una a la otra formando una 'cadena' de descripciones instantáneas de acuerdo con ciertas reglas de cambio (definición A.7) y que la última descripción instantánea de la 'cadena' tenga la propiedad de ser una descripción instantánea terminal (definición A.8). De acuerdo con los ejemplos anteriores, es claro que se debe redefinir lo que es una descripción instantánea, para poder describir las situaciones de simultaneidad que se presentan en el proceso de computación, y se deben también redefinir las reglas que regulan el cambio de descripción instantánea. Sin embargo, aunque sigue siendo razonable exigir finitud en la secuencia de descripciones instantáneas y que la última descripción instantánea sea terminal, surge el interrogante de si se deben exigir o no ciertas propiedades de 'consistencia' en la última descripción de la secuencia, para que esta pueda ser tomada como un resultado 'coherente' de la computación.

Si se toma de una manera ortodoxa el deseo de tener superadas todas las contradicciones en la última descripción instantánea, para poder aceptar la secuencia de descripciones instantáneas como una computación, ninguno de los ejemplos anteriores podría ser tomado como una computación, puesto que en todos ellos se demuestran contradicciones en la última descripción instantánea, ya sea con respecto a los símbolos, estado o posición. Aunque habrán otros casos en los que todas las contradicciones se pueden superar, esta condición resulta ser demasiado restrictiva.

Una forma de aceptar todas las secuencias finitas de descripciones instantáneas como computaciones, sin renunciar al ideal de la superación total de las contradicciones, sería tomar como resultado de la computación la última descripción instantánea 'consistente' (sin contradicciones) y no necesariamente la última de la secuencia. Esto implicaría perder, en muchos casos, buena parte del procesamiento que la máquina realiza.

Una condición, un poco más razonable, es pedir consistencia únicamente con respecto a los símbolos sobre la cinta en la última descripción instantánea. De esta forma muchas de las secuencias de descripciones instantáneas que no eran tenidas en cuenta como computaciones en el primer caso, ahora si lo serían. Además, el resultado podría ser interpretado de la forma usual sin 'ambigüedades'.

La opción más heterodoxa consiste en no pedir ninguna condición de 'consistencia' a la descripción instantánea terminal, dejando el problema de la resolución de las 'ambigüedades' a la interpretación de los resultados, o simplemente aceptando que el proceso de computación puede arrojar múltiples resultados, uno por cada una de las posibles combinaciones de símbolos 'simultáneos' en la descripción instantánea final.

A continuación se define formalmente lo que es una 'computación paraconsistente', en el sentido que es obtenida de lo que se puede deducir en las teorías paraconsistentes $\Delta'_{C_1^=}(\mathcal{M}(n))$. Para esto, inicialmente se redefine lo que es una descripción instantánea (definición 3.1), luego se redefinen las reglas de cambio de descripción instantánea (definición 3.2), se redefine lo que es una descripción instantánea terminal (definición 3.3) y por último se redefine lo que es una computación (definición 3.4), sin pedir ninguna condición de 'consistencia' a la descripción instantánea terminal, con la idea de permitir que la máquina opere libremente bajo situaciones de 'contradicción', delegando el control de las 'contradicciones' a quien define y usa la máquina.

En la sección 3.4 se introducen algunas condiciones que se pueden adicionar a las instrucciones con el objetivo de controlar las 'contradicciones'.

Definición 3.1 (Descripción instantánea bajo la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$). *Sea $\mathcal{M} =< Q, \Sigma, M, I >$ una máquina de Turing con $Q = \{q_1, q_2, \ldots, q_n\}$, $\Sigma = \{s_0, s_1, \ldots, s_m\}$, $M = \{R, L\}$ y $I = \{i_1, i_2, \ldots, i_k\}$, cuyo comportamiento para toda entrada n está determinado por la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$. Una descripción instantánea δ_t de $\mathcal{M}$ es una secuencia finita de elementos $\mathcal{S}_j \in P(\Sigma) - \{\varnothing\}$ ($P(\Sigma)$ denota el conjunto de partes o conjuno potencia de Σ) y elementos $\mathcal{Q}_i \in P(Q) - \{\varnothing\}$, donde existe por lo menos un elemento $\mathcal{Q}_i$ y un elemento $\mathcal{S}_j$, y ninguno de los elementos $\mathcal{Q}_i$ es el último de izquierda a derecha en la secuencia.*

Ejemplo 3.7.

$$\delta_t \equiv \{q_2\}\{s_0, s_1\}\{q_2\}\{s_0, s_1\},$$

δ_t describe la situación de $\mathcal{M}_6(n_6)$ en el tiempo $t = 0''$ (ejemplo 3.6).

Definición 3.2 (Cambio de descripción instantánea bajo la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$). *Sea $\mathcal{M}$ una máquina de Turing cuyo comportamiento para* *toda entrada n está determinado por la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$ y sean δ y γ descripciones instantáneas de $\mathcal{M}$. El cambio de la descripción instantánea δ a la descripción instantánea γ bajo la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$ se representa por $\delta \xrightarrow{\mathcal{M}_{C_1^=}} \gamma$, lo cual significa que se satisfacen las condiciones*[3]*:*

1. *Si δ sólo tiene un símbolo $\mathcal{Q}_i = \{q_{i,1}, q_{i,2}, \ldots, q_{i,u}\} \in P(Q) - \{\varnothing\}$ entonces:*

   1.1. *Si $\mathcal{M}$ sólo contiene instrucciones $i_1, i_2, \ldots, i_k$ de la forma* (A5+j (I))*, tales que $i_i = q_{i,a}s_{j,b}s_{k,c}q_{l,d}$, entonces δ es de la forma $\delta \equiv P\mathcal{Q}_i\mathcal{S}_jQ$ y γ debe ser de la forma $\gamma \equiv P\mathcal{Q}_l\mathcal{S}_kQ$.*

   1.2. *Si $\mathcal{M}$ sólo contiene instrucciones $i_1, i_2, \ldots, i_k$ de la forma* (A5+j (II))*, tales que $i_i = q_{i,a}s_{j,b}Rq_{l,c}$, entonces:*

      1.2.1. *Si δ es de la forma $\delta \equiv P\mathcal{Q}_i\mathcal{S}_j\mathcal{S}_kQ$ entonces γ debe ser de la forma $\gamma \equiv P\mathcal{S}_j\mathcal{Q}_l\mathcal{S}_kQ$.*

      1.2.2. *Si δ es de la forma $\delta \equiv P\mathcal{Q}_i\mathcal{S}_j$ entonces γ debe ser de la forma $\gamma \equiv P\mathcal{S}_j\mathcal{Q}_l\{s_0\}$.*

   1.3. *Si $\mathcal{M}$ sólo contiene instrucciones $i_1, i_2, \ldots, i_k$ de la forma* (A5+j (III))*, tales que $i_i = q_{i,a}s_{j,b}Lq_{l,c}$, entonces:*

      1.3.1. *Si δ es de la forma $\delta \equiv P\mathcal{S}_k\mathcal{Q}_i\mathcal{S}_jQ$ entonces γ debe ser de la forma $\gamma \equiv P\mathcal{Q}_l\mathcal{S}_k\mathcal{S}_jQ$.*

      1.3.2. *Si δ es de la forma $\delta \equiv \mathcal{Q}_i\mathcal{S}_jP$ entonces γ debe ser de la forma $\gamma \equiv \mathcal{Q}_l\{s_0\}\mathcal{S}_jP$.*

   1.4. *Si $\mathcal{M}$ contiene instrucciones $i_1, i_2, \ldots, i_k$ de la forma* (A5+j (I)) *y de la forma* (A5+j (II))*, tales que $i_i = q_{i,a}s_{j,b}s_{k,c}q_{l,d}$ o $i_i = q_{i,a}s_{j,b}Rq_{m,c}$, entonces:*

      1.4.1. *Si δ es de la forma $\delta \equiv P\mathcal{Q}_i\mathcal{S}_j\mathcal{S}_lQ$ entonces γ debe ser de la forma $\gamma \equiv P\mathcal{Q}_l\mathcal{S}_k\mathcal{Q}_m\mathcal{S}_lQ$.*

      1.4.2. *Si δ es de la forma $\delta \equiv P\mathcal{Q}_i\mathcal{S}_j$ entonces γ debe ser de la forma $\gamma \equiv P\mathcal{Q}_l\mathcal{S}_k\mathcal{Q}_m\{s_0\}$.*

   1.5. *Si $\mathcal{M}$ contiene instrucciones $i_1, i_2, \ldots, i_k$ de la forma* (A5+j (I)) *y de la forma* (A5+j (III))*, tales que $i_i = q_{i,a}s_{j,b}s_{k,c}q_{l,d}$ o $i_i = q_{i,a}s_{j,b}Lq_{m,c}$, entonces:*

---
[3] Los símbolos $\mathcal{Q}_i = \{q_{i,1}, q_{i,2}, \ldots, q_{i,u}\} \in P(Q) - \{\varnothing\}$ representan conjuntos de estados indexados por i. Los símbolos $\mathcal{S}_j = \{s_{j,1}, s_{j,2}, \ldots, s_{j,v}\} \in P(\Sigma) - \{\varnothing\}$ representan conjuntos de símbolos indexados por j y P, Q representan secuencias (posiblemente vacías) de símbolos $\mathcal{S}_k \in P(\Sigma) - \{\varnothing\}$.

1.5.1. Si δ es de la forma $\delta \equiv P\mathcal{S}_l\mathcal{Q}_i\mathcal{S}_jQ$ entonces γ debe ser de la forma $\gamma \equiv P\mathcal{Q}_m\mathcal{S}_l\mathcal{Q}_l\mathcal{S}_kQ$.

1.5.2. Si δ es de la forma $\delta \equiv \mathcal{Q}_i\mathcal{S}_jP$ entonces γ debe ser de la forma $\gamma \equiv \mathcal{Q}_m\{s_0\}\mathcal{Q}_l\mathcal{S}_kP$.

1.6. Si $\mathcal{M}$ contiene instrucciones $i_1, i_2, \ldots, i_k$ de la forma (A5+j (II)) *y de la forma* (A5+j (III))*, tales que $i_i = q_{i,a}s_{j,b}Rq_{l,d}$ o $i_i = q_{i,a}s_{j,b}Lq_{m,c}$, entonces:*

1.6.1. Si δ es de la forma $\delta \equiv P\mathcal{S}_k\mathcal{Q}_i\mathcal{S}_j\mathcal{S}_nQ$ entonces γ debe ser de la forma $\gamma \equiv P\mathcal{Q}_m\mathcal{S}_k\mathcal{S}_j\mathcal{Q}_l\mathcal{S}_nQ$.

1.6.2. Si δ es de la forma $\delta \equiv P\mathcal{S}_k\mathcal{Q}_i\mathcal{S}_j$ entonces γ debe ser de la forma $\gamma \equiv P\mathcal{Q}_m\mathcal{S}_k\mathcal{S}_j\mathcal{Q}_l\{s_0\}$.

1.6.3. Si δ es de la forma $\delta \equiv \mathcal{Q}_i\mathcal{S}_j\mathcal{S}_kP$ entonces γ debe ser de la forma $\gamma \equiv \mathcal{Q}_m\{s_0\}\mathcal{S}_j\mathcal{Q}_l\mathcal{S}_kP$.

1.6.4. Si δ es de la forma $\delta \equiv \mathcal{Q}_i\mathcal{S}_j$ entonces γ debe ser de la forma $\gamma \equiv \mathcal{Q}_m\{s_0\}\mathcal{S}_j\mathcal{Q}_l\{s_0\}$.

1.7. Si $\mathcal{M}$ contiene instrucciones $i_1, i_2, \ldots, i_k$ de la forma (A5+j (I))*, de la forma* (A5+j (II)) *y de la forma* (A5+j (III))*, tales que $i_i = q_{i,a}s_{j,b}s_{u,b}q_{v,d}$, $i_i = q_{i,a}s_{j,b}Rq_{l,d}$ o $i_i = q_{i,a}s_{j,b}Lq_{m,c}$, entonces:*

1.7.1. Si δ es de la forma $\delta \equiv P\mathcal{S}_k\mathcal{Q}_i\mathcal{S}_j\mathcal{S}_nQ$ entonces γ debe ser de la forma $\gamma \equiv P\mathcal{Q}_m\mathcal{S}_k\mathcal{Q}_v\mathcal{S}_u\mathcal{Q}_l\mathcal{S}_nQ$.

1.7.2. Si δ es de la forma $\delta \equiv P\mathcal{S}_k\mathcal{Q}_i\mathcal{S}_j$ entonces γ debe ser de la forma $\gamma \equiv P\mathcal{Q}_m\mathcal{S}_k\mathcal{Q}_v\mathcal{S}_u\mathcal{Q}_l\{s_0\}$.

1.7.3. Si δ es de la forma $\delta \equiv \mathcal{Q}_i\mathcal{S}_j\mathcal{S}_kP$ entonces γ debe ser de la forma $\gamma \equiv \mathcal{Q}_m\{s_0\}\mathcal{Q}_v\mathcal{S}_u\mathcal{Q}_l\mathcal{S}_kP$.

1.7.4. Si δ es de la forma $\delta \equiv \mathcal{Q}_i\mathcal{S}_j$ entonces γ debe ser de la forma $\gamma \equiv \mathcal{Q}_m\{s_0\}\mathcal{Q}_v\mathcal{S}_u\mathcal{Q}_l\{s_0\}$.

2. Si δ tiene más de un símbolo $\mathcal{Q}_i = \{q_{i,1}, q_{i,2}, \ldots, q_{i,u}\} \in P(Q) - \{\varnothing\}$, se deben aplicar las reglas especificadas en el paso 1 en cada una de las posiciones donde se encuentra un $\mathcal{Q}_i$, teniendo en cuenta que se deben unir en la descripción instantánea resultante γ los $\mathcal{Q}_i$ contiguos y se deben unir los símbolos $\mathcal{S}_j$ de δ a los $\mathcal{S}_k$ de γ correspondientes. Además, para los símbolos $\mathcal{Q}_i$ donde no se puede aplicar ninguna instrucción este símbolo desaparece en γ.

Ejemplo 3.8. *Para el ejemplo 3.4 se tiene la siguiente secuencia de cambios* *de descripciones instantáneas:*

$$\{q_1\}\{s_1\}$$
$$\xrightarrow{\mathcal{M}_{C_1^=}}\{q_2\}\{s_0, s_1\}$$

Definición 3.3 (Descripción instantánea terminal bajo la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$). *Sea $\mathcal{M}$ un máquina de Turing cuyo comportamiento para toda entrada n está determinado por la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$. Una descripción instantánea δ es llamada una descripción instantánea terminal, con respecto a $\mathcal{M}$, si no existe una descripción instantánea γ tal que $\alpha \xrightarrow{\mathcal{M}_{C_1^=}} \gamma$.*

Definición 3.4 (Computación bajo la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$). *Una computación de una máquina de Turing $\mathcal{M}$, cuyo comportamiento para toda entrada n está determinado por la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$, es una secuencia de descripciones instantáneas $\delta_0, \delta_1, \ldots, \delta_n$ tal que $\delta_i \xrightarrow{\mathcal{M}_{C_1^=}} \delta_{i+1}$, para $0 \leq i \leq n-1$, y δ_n es una descripción instantánea terminal con respecto a $\mathcal{M}$. En este caso se escribe $\delta_n = Res_{\mathcal{M}_{c_1^=}}(\delta_0)$ y se denomina a δ_n el resultado de δ_0 con respecto a $\mathcal{M}$ bajo la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$. Como δ_n puede tener en diferentes posiciones símbolos "simultaneos", es decir, símbolos $\mathcal{S}_j = \{s_{j,1}, s_{j,2}, \ldots, s_{j,n}\} \in P(\Sigma) - \{\varnothing\}$ con cardinal mayor que 1, cada una de las secuencias de símbolos $s_j \in \Sigma$ obtenidas seleccionando de cada $\mathcal{S}_j$ en δ_n sólo uno de los símbolos $s_{j,i} \in \mathcal{S}_j$ es tomada como un resultado de $\mathcal{M}(\delta_0)$. De esta forma, δ_n puede contener múltiples resultados.*

Ejemplo 3.9. *La secuencia de cambios de descripciones instantáneas del ejemplo 3.8 es una computación, en la cual los múltiples resultados obtenidos son: s_0 y s_1.*

De acuerdo con la definición de computación bajo la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$ (definición 3.4) se define formalmente la noción de Turing paraconsistente.

Definición 3.5 (Máquina de Turing paraconsistente (MTP)). *Una MTP se define formalmente mediante una estructura matemática $\mathcal{M} =< Q, \Sigma, M, I >$ donde:*

- *$Q = \{q_1, q_2, \ldots, q_n\}$: Conjunto finito de estados de la máquina ($Q \neq \emptyset$).*
- *$\Sigma = \{s_0, s_1, \ldots, s_m\}$: Conjunto finito de símbolos de entrada/salida. Por convención s_0 representa el símbolo vacío ($\Sigma - \{s_0\} \neq \emptyset$).*

- *$M = \{R, L\}$: Conjunto de movimientos (R: derecha, L: izquierda).*
- *I: Subconjunto de $Q \times \Sigma \times \{\Sigma \cup M\} \times Q$. Conjunto de instrucciones $\{i_1, i_2, \ldots, i_k\}$ donde cada i_j es una cuadrupla de alguna de las formas:*

$$q_i s_j s_k q_l \tag{I'}$$
$$q_i s_j R q_l \tag{II'}$$
$$q_i s_j L q_l \tag{III'}$$

Puesto que en la máquina se pueden ejecutar de forma simultánea varias instrucciones, se pueden presentar situaciones de 'simultaneidad' tanto en símbolos como en estados, por lo tanto, estos elementos deben ser representados por símbolos $\mathcal{S}_j \in P(\Sigma) - \{\varnothing\}$ y $\mathcal{Q}_i \in P(Q) - \{\varnothing\}$ respectivamente. De esta forma, el tipo de instrucción (I') *especifica que para todas las posiciones x en que la máquina se encuentre en el estado q_i ($q_i \in \mathcal{Q}_i$), leyendo el símbolo s_j ($s_j \in \mathcal{S}_j$), en el siguiente instante de tiempo tendrá el símbolo s_k ($s_k \in \mathcal{S}_k$) y estará en el estado q_l ($q_l \in \mathcal{Q}_l$). El tipo de instrucción* (II') *especifica que para todas las posiciones x en que la máquina se encuentre en el estado q_i ($q_i \in \mathcal{Q}_i$), leyendo el símbolo s_j ($s_j \in \mathcal{S}_j$), en el siguiente instante de tiempo, en la posición $x+1$ la máquina estará en el estado q_l ($q_l \in \mathcal{Q}_l$). El tipo de instrucción* (III') *especifica que para todas las posiciones x en que la máquina se encuentre en el estado q_i ($q_i \in \mathcal{Q}_i$), leyendo el símbolo s_j ($s_j \in \mathcal{S}_j$), en el siguiente instante de tiempo, en la posición $x-1$ la máquina estará en el estado q_l ($q_l \in \mathcal{Q}_l$). Además, si la máquina se encuentra simultáneamente en más de una posición, es decir, existe más de un $\mathcal{Q}_i$, entonces en el siguiente instante de tiempo, en todas las posiciones, la máquina conservará los símbolos presentes en las casillas de la cinta, adicionando los símbolos que surjan como resultado de las instrucciones.*

3.4. Modificaciones en $\Delta'_{C_1^=}(\mathcal{M}(n))$ para controlar las contradicciones

En los ejemplos 3.1 a 3.6 se puede observar que, en la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$, normalmente las contradicciones con respecto a los predicados $S_j(t,x)$ están relacionados con situaciones de simultaneidad de símbolos en $\mathcal{M}(n)$ y las contradicciones con respecto a los predicados $Q_i(t,x)$ normalmente están relacionados con situaciones de simultaneidad de estados o de posiciones en

$\mathcal{M}(n)$. Sólo se obtienen contradicciones que no están relacionadas con situaciones de simultaneidad en los casos en los que se puede aplicar el axioma construido bajo el esquema (A5+$\overline{\overline{I}}$+$\overline{\overline{\Sigma}}$+$\overline{\overline{Q}}$+4) antes de que el proceso de computación finalice (como en el ejemplo 3.1). Este esquema axiomático se introdujo en la sección 1.2 para asegurar el supuesto de que la máquina es 'apagada' en el primer momento (si existe alguno) después de que la máquina se detiene, pero no debe influenciar el comportamiento de la máquina. Si se elimina este esquema axiomático de $\Delta'_{C_1^=}(\mathcal{M}(n))$, controlar las contradicciones en $\Delta'_{C_1^=}(\mathcal{M}(n))$ significaría controlar las situaciones de simultaneidad en $\mathcal{M}(n)$.

En la lógica $C_1^=$ (sección C.0.1) la expresión $A^\circ \equiv \neg(A \wedge \neg A)$ simboliza el 'buen comportamiento' de la expresión A, lo que indica que la expresión A se comporta 'clásicamente'. De forma dual, $\neg A^\circ \equiv \neg\neg(A \wedge \neg A) \equiv A \wedge \neg A$ simboliza el 'no buen comportamiento' o 'mal comportamiento' de la expresión A, lo que indica que la expresión A es 'contradictoria'. Para el caso de la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$ el 'mal comportamiento' está relacionado con 'simultaneidad'. Siempre que se dan situaciones de simultaneidad en $\mathcal{M}(n)$ se demuestran contradicciones en $\Delta'_{C_1^=}(\mathcal{M}(n))$ y siempre que se demuestran contradicciones en $\Delta'_{C_1^=}(\mathcal{M}(n))$ (sin el esquema axiomático (A5+$\overline{\overline{I}}$+$\overline{\overline{\Sigma}}$+$\overline{\overline{Q}}$+4)) se dan situaciones de simultaneidad en $\mathcal{M}(n)$. Sin embargo, para las situaciones en las que no se da simultaneidad en $\mathcal{M}(n)$ la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$ no demuestra el 'buen comportamiento' del predicado respectivo.

Con el objetivo de controlar las contradicciones en $\Delta'_{C_1^=}(\mathcal{M}(n))$ se pueden adicionar en los antecedentes de los axiomas que codifican las instrucciones (esquemas axiomáticos (A5+j (I)), (A5+j (II)) y (A5+j (III))) condiciones de 'mal comportamiento' así[4]:

$$\Big(\big(Q_i(t,x) \wedge S_j(t,x)\big) \wedge \big(\neg Q_i(t,x)^\circ\big)\Big) \to \alpha, \qquad \text{(COND1)}$$

$$\Big(\big(Q_i(t,x) \wedge S_j(t,x)\big) \wedge \big(\neg S_j(t,x)^\circ\big)\Big) \to \alpha, \qquad \text{(COND2)}$$

$$\Big(\big(Q_i(t,x) \wedge S_j(t,x)\big) \wedge \big(\neg Q_i(t,x)^\circ \wedge \neg S_j(t,x)^\circ\big)\Big) \to \alpha, \qquad \text{(COND3)}$$

$$\Big(\big(Q_i(t,x) \wedge S_j(t,x)\big) \wedge \big(\neg Q_i(t,x)^\circ \vee \neg S_j(t,x)^\circ\big)\Big) \to \alpha, \qquad \text{(COND4)}$$

donde α simboliza el consecuente de los esquemas axiomáticos (A5+j (I)), (A5+j (II)) o (A5+j (III)). De este modo, la implicación sólo podrá ser uti-

[4]No se pueden adicionar condiciones de 'buen comportamiento', puesto que $\Delta'_{C_1^=}(\mathcal{M}(n))$ no prueba el 'buen comportamiento' de los predicados respectivos cuando no se presentan situaciones de 'simultaneidad'.

lizada cuando se demuestren los predicados $Q_i(t,x)$, $S_j(t,x)$ y además se demuestre el 'mal comportamiento' de $Q_i(t,x)$, el 'mal comportamiento' de $S_j(t,x)$, ambos 'malos comportamientos' o alguno de los dos 'malos comportamientos' respectivamente.

Se puede definir entonces una nueva teoría $\Delta'''_{C_1^=}(\mathcal{M}(n))$, construida tal y como se construye $\Delta'_{C_1^=}(\mathcal{M}(n))$, pero eliminando el esquema axiomático (A5+$\overline{\overline{I}}$+$\overline{\overline{\Sigma}}$+$\overline{\overline{Q}}$+4) y permitiendo definir condiciones del tipo (COND1), (COND2), (COND3) y (COND4). Con esta nueva teoría se puede redefinir la noción de computación, tal como se hizo en la definición 3.4 con la teoría $\Delta'_{C_1^=}(\mathcal{M}(n))$. De esta nueva noción de computación se puede definir otro modelo de máquina de MTP, donde en las instrucciones se pueden adicionar condiciones para que la instrucción se ejecute sólo cuando se presenten situaciones de 'simultaneidad', con respecto al símbolo que está leyendo, con respecto al estado o posición en el que se encuentra la máquina, con respecto a ambos o con respecto a alguno de ellos.

3.5. Sobre el poder computacional de la máquina de Turing paraconsistente

Con la definición de MTP se buscaba obtener un modelo de computación que fuera computacionalmente más potente que una máquina de Turing. Sin embargo, en esta sección se demestra que el resultado es negativo.

Teorema 3.1. *Para toda MTP $\mathcal{M}$ (definición 3.5) se puede construir una MTD $\mathcal{M}'$ que simula el comportamiento de $\mathcal{M}$.*

Demostración. En [17] se presentan diferentes modificaciones a la definición de máquina de Turing y se demuestra que estos nuevos modelos son equivalentes computacionalmente a la máquina de Turing original. En particular, se define el modelo de máquina de Turing 'multicinta' [17, p. 198]. Siguiendo la demostración de equivalencia computacional entre este modelo de máquina de Turing y el modelo original, se puede demostrar que toda MTP $\mathcal{M}$ puede ser simulada por una MTD $\mathcal{M}'$. Para este caso, se debe construir una máquina de Turing $\mathcal{M}''$ con la cinta dividida en tres pistas. En la primer pista se almacenan los símbolos presentes en la cinta de $\mathcal{M}$, utilizando símbolos $\mathcal{S}_j \in P(\Sigma)-\{\varnothing\}$ para representar las situaciones de simultaneidad. En la segunda pista se almacenan símbolos $\mathcal{Q}_i \in P(Q)$ que indican los estados en los que se encuentra $\mathcal{M}$ en cada posición, además, unos marcadores de final para delimitar la primer y última posición donde hay símbolos de estados $\mathcal{Q}_i \neq \{\varnothing\}$. La tercer pista se utiliza para almacenar los símbolos de estados $\mathcal{Q}_j$ que serán adicionados a los estados en una posición debido a instrucciones con movimientos a izquierda o derecha. Un paso de computación en $\mathcal{M}$ es simulado por $\mathcal{M}''$, realizando primero un barrido de izquierda a derecha, determinando los símbolos de estado $\mathcal{Q}_j$ que serán adicionados a los estados de la posición siguiente, debido a instrucciones de la forma (A5+j (II)), y moviendo el marcador de final, si es necesario. Realizando un segundo barrido de derecha a izquierda, determinando los símbolos de estado $\mathcal{Q}_j$ que serán adicionados a los estados de la posición anterior, debido a instrucciones de la forma (A5+j (III)), y moviendo el marcador de final, si es necesario. Realizando un tercer barrido de izquierda a derecha, modificando los símbolos $\mathcal{S}_j$ presentes en la primer pista, de acuerdo con los contenidos de las dos primeras pistas y las instrucciones definidas en $\mathcal{M}$, y modificando los símbolos $\mathcal{Q}_i$ presentes en la segunda pista de acuerdo con los contenidos de las tres pistas y las instrucciones definidas en $\mathcal{M}$. Y realizando un último barrido de derecha a izquierda eliminando los símbolos de la tercer pista para comenzar la simulación del próximo paso de computación. Como se demuestra en [17, p. 195], una máquina de Turing con cada celda (casilla) de la cinta dividida en múltiples pistas, puede ser simulada por una máquina de Turing sin divisiones en las casillas de la cinta, por lo tanto, para $\mathcal{M}''$ existe una MTD $\mathcal{M}'$, sin divisiones en las casillas de la cinta, que simula el comportamiento de $\mathcal{M}''$ y por consiguiente el de $\mathcal{M}$. □

El anterior teorema también aplica si se adicionan condiciones a la MTP (sección 3.4).

Puesto que las MTDs son un subconjunto de las MTPs, por el teorema 3.1 queda demostrado que las MTPs son computacionalmente equivalentes a las MTDs.

Ejemplo 3.10. *Como ejemplo de aplicación del teorema 3.1: sea* $\mathcal{M} =< Q, \Sigma, M, I >$ *una MTP con* $Q = \{q_1, q_2, q_3\}$, $\Sigma = \{s_0, s_1\}$, $I = \{i_1, i_2, i_3, i_4\}$, *donde* $i_1 = q_1 s_1 s_0 q_1$, $i_2 = q_1 s_1 R q_2$, $i_3 = q_1 s_0 s_0 q_2$ *y* $i_4 = q_2 s_1 L q_3$ *(figura 3.13) y sea* $n = s_1 s_1$ *la secuencia de entrada para* $\mathcal{M}$.

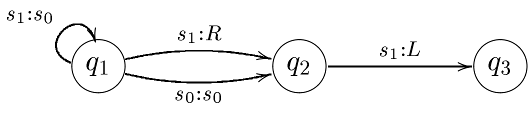

Figura 3.13: Grafo máquina $\mathcal{M}$

La computación de $\mathcal{M}(n)$ se representa en la figura 3.14.

$t = 0$

			$\downarrow\{q_1\}$			
$\cdots$	$\{s_0\}$	$\{s_0\}$	$\{s_1\}$	$\{s_1\}$	$\{s_0\}$	$\cdots$
	-2	-1	0	1	2	

$t = 0'$

			$\downarrow\{q_1^\bullet\}$	$\downarrow\{q_2^\bullet\}$		
$\cdots$	$\{s_0\}$	$\{s_0\}$	$\{s_0\}$	$\{s_1\}$	$\{s_0\}$	$\cdots$
	-2	-1	0	1	2	

$t = 0''$

			$\downarrow\{q_2^\bullet, q_3^\bullet\}$			
$\cdots$	$\{s_0\}$	$\{s_0\}$	$\{s_0\}$	$\{s_1\}$	$\{s_0\}$	$\cdots$
	-2	-1	0	1	2	

Figura 3.14: Computación de $\mathcal{M}(n)$

Para la máquina de Turing $\mathcal{M}''$, con la cinta dividida en tres pistas, que se construye para simular $\mathcal{M}$, la decripción instantánea inicial se representa en la figura 3.15.

			$\downarrow q_1$			
	s_{10}	s_{10}	s_{01}	s_{01}	s_{10}	
$\cdots$	$\varnothing$	\$	q_{100}	\$	$\varnothing$	$\cdots$
	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	
	-2	-1	0	1	2	

Figura 3.15: Simulación MTP $\mathcal{M}(n)$ - situación inicial

Los símbolos q_{ijk} representan subconjuntos de estados ($\{q_i, \dots, q_j\} \in P(Q)$), con la convención de que $i = 1$ si q_1 pertenece al conjunto o $i = 0$ en otro caso, $j = 1$ si q_2 pertenece al conjunto o $j = 0$ en otro caso y $k = 1$ si q_3 pertenece al conjunto o $K = 0$ en otro caso. De forma similar, los símbolos s_{ij} representan subconjuntos de símbolos ($\{s_i, \dots, s_j\} \in P(\Sigma)$), con la convención de que $i = 1$ si s_0 pertenece al conjunto o $i = 0$ en otro caso y $j = 1$ si s_1 pertenece al conjunto o $j = 0$ en otro. El símbolo \$ se utiliza como marcador de final.

Para el primer paso de computación, la situación de $\mathcal{M}''(n)$ luego del primer barrido se representa por la figura 3.16.

$\downarrow q_i$ (sobre la celda 2)

	s_{10}	s_{10}	s_{01}	s_{01}	s_{10}	
$\cdots$	$\varnothing$	\$	q_{100}	$\varnothing$	\$	$\cdots$
	$\varnothing$	$\varnothing$	$\varnothing$	q_{010}	$\varnothing$	
	-2	-1	0	1	2	

Figura 3.16: Simulación MTP $\mathcal{M}(n)$ - paso 1 - barrido 1

La situación de $\mathcal{M}''(n)$ luego del segundo barrido se representa por la figura 3.17.

$\downarrow q_j$ (sobre la celda -1)

	s_{10}	s_{10}	s_{01}	s_{01}	s_{10}	
$\cdots$	$\varnothing$	\$	q_{100}	$\varnothing$	\$	$\cdots$
	$\varnothing$	$\varnothing$	$\varnothing$	q_{010}	$\varnothing$	
	-2	-1	0	1	2	

Figura 3.17: Simulación MTP $\mathcal{M}(n)$ - paso 1 - barrido 2

La situación de $\mathcal{M}''(n)$ luego del tercer barrido se representa por la figura 3.18.

$\downarrow q_k$ (sobre la celda 2)

	s_{10}	s_{10}	s_{10}	s_{01}	s_{10}	
$\cdots$	$\varnothing$	\$	q_{100}	q_{010}	\$	$\cdots$
	$\varnothing$	$\varnothing$	$\varnothing$	q_{010}	$\varnothing$	
	-2	-1	0	1	2	

Figura 3.18: Simulación MTP $\mathcal{M}(n)$ - paso 1 - barrido 3

La situación de $\mathcal{M}''(n)$ luego del cuarto barrido se representa por la figura 3.19.

		$\downarrow q_l$				
	s_{10}	s_{10}	s_{10}	s_{01}	s_{10}	
$\cdots$	$\varnothing$	\$	q_{100}	q_{010}	\$	$\cdots$
	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	
	-2	-1	0	1	2	

Figura 3.19: Simulación MTP $\mathcal{M}(n)$ - paso 1 - barrido 4

Para el segundo paso de computación, la situación de $\mathcal{M}''(n)$ luego del primer barrido se representa por la figura 3.20.

					$\downarrow q_{i'}$	
	s_{10}	s_{10}	s_{10}	s_{01}	s_{10}	
$\cdots$	$\varnothing$	\$	q_{100}	q_{010}	\$	$\cdots$
	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	
	-2	-1	0	1	2	

Figura 3.20: Simulación MTP $\mathcal{M}(n)$ - paso 2 - barrido 1

La situación de $\mathcal{M}''(n)$ luego del segundo barrido se representa por la figura 3.21.

		$\downarrow q_{j'}$				
	s_{10}	s_{10}	s_{10}	s_{01}	s_{10}	
$\cdots$	$\varnothing$	\$	q_{100}	q_{010}	\$	$\cdots$
	$\varnothing$	$\varnothing$	q_{001}	$\varnothing$	$\varnothing$	
	-2	-1	0	1	2	

Figura 3.21: Simulación MTP $\mathcal{M}(n)$ - paso 2 - barrido 2

La situación de $\mathcal{M}''(n)$ luego del tercer barrido se representa por la figura 3.22.

					$\downarrow q_{k'}$	
	s_{10}	s_{10}	s_{10}	s_{01}	s_{10}	
$\cdots$	$\varnothing$	\$	q_{011}	$\varnothing$	\$	$\cdots$
	$\varnothing$	$\varnothing$	q_{001}	$\varnothing$	$\varnothing$	
	-2	-1	0	1	2	

Figura 3.22: Simulación MTP $\mathcal{M}(n)$ - paso 2 - barrido 3

La situación de $\mathcal{M}''(n)$ luego del cuarto barrido se representa por la figura 3.23.

		$\downarrow q_{l'}$				
	s_{10}	s_{10}	s_{10}	s_{01}	s_{10}	
$\cdots$	$\varnothing$	\$	q_{011}	$\varnothing$	\$	$\cdots$
	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	
	-2	-1	0	1	2	

Figura 3.23: Simulación MTP $\mathcal{M}(n)$ - paso 2 - barrido 4

No se definen las instrucciones para $\mathcal{M}''$, por tratarse de un trabajo muy engorroso y que no agrega mayor valor. Teniendo construida $\mathcal{M}''$, se puede construir $\mathcal{M}'$ siguiendo el procedimiento planteado en [17, p. 195].

Capítulo 4

Conclusiones

- Se definió un proceso de axiomatización del comportamiento de las MTDs, contruyendo teorías en lógica de predicados de primer orden clásica con igualdad (capítulo 1). Este procedimiento se obtuvo adicionando, al método de axiomatización definido por Boolos y Jeffreys, nuevos esquemas axiomáticos (sección 1.2). Se demostró que las teorías que se obtienen son válidas y completas, con respecto a una estructura matemática definida como semántica para las MTDs (sección 1.2). Con esto se alcanza el objetivo específico 1.

- Se definió un proceso de axiomatización del comportamiento de las MTNDs, modificando el procedimiento definido para las MTDs y se demostró que las teorías que se obtienen son válidas y completas, de forma similar a como se demostró validez y completud para el caso de las MTDs (capítulo 2). Con esto se alcanza el objetivo específico 2.

- Se generaron teorías contradictorias de máquinas de Turing, permitiendo que la máquina ejecutara varias instrucciones en un mismo instante de tiempo. Esta fué la base para la construcción de las MTPs (sección 3.1). Con esto se alcanza el objetivo específico 3.

- La axiomatización de las MTDs y la sustitución de la lógica subyacente por la lógica paraconsistente $C_1^=$, permitió construir una teoría axiomática paraconsistente de máquinas de Turing (sección 3.2), a partir de la cual se construyó un nuevo modelo de máquina de Turing, al cual se denominó máquina de Turing paraconsistente (MTP) (sección 3.3). Con esto se alcanza el objetivo específico 4.

- El modelo de máquinas de Turing paraconsistentes obtenido, de forma similar al modelo de MTND y muchos otros modelos de máquina de Turing (multicinta, con múltiples cabezas, etc), es una generalización a la definición original de máquina de Turing, en la cual se elimina la restricción de que la máquina sólo ejecuta una instrucción en cada instante de tiempo, restricción que se impone en todos los modelos existentes (o por lo menos los que conoce el autor), asegurando así la 'consistencia' en el proceso de computación. Visto de esta manera, el principal aporte del presente trabajo es, la construcción de un nuevo modelo de máquina de Turing en el cual se elimina una restricción que parecía inviolable.

- Se demostró que el modelo de máquinas de Turing paraconsistentes obtenido, no es un modelo de hipercomputación (sección 3.5). Aunque el resultado haya sido negativo, se logró alcanzar el objetivo específico 5.

- Teniendo en cuenta las infinitas lógicas paraconsistentes que existen y las diferentes posibilidades de modificación de la teoría de máquinas de Turing, el hecho de que el modelo de MTP obtenido no sea un modelo de hipercomputación, no indica que no se pueda construir un modelo de máquinas de Turing paraconsistentes que cumpla con esta característica. Sólo se cierra un camino, quedan múltiples caminos (quizas infinitos) por recorrer. Como sugerencia para el recorrido de estos nuevos caminos, se debe modificar la teoría (no sólo sustituir la lógica) buscando introducir comportamientos que no puedan ser simulados por una MTD, posiblemente estos comportamientos se puedan introducir ligando, de alguna manera, el comportamiento de la máquina a una característica que crezca de forma no controlada (quizas las contradicciones). El modelo de máquinas de Turing paraconsistentes obtenido puede resultar ser la base para construir un modelo de máquinas de Turing paraconsistentes que sea un modelo de hipercomputación.

Apéndice A

Máquinas de Turing

Una máquina de Turing es un modelo abstracto de máquina que cuenta con una cabeza de lectura/escritura y una cinta unidimensional infinita en ambos sentidos y dividida en casillas, cada una de las cuales tiene capacidad para contener un y sólo un símbolo (figura A.1). En todo instante discreto de tiempo, la cabeza está en un estado o configuración y posicionada sobre una casilla de la cinta leyendo el símbolo que hay escrito en ella. La máquina cuenta además con un conjunto de instrucciones que dependiendo del estado actual y del símbolo que se está leyendo, determina que símbolo escribir o que movimiento realizar (moverse una casilla a la izquierda o a la derecha). En un instante de tiempo sólo una instrucción puede ser ejecutada. La máquina se detiene cuando no encuentra una instrucción a ejecutar de acuerdo con el estado y símbolo actual sobre la cinta. El conjunto de estados y símbolos de lectura/escritura son finitos [13, 16, 26].

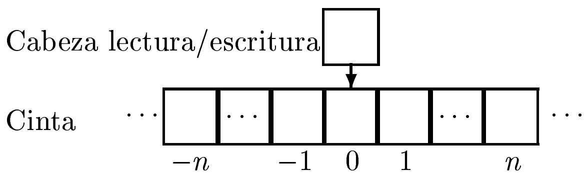

Figura A.1: Máquina de Turing

Definición A.1 (Máquina de Turing). *Una máquina de Turing se define formalmente mediante una estructura matemática* $\mathcal{M} =< Q, \Sigma, M, I >$ *donde:*

- $Q = \{q_1, q_2, \ldots, q_n\}$: *Conjunto finito de estados de la máquina* ($Q \neq \emptyset$).

- $\Sigma = \{s_0, s_1, \ldots, s_m\}$: *Conjunto finito de símbolos de entrada/salida. Por convención* s_0 *representa el símbolo vacío* ($\Sigma - \{s_0\} \neq \emptyset$).

- $M = \{R, L\}$: *Conjunto de movimientos (R: derecha, L: izquierda).*

- I: *Subconjunto de* $Q \times \Sigma \times \{\Sigma \cup M\} \times Q$. *Conjunto de instrucciones* $\{i_1, i_2, \ldots, i_k\}$ *donde cada* i_j *es una cuadrupla*[1] *de alguna de las formas:*

$$q_i s_j s_k q_l \tag{I}$$
$$q_i s_j R q_l \tag{II}$$
$$q_i s_j L q_l \tag{III}$$

El tipo de instrucción (I) *especifica que si la máquina se encuentra en el estado* q_i, *leyendo el símbolo* s_j, *escribe el símbolo* s_k *y pasa al estado* q_l. *El tipo de instrucción* (II) *especifica que si la máquina se encuentra en el estado* q_i *leyendo el símbolo* s_j, *se mueve una casilla a la derecha y pasa al estado* q_l. *El tipo de instrucción* (III) *es similar al segundo, sólo que el movimiento se realiza hacia la izquierda [13, 16].*

Las instrucciones de $\mathcal{M}$ *pueden ser representadas por medio de un grafo, donde existe un nodo por cada estado* $q_i \in Q$ *y se crea una flecha del nodo* q_i *al nodo* q_l, *etiquetado con* s_j*:acción (figura A.2), donde la acción puede ser* s_k, R *ó* L *dependiendo de si* $\mathcal{M}$ *tiene una instrucción del tipo* (I), (II) *ó* (III), *respectivamente.*

$$q_i \xrightarrow{s_j\text{:acción}} q_l$$

Figura A.2: Grafo para una instrucción

[1] En la definición original de Turing [26] las instrucciones se definen por medio de quíntuplas. Sin embargo, se puede demostrar que ambas definiciones son equivalentes. Se utiliza la definición por medio de cuadruplas [13] para facilitar el proceso de axiomatización presentado en el capítulo 1.

A.0.1. Máquinas de Turing deterministas y no deterministas

Se dice que una máquina de Turing es determinista si para cada combinación de estado y símbolo de entrada existe, a lo sumo, una instrucción a ser ejecutada. Por lo tanto, una máquina de Turing es no determinista si para al menos una combinación de estado y símbolo de entrada existe más de una instrucción a ser ejecutada, en cuyo caso, la máquina elige y ejecuta sólo una de ellas.

Definición A.2 (Máquina de Turing determinista (MTD)). *Una máquina de Turing* $\mathcal{M} =< Q, \Sigma, M, I >$ *es llamada determinista si* I *no contiene dos elementos (instrucciones) con los mismos dos símbolos iniciales. En este caso,* I *es una función de un subconjunto de* $Q \times \Sigma$ *en* $(\Sigma \cup M) \times Q$.

Definición A.3 (Máquina de Turing no determinista (MTND)). *Una máquina de Turing* $\mathcal{M} =< Q, \Sigma, M, I >$ *es llamada no determinista, si existen en* I *al menos dos elementos (instrucciones) con los mismos dos símbolos iniciales. En este caso,* I *no es una función de* $\Sigma \times Q$ *en* $\{\Sigma \cup M\} \times Q$*, sino que es una relación debido a que no se cumple la propiedad de unicidad. Una máquina de Turing no determinista, al encontrarse en una situación en la cual hay varias instrucciones que pueden ser ejecutadas, elige y ejecuta sólo una de ellas.*

En [26, p. 232] Turing hace la diferenciación entre 'máquinas automáticas' (*a-machines*) y 'máquinas de elección (o de opciones)' (*c-machines*). Indicando que una *a-machine* es aquella en la cual la acción de la máquina está *completamente* determinada por la configuración (estado actual y símbolo que está leyendo la máquina), mientras que en una *c-machine* la acción de la máquina está sólo *parcialmente* determinada por la configuración, cuando la máquina llega a una configuración 'ambigua', la máquina no puede continuar hasta que un operador externo realize una elección. Las MTDs corresponden entonces a las *a-machines* y las MTNDs a las *c-machines*, aunque en la definición de MTND no se hace explícito que la elección de la instrucción a ejecutar tenga que ser realizada por un ente externo.

En [20, p. 48], al definir el conjunto de instrucciones de la máquina de Turing, Odifreddi adiciona la condición de que éstas deben ser 'consistentes', indicando con esto que no pueden haber pares de instrucciones 'contradictorios', es decir, con las mismas *premisas* $q_i s_j$ y con diferentes *conclusiones*[2]. Lo que corresponde a la definición de MTD. Odifreddi define entonces las

[2] Odifreddi denomina *premisas* a los primeros dos símbolos de las instrucciones y *conclusiones* a los siguientes símbolos.

MTNDs eliminando de la definición de máquina de Turing la condición de 'consistencia' en las instucciones.

A.0.2. Computación en una máquina de Turing

Para definir formalmente lo que es una computación en una máquina de Turing, es necesario contar con una descripción completa de la ejecución de la máquina, es decir, es necesario conocer la evolución temporal de la máqina. Para ello, se requiere poder describir la situación (símbolos y posición sobre la cinta, y estado) de la máquina para un instante de tiempo t, y poder definir como esta situación va cambiando de un instante de tiempo a otro, de acuerdo con las instrucciones de la máquina. Por esto, antes de definir formalmente lo que es una computación en una máquina de Turing, se harán otras definiciones, siguiendo a [13, 16].

Definición A.4 (Sucesión). *Sea $\mathcal{M} =< Q, \Sigma, M, I >$ una máquina de Turing. Una sucesión es una palabra finita (posiblemente vacía) construida sobre al alfabeto $Q \cup \Sigma \cup M$.*

Definición A.5 (Descripción instantánea). *Sea $\mathcal{M} =< Q, \Sigma, M, I >$ una máquina de Turing. Una descripción instantánea α de $\mathcal{M}$ es una sucesión tal que:*

- *α no contiene ningún símbolo $m \in M$.*
- *α contiene exactamente un símbolo $q_i \in Q$.*
- *El símbolo q_i en α no es el último símbolo de izquierda a derecha.*
- *Todos los s_i que ocurren en α pertenecen a Σ.*

Por medio de una descripción instantánea se pueden describir todos los elementos de la máquina de Turing $\mathcal{M}$ en el instante de tiempo t. Suponiendo que $\mathcal{M}$ en el instante de tiempo t se encuentra en la situación representada en la figura A.3, la descripción instantánea sería $\alpha_t \equiv s_k \dots s_l q_i s_j s_n \dots s_m$.

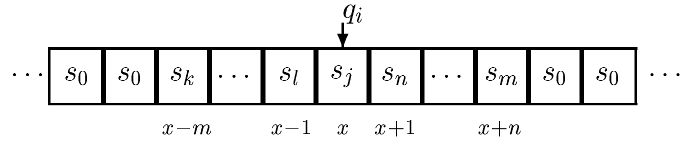

Figura A.3: Situación máquina $\mathcal{M}$ en el tiempo t

Definición A.6 (Sucesión de cinta). *Una sucesión de cinta es una sucesión que consta únicamente de símbolos* $s_i \in \Sigma$.

Definición A.7 (Cambio descripción instantánea). *Sea* $\mathcal{M}$ *una máquina de Turing y sean* α *y* β *descripciones instantáneas de* $\mathcal{M}$. *El cambio de la descripción instantánea* α *a la descripción instantánea* β *se representa por* $\alpha \xrightarrow{\mathcal{M}} \beta$, *lo cual significa que alguna de las siguientes condiciones se satisface:*

- *Existen sucesiones de cinta* P *y* Q *(posiblemente vacías) tales que:*
  $\alpha \equiv Pq_is_jQ$,
  $\beta \equiv Pq_ls_kQ$
  y $\mathcal{M}$ *contiene una instrucción del tipo* (I).

- *Existen sucesiones de cinta* P *y* Q *(posiblemente vacías) tales que:*
  $\alpha \equiv Pq_is_js_kQ$,
  $\beta \equiv Ps_jq_ls_kQ$
  y $\mathcal{M}$ *contiene una instrucción del tipo* (II).

- *Existe sucesión de cinta* P *(posiblemente vacía) tal que:*
  $\alpha \equiv Pq_is_j$,
  $\beta \equiv Ps_jq_ls_0$
  y $\mathcal{M}$ *contiene una instrucción del tipo* (II).

- *Existen sucesiones de cinta* P *y* Q *(posiblemente vacías) tales que:*
  $\alpha \equiv Ps_kq_is_jQ$,
  $\beta \equiv Pq_ls_ks_jQ$
  y $\mathcal{M}$ *contiene una instrucción del tipo* (III).

- *Existe sucesión de cinta* Q *(posiblemente vacía) tal que:*
  $\alpha \equiv q_is_jQ$,
  $\beta \equiv q_ls_0s_jQ$
  y $\mathcal{M}$ *contiene una instrucción del tipo* (III).

Definición A.8 (Descripción instantánea terminal). *Sea* $\mathcal{M}$ *un máquina de Turing. Una descripción instantánea* α *es llamada una descripción instantánea terminal, con respecto a* $\mathcal{M}$, *si no existe una descripción instantánea* β *tal que* $\alpha \xrightarrow{\mathcal{M}} \beta$.

Definición A.9 (Computación de una máquina de Turing). *Una computación de una máquina de Turing* $\mathcal{M}$ *es una secuencia finita de descripciones instantáneas* $\alpha_0, \alpha_1, \ldots, \alpha_n$, *tal que* $\alpha_i \xrightarrow{\mathcal{M}} \alpha_{i+1}$, *para* $0 \leq$

$i < n-1$, y α_n es una descripción instantánea terminal con respecto a $\mathcal{M}$. En este caso se escribe $\alpha_n = Res_{\mathcal{M}}(\alpha_0)$ y se denomina a α_n el resultado de α_0 con respecto a $\mathcal{M}$.

Ejemplo A.1. *Sea $\mathcal{M} =< Q, \Sigma, M, I >$ una máquina de Turing con $Q = \{q_1, q_2\}$, $\Sigma = \{s_0, s_1\}$ y $I = \{i_1, i_2\}$, donde $i_1 = q_1 s_1 R q_1$ y $i_2 = q_1 s_0 s_1 q_2$ (figura A.4).*

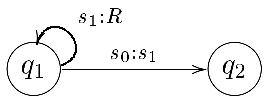

Figura A.4: Grafo máquina $\mathcal{M}$

La secuencia de descripciones instantáneas $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ donde:

$$\begin{aligned}
\alpha_0 &\equiv q_1 s_1 s_1 \\
\alpha_1 &\equiv s_1 q_1 s_1 \\
\alpha_2 &\equiv s_1 s_1 q_1 s_0 \\
\alpha_3 &\equiv s_1 s_1 q_2 s_1
\end{aligned}$$

es una computación de $\mathcal{M}$, puesto que $\alpha_0 \xrightarrow{\mathcal{M}} \alpha_1 \xrightarrow{\mathcal{M}} \alpha_2 \xrightarrow{\mathcal{M}} \alpha_3$ y α_3 es una descripción instantánea terminal con respecto a $\mathcal{M}$.

Apéndice B

Hipercomputación

Alan Turing, al proponer su modelo teórico de máquina de computación [26], ahora denominado 'máquina de Turing', demuestra que dicho modelo tiene sus límites. En forma particular, demuestra que no se puede construir una máquina de Turing $\mathcal{M}$ que, tomando como entrada la descripción de una máquina de Turing arbitraria $\mathcal{M}'$ (en una codificación determinada) y un dato de entrada n, determine si $\mathcal{M}'$ con la entrada n se detiene o no [26, secc. 8]. Este problema es conocido como el problema de la parada o *halting problem*. Este resultado lo utiliza Turing para ofrecer una respuesta negativa al '*Entscheidungsproblem*' o 'problema de la decisión', planteado por David Hilbert. Para el caso particular de la lógica de predicados de primer orden, el problema de la decisión plantea la pregunta ¿existe un procedimiento general para, dada una fórmula arbitraria α, determinar si α es un teorema de la lógica de predicados de primer orden? [26, secc. 11].

El término 'hipercomputación' es introducido por Jack Copeland y Diane Proudfoot, para designar a los modelos de computación que permiten computar objetos (funciones, números, etc.) que no pueden ser computados por una máquina de Turing (Turing-computable) [8, 23].

"[Los modelos de hipercomputación] incialmente fueron referenciados como '*super Turing computation*', '*computing beyond the Turing limit*', '*breaking the Turing barrier*', etc; esta referencia es errónea debido a que el mismo Alan Turing propuso uno de los primeros de tales modelos [(máquina de Turing con oráculo)]" [23, p. 1]. Estas referencias erróneas surgen debido a malas interpretaciones acerca de la tesis de Church-Turing, que usualmente es entendida como la afirmación de que "cualquier procedimiento que pueda ser computado, puede ser computado por una máquina de Turing". Sin embargo, lo que en realidad afirma la tesis de Church-Turing es:

> "Any procedure that can be carried out by an idealized human clerk working mechanically with paper and pencil can also be carried out by a Turing machine" [9, p. 50].

Esta tesis impone límites sobre lo que un humano puede realizar, trabajando de forma mecánica con papel y lápiz ilimitados, pero no impone límites a lo que puede ser realizado por una máquina, en forma general [9].

En la actualidad existen múltiples modelos teóricos de hipercomputación[1], sin embargo, sigue siendo una pregunta abierta si estos modelos pueden o no ser construidos. Varios modelos de hipercomputación son construidos realizando modificaciones a la definición de máquina de Turing, otros son obtenidos de diferentes teorías. Con el objetivo de ofrecer ilustraciones, se presentan brevemente algunos de estos modelos:

B.1. Máquinas de Turing con oráculo

El mismo Alan Turing introduce, en su tesis doctoral [27], el modelo de máquina de Turing con oráculo (*o-machine*), con las siguientes palabras:

> "Let us suppose that we are supplied with some unspecified means of solving number-theoretic problems; a kind of oracle as it were. We shall not go any further into the nature of this oracle apart from saying that it cannot be a machine. With the help of the oracle we could form a new kind of machine (call them *o-machines*), having as one of its fundamental processes that of solving a given number-theoretic problem" [27, p. 173].

Una *o-machine* es una máquina de Turing a la cual se le adiciona un nuevo componente (oráculo), al cual la máquina puede realizar preguntas, obteniendo respuestas binarias. De este modo, si el oráculo ofrece respuestas a preguntas que no pueden ser computadas por una máquina de Turing (el *halting problem* por ejemplo) la *o-machine* puede computar objetos no Turing computables. Infortunadamente, Turing no presenta, ni siquiera ofrece una idea, de como dichos oráculos pueden ser construidos [9, 23].

B.2. Máquinas de Turing acopladas

Las máquinas de Turing no aceptan entradas durante el proceso de computación, el dato de entrada es una secuencia finita de símbolos que son

[1]En [7] se presenta una revisión histórica de diferentes modelos de hipercomputación.

ingresados antes de que la máquina comience a operar. Una máquina de Turing acoplada es el resultado de adicionar, a una máquina de Turing, un canal de entrada que permite que la máquina, durante el proceso de computación, reciba entradas desde el exterior. El poder computacional de una máquina de Turing acoplada depende de la naturaleza de las entradas que recibe del exterior. Para que la máquina acoplada sea un modelo de hipercomputación se requiere que la secuencia de entradas sea infinita y no computable. Una máquina de Turing acoplada nunca se detiene, como es el caso de un controlador automático de tráfico aéreo [9].

B.3. Redes neuronales recurrentes análogas

Los modelos de redes neuronales son clasificados, de acuerdo a la forma como están conectadas sus neuronas, en redes *feedforward* (de alimentación hacia adelante o por niveles) y redes *feedback* (de alimentación hacia atrás o recurrentes). En las redes *feedforward* no se permiten ciclos en los grafos de interconexión de las neuronas, mientras que las redes *feedback* si los permiten. Las redes neuronales recurrentes análogas (ARNNs) son un modelo de red neuronal, del tipo *feedback*, que permite que los pesos asociados a las conexiones entre las neuronas sean números reales. Las ARNNs fueron propuestas por Hava Siegelman y Eduardo Sontag, quienes demostraron que: al restringir los pesos de las conexiones entre las neuronas a los números enteros, las ARNNs son equivalentes a los autómatas de estado finito. Al restringir los pesos de las conexiones entre las neuronas a los números racionales, las ARNNs son equivalentes a las máquinas de Turing. Pero si se permite que los pesos de las conexiones entre las neuronas sean números reales (con un número real no Turing-computable), las ARNNs son un modelo de hipercomputación [24, 1].

Apéndice C

Lógicas paraconsistentes

Normalmente los sistemas de lógica se estudian desde dos puntos de vista diferentes pero íntimamente relacionados, el enfoque sintáctico y el enfoque semántico. Para ofrecer una breve definición de lo que es un lógica paraconsistente se seguirá la opción sintáctica. Desde este punto de vista, el primer paso para construir un sistema de lógica consiste es definir un lenguaje o conjunto de símbolos primitivos $\mathcal{L}$, que se utilizarán en el *discurso lógico*. A partir de los símbolos del lenguaje, mediante la concatenación (unión en forma de cadena) de ellos, bajo algunas reglas de formación establecidas, se construye lo que se denomina el conjunto de fórmulas o expresiones bien formadas $\mathcal{F}$ de la lógica. Un sistema de lógica L se construye estableciendo un subconjuto de fórmulas como el conjunto de axiomas A de la lógica y definiendo una o más reglas de deducción que permiten pasar lícitamente de una fórmula a otra.

Una demostración de una fórmula θ en una lógica L es una sucesión finita de fórmulas $\alpha_1, \alpha_2, \ldots, \alpha_n$ de L tales que $\alpha_n = \theta$ y α_i es o bien un axioma ($\alpha_i \in A$) o es el resultado de aplicar una regla de deducción a fórmulas anteriores en la secuencia.

Una teoría deductiva (o axiomática) Σ es un conjunto de fórmulas (axiomas o postulados) que tienen subyacente una lógica particular L. Una demostración de una fórmula θ en una teoría Σ es una secuencia finita de fórmulas $\alpha_1, \alpha_2, \ldots, \alpha_n$ tales que $\alpha_n = \theta$ y α_i es o bien un axioma de la teoría ($\alpha_i \in \Sigma$) o es un axioma de la lógica L o es el resultado de aplicar una regla de deducción de L a fórmulas anteriores en la secuencia. El hecho de que la fórmula θ sea demostrable en la teoría Σ bajo la lógica L se simboliza por $\Sigma \vdash_L \theta$.

Dada una lógica L cuyo lenguaje incluya un símbolo de negación $\neg$, una teoría Σ, cuya lógica subyacente sea L, se denomina *contradictoria* si $\Sigma \vdash_L \theta$ y $\Sigma \vdash_L \neg\theta$ (para alguna fórmula θ), se denomina *trivial* si $\Sigma \vdash_L \theta$ para toda fórmula θ y se denomina *explosiva* si $\Sigma \cup \{\theta, \neg\theta\}$ es trivial (para cualquier fórmula θ). La lógica subyacente L, por su lado, se denomina *contradictoria*, *trivial* o *explosiva* si todas sus teorías son, respectivamente, *contradictorias*, *triviales* o *explosivas* [5, p. 4].

La lógica clásica (LC) tiene la propiedad de ser *explosiva*. En LC no hay distinción entre teoría *contradictoria* y teoría *trivial*. Las lógicas paraconsistentes, por el contrario, son lógicas que se distinguen por ser no *explosivas*, y por lo tanto, por servir como lógicas subyacentes a teorías *contradictorias* no *triviales* [5, 12].

Existen sistemas de lógica paraconsistente propuestos desde diferentes ópticas y persiguiendo motivaciones filosóficas diferentes[1]. Sin embargo, sólo se presenta una breve descripción de las jerarquías C_n, C_n^* y $C_n^=$ propuestas por Newton da Costa [10, 11], para ofrecer un bosquejo general de $C_1^=$, que es el sistema que interesa en el presente trabajo, por ser el elegido para construir el modelo de máquina de Turing paraconsistente.

C.0.1. Cálculos C_n, C_1^* y $C_n^=$

En [11] Newton da Costa propone una jerarquía de cálculos proposicionales paraconsistentes siguiendo las condiciones:

1. El principio de no contradicción, en la forma $\neg(A \wedge \neg A)$, no debe ser un esquema válido.

2. De dos fórmulas contradictorias, A y $\neg A$, no debe ser posible en forma general deducir una fórmula arbitraria B.

3. Debe ser posible extender de estos cálculos al cálculo de predicados (con o sin igualdad) correspondiente.

4. Estos cálculos deben contener la mayor parte de los esquemas y reglas del cálculo proposicional clásico, que no interfieran con las condiciones anteriores.

A esta jerarquía de cálculos proposicionales la denomina C_n, $1 \leq n \leq \omega$.

[1] En [19] se presenta un análisis cuidadoso de las diferentes propuestas de lógicas paraconsistentes.

El cálculo proposicional C_1 está constituido por los axiomas:

$$A \to (B \to A), \tag{A1}$$
$$(A \to B) \to ((A \to (B \to C)) \to (A \to C)), \tag{A2}$$
$$(A \wedge B) \to A, \tag{A3}$$
$$(A \wedge B) \to B, \tag{A4}$$
$$A \to (B \to (A \wedge B)), \tag{A5}$$
$$A \to (A \vee B), \tag{A6}$$
$$B \to (A \vee B), \tag{A7}$$
$$(A \to C) \to ((B \to C) \to ((A \vee B) \to C)), \tag{A8}$$
$$A \vee \neg A, \tag{A9}$$
$$\neg\neg A \to A, \tag{A10}$$
$$B^\circ \to ((A \to B) \to ((A \to \neg B) \to \neg A)), \tag{A11}$$
$$(A^\circ \wedge B^\circ) \to (A \wedge B)^\circ, \tag{A12}$$
$$(A^\circ \wedge B^\circ) \to (A \vee B)^\circ, \tag{A13}$$
$$(A^\circ \wedge B^\circ) \to (A \to B)^\circ. \tag{A14}$$

Donde $A^\circ \equiv \neg(A \wedge \neg A)$ simboliza el 'buen comportamiento' de la fórmula A, lo que indica que A no es contradictoria y se comporta 'clásicamente'. La única regla de deducción de C_1 es el *modus ponens* (MP)[2].

Los axiomas (A1) a (A8) constituyen la base de la lógica positiva de Hilbert-Bernays. El axioma (A9) es el principio del 'tercero excluido'. El axioma (A10) establece la eliminación de la doble negación. El axioma (A11) asegura que se pueda aplicar 'reducción al absurdo' cuando el consecuente sea 'bien comportado' y los axiomas (A12) a (A14) aseguran que el 'buen comportamiento' se propague a las fórmulas compuestas.

Algunos teoremas importantes que se prueban en C_1 son [10]:

Teorema C.1. *En C_1 todos los esquemas y reglas de deducción del cálculo proposicional positivo clásico son válidos.*

Teorema C.2. *Si Σ es un conjunto de fórmulas, A es una fórmula y A_1, A_2, ..., A_n son las 'componentes primas'[3] de las fórmulas de $\Sigma \cup \{A\}$, entonces, $\Sigma \vdash_{LC} A$ si y sólo si $\Sigma \cup \{A_1^\circ, A_2^\circ, \ldots, A_n^\circ\} \vdash_{C_1} A$.*

[2] En el apéndice D se especifica el *modus ponens*.

[3] El término 'componente prima' se toma en el sentido en el que es definido en [18, p. 111].

Teorema C.3. *C_1 es finitamente trivializable*[4].

En C_1 no son válidos, en general, los esquemas de contraposición, introducción de la doble negación y reducción al absurdo. Estos esquemas sólo son válidos bajo ciertas condiciones de 'buen comportamiento'.

En C_1 se puede definir una 'negación fuerte', $\neg * A \equiv \neg A \wedge A^\circ$, que cuenta con las mismas propiedades de la negación clásica.

En [14, p. 44-45] también se muestra que:

Teorema C.4. *En C_1 es válido el teorema de deducción.*

Teorema C.5. *El teorema de sustitución de equivalentes no es válido en C_1.*

Los cálculos C_n, $1 < n < \omega$, se obtienen modificando en C_1 los axiomas (A11) a (A14) así:

$$B^{(n)} \to ((A \to B) \to ((A \to \neg B) \to \neg A)), \tag{A11'}$$

$$(A^{(n)} \wedge B^{(n)}) \to (A \wedge B)^{(n)}, \tag{A12'}$$

$$(A^{(n)} \wedge B^{(n)}) \to (A \vee B)^{(n)}, \tag{A13'}$$

$$(A^{(n)} \wedge B^{(n)}) \to (A \to B)^{(n)}. \tag{A14'}$$

Donde $A^{(n)} \equiv A^1 \wedge A^2 \wedge \cdots \wedge A^n$ y $A^n \equiv A^{\overbrace{\circ \cdots \circ}^{n \text{ veces}}}$. Para los cálculos C_n se prueban teoremas similares a los de C_1, modificando A° por $A^{(n)}$.

El cálculo C_ω se obtiene eliminando de C_1 los axiomas (A11) a (A14). El cálculo C_ω no es finitamente trivializable.

En la jerarquía de cálculos C_n, cada cálculo C_{n+1} es estríctamente más débil que C_n.

En [14] se presentan diferentes semánticas y métodos de decisión para los cálculos de la jerarquía C_n.

Los cálculos C_n^* son cálculos de predicados de primer orden paraconsistentes, construidos a partir de los cálculos C_n.

El cálculo C_1^* es el resultado de adicionar a C_1 las reglas y axiomas que define Kleene para el manejo de los cuantificadores (con las restricciones

---

[4]Se dice que un sistema no trivial S es finitamente trivializable si existe una fórmula (no un esquema) α tal que, adicionando α a S como un nuevo axioma, el sistema resultante es trivial [11, p. 500].

usuales) [18, p. 82]:

$$\frac{C \to A(x)}{C \to \forall x A(x)}, \tag{I}$$

$$\forall x A(x) \to A(t), \tag{II}$$

$$A(t) \to \exists x A(x), \tag{III}$$

$$\frac{A(x) \to C}{\exists x A(x) \to C}. \tag{IV}$$

dos axiomas para poder utilizar el símbolo de 'buen comportamiento' en los predicados:

$$\forall x (A(x))^{\circ} \to (\forall x A(x))^{\circ}, \tag{V}$$

$$\forall x (A(x))^{\circ} \to (\exists x A(x))^{\circ}. \tag{VI}$$

Y la regla[5]:

> Si A y B son 'fórmulas congruentes', o una puede ser obtenida de la otra mediante eliminación de cuantificadores vacíos, entonces $A \leftrightarrow B$ es un axioma. (VII)

Los cálculos C_n^*, $n > 1$, se obtienen modificando en C_1^* los axiomas (V) y (VI) por:

$$\forall x (A(x))^{(n)} \to (\forall x A(x))^{(n)}, \tag{V'}$$

$$\forall x (A(x))^{(n)} \to (\exists x A(x))^{(n)}. \tag{VI'}$$

Algunos teoremas importantes de los cálculos C_n^* son:

Teorema C.6. *Si Σ es un conjunto de fórmulas, A es una fórmula y A_1, A_2, $\ldots$, A_n son las 'componentes primas' de las fórmulas de $\Sigma \cup \{A\}$, entonces, $\Sigma \vdash_{C_0^*} A$ si y sólo si $\Sigma \cup \{A_1^{(n)}, A_2^{(n)}, \ldots, A_n^{(n)}\} \vdash_{C_n^*} A$. Donde C_0^* representa el cálculo proposicional de primer orden clásico.*

Teorema C.7. *Si A es una fórmula que contiene sólo letras de predicados sin variables, entonces $\vdash_{C_n} A$ si y sólo si $\vdash_{C_n^*} A$.*

Teorema C.8. *Los cálculos C_n^* son indecidibles.*

[5]El término 'fórmula congruente' es tomado en el sentido en el que es definido en [18, p. 153].

Los cálculos $C_n^=$ son cálculos de predicados de primer orden con igualdad paraconsistentes, construidos a partir de los cálculos C_n^*, adicionando los axiomas que define Kleene para la igualdad [18, p. 399]:

$$x = x, \tag{VIII}$$

$$(x = y) \to (A(x) \to A(y)). \tag{IX}$$

El axioma IX está sujeto a las restricciones usuales.

Apéndice D

Reglas de deducción

Gentzen, en su sistema de deducción natural [15], propone 8 reglas de deducción básicas para el cálculo proposicional clásico. Para cada conector ($\neg, \wedge, \vee, \rightarrow$) define una regla de 'introducción' y una de 'eliminación', estas reglas se presentan en la tabla D.1.

Las 8 reglas básicas son suficientes para realizar cualquier deducción en el cálculo proposicional clásico. Sin embargo, para simplificar las pruebas, normalmente se usan reglas derivadas. Algunas de las reglas derivadas se presentan en la tabla D.2.

Para el cálculo de predicados se adicionan reglas que permiten introducir y eliminar cuantificadores. Estas reglas se presentan en la tabla D.3.

Reglas de introducción	Reglas de eliminación
Implicación ($\to$)	
II: Teorema de deducción (TD)	EI: Modus ponens (MP)
$\dfrac{\begin{array}{l} A \\ \vdots \\ B \end{array}}{A \to B}$	$\dfrac{\begin{array}{l} A \to B \\ A \end{array}}{B}$
Conjunción ($\wedge$)	
IC: Conjunción (CONJ)	EC: Simplificación (SIMP)
$\dfrac{\begin{array}{c} A \\ B \end{array}}{A \wedge B}$	$\dfrac{A \wedge B}{A}$ $\qquad$ $\dfrac{A \wedge B}{B}$
Disyunción ($\vee$)	
ID: Adición (ADIC)	ED: Prueba por caso (CASOS)
$\dfrac{A}{A \vee B}$ $\qquad$ $\dfrac{B}{A \vee B}$	$\dfrac{\begin{array}{l} A \vee B \\ A \\ \vdots \\ C \\ B \\ \vdots \\ C \end{array}}{C}$
Negación ($\neg$)	
IN: Reducción al absurdo (ABS)	EN: Doble negación (DN)
$\dfrac{\begin{array}{l} A \\ \vdots \\ B \wedge \neg B \end{array}}{\neg A}$	$\dfrac{\neg\neg A}{A}$

Tabla D.1: Reglas básicas para el cálculo proposicional clásico

Silogismo hipotético (SH)	Silogismo disyuntivo (SD)
$$\frac{\begin{array}{c}A \to B\\ B \to C\end{array}}{A \to C}$$	$$\frac{\begin{array}{l}A \vee B\\ \neg A\end{array}}{B}$$

Conmutatividad conjunción y disyunción (CONM)	
$$\frac{A \wedge B}{B \wedge A}$$	$$\frac{A \vee B}{B \vee A}$$

Contraposición (CTRP)	Modus tollens (MT)
$$\frac{A \to B}{\neg B \to \neg A}$$	$$\frac{\begin{array}{l}A \to B\\ \neg B\end{array}}{\neg A}$$

Tabla D.2: Reglas derivadas para el cálculo proposicional clásico

Reglas de introducción	Reglas de eliminación
Generalización universal (GU)	Instanciación universal (IU)
$\dfrac{Pa}{\forall xPx}$ Condición: 'a' no debe ocurrir en ningún supuesto previo no cancelado.	$\dfrac{\forall xPx}{Pa}$
Generalización existencial (GE)	Instanciación existencial (IE)
$\dfrac{Pa}{\exists xPx}$	$\dfrac{\begin{array}{c} \exists xPx \\ \left[\begin{array}{c} Pa \\ \vdots \\ A \end{array}\right. \end{array}}{A}$ Condición: 'a' no debe ocurrir en $\exists xPx$, ni en A, ni en ningún supuesto previo no cancelado.

Tabla D.3: Reglas para el cálculo de predicados clásico

Bibliografía

[1] Juan C. Agudelo. Computabilidad de las redes neuronales recurrentes análogas. Tesis de Maestría, Universidad EAFIT (2001). Eprint: `sigma.eafit.edu.co:90/~asicard/archivos/arnn.tar.gz`.

[2] George Boolos y Richard Jeffrey. "Computability and logic". Cambridge University Press, 3a. edición (1989).

[3] Walter Carnielli, Marcelo Coniglio y João Marcos. Logics of formal inconsistency. Eprint: `http://www.cs.math.ist.utl.pt/ftp/pub/MarcosJ/03-CCM-lfi.pdf`. Próximo a ser publicado en el *Handbook of Philosophical Logic*, segunda edición, volumen 12 (2003).

[4] Walter A. Carnielli, Marcello E. Coniglio y Itala M. L. D'Ottaviano, editores. "Paraconsistency. The logical way to the inconsistent". New York: Marcel Dekker (2002).

[5] Walter A. Carnielli y João Marcos. A taxonomy of C-systems. *En: [4]* páginas 1–94 (2002). Eprint: `ftp://logica.cle.unicamp.br/pub/e-prints/Taxonomy.pdf`.

[6] Walter A. Carnielli, João Marcos y Sandra de Amo. Formal inconsistency and evolutionary databases. *Logic and logical philosophy* páginas 115–152 (2000). Eprint: `ftp://logica.cle.unicamp.br/pub/e-prints/Taxonomy.pdf`.

[7] Jack Copeland. Hipercomputation. *Minds and machines* **12**, 461–502 (2002).

[8] Jack Copeland y Diane Proudfoot. Un Alan Turing desconocido. *Investigación y Ciencia* páginas 15–19 (Junio 1999).

[9] Jack Copeland y Richard Sylvan. Beyond the universal Turing machine. *Australasian journal of philosophy* **77**, 44–66 (1999).

Eprint: www.phil.canterbury.ac.nz/philsite/people/jack/jack\protect\unhbox\voidb@x\kern.06em\vbox{\hrulewidth.3em}\protect\kern+.1667em\relaxcopeland.html.

[10] NEWTON C. A. DA COSTA. On the theory of inconsistent formal system. *Notre Dame Journal of Formal Logic* **XV**(4), 497–510 (octuber 1974).

[11] NEWTON C. A. DA COSTA. "Sistemas formais inconsistentes". Curitaba, Paraná, Brasil: Editora UFPR, 2a. edición (1993).

[12] NEWTON C. A. DA COSTA Y RENATO A. LEWIN. Lógica paraconsistente. En "Encliclopedia IberoAmericana de filosofía", tomo 7: Lógica, páginas 185–204. Madrid: Editorial Trotta, S.A. (1995).

[13] MARTIN DAVIS. "Computability and unsolvability". New York: Dover Publications, Inc. (1982).

[14] JOÃO MARCOS DE ALMEIDA. Semânticas de traduções possíveis. Tesis de Maestría, Universidade Estadual de Campinas, Instituto de Filosofia e Ciências Humanas (1999).

[15] MANUEL GARRIDO. "Lógica simbólica". Madrid: Editorial Tecnos, 3a edición (1997).

[16] RAÚL GÓMEZ Y ANDRÉS SICARD. "Informática teórica: elementos propedeúticos". Medellín: Fondo editorial Universidad EAFIT (2001). Eprint: sigma.eafit.edu.co:90/~asicard/archivos/parteInformaticaTeorica.ps.gz.

[17] DEAN KELLEY. "Teoría de autómatas y lenguajes formales". Prentice-Hall (UK) Ltd. (1995).

[18] STEPHEN KLEENE. "Introducción a la metamatemática". Colección: Estructura y Función. Madrid: Editorial Tecnos (1974).

[19] ANDRÉS BOBENRIETH MISERDA. "¿Inconsistencias, por qué no?" Santafé de Bogotá: Tercer Mundo Editores, División Gráfica (1996).

[20] PIERGIORGIO ODIFREDDI. "Classical recursion theory. The theory of functions and sets of natural numbers". Studies in logic and the foundations of mathematics, volume 125. Amsterdan: North-Holland (1989).

[21] ANDRÉS SICARD. Computación paraconsistente. Informe técnico, Universidad EAFIT (2001). Eprint: `sigma.eafit.edu.co:90/~asicard/archivos/proyectoCP.ps.gz`.

[22] ANDRÉS SICARD Y MARIO VÉLEZ. Prototipo de un modelo de computación cuántica continua. Informe técnico, Universidad EAFIT (2000). Eprint: `sigma.eafit.edu.co:90/~asicard/archivos/proyectoCCC.tar.gz`.

[23] ANDRÉS SICARD Y MARIO VÉLEZ. Hipercomputación: la próxima generación de la computación teórica. *Revista Universidad EAFIT* **123**, 47–51 (2001). En: Informe de investigación [22]. Eprint: `sigma.eafit.edu.co:90/~asicard/archivos/hipercomputacion.pdf.gz`.

[24] HAVA T. SIEGELMANN. "Neural networks and analog computation. Beyond the Turing limit". Progress in Theorical Computer Science. Boston: Birkhäuser (1999).

[25] RICHARD SYLVAN Y JACK COPELAND. Computability is logic-relative. *Philosophy Research Papers, University of Canterbury* **5**, 1–16 (november 1998).

[26] ALAN M. TURING. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc* páginas 230–265 (1936). A correction, ibid, vol 43. 1936-1937. págs. 544 - 546.

[27] ALAN M. TURING. Systems of logic based on ordinales. *P. London Math. Soc* **2**(45), 161–228 (1939).

[28] JOUKO VÄÄNÄANEN. A short course on finite model theory. Notes based on lectures first gave at the *Summer School of Logic, Language and Information* in Lisbon in August 1993 and then in the department of mathematics of the University of Helsinki in September 1994. Eprint: `http://www.math.helsinki.fi/~logic/people/jouko.vaananen/shortcourse.pdf`.