Vol. XII Nº 2 Dic. (2004)
Matemáticas: 37–51

Matemáticas:
Enseñanza Universitaria

Máquinas de Turing paraconsistentes: una posible definición

Juan C. Agudelo Andrés Sicard

Recibido Mar. 15, 2004 Aceptado Jul. 21, 2004

Resumen

Se define un método para axiomatizar las máquinas de Turing, mediante el cual, dada una máquina $\mathcal{M}$ y una entrada n, se construye una teoría en la lógica clásica de predicados de primer orden que da cuenta del comportamiento de $\mathcal{M}(n)$. Cuando dicho método es utilizado para axiomatizar máquinas de Turing no deterministas produce teorías inconsistentes y por lo tanto triviales, teniendo en cuenta que la lógica subyacente es la lógica clásica. Mediante la sustitución de la lógica clásica por la lógica paraconsistente $C_1^=$ se construye un nuevo modelo de computación el cual se denominó máquinas de Turing paraconsistentes. Si bien estas nuevas máquinas son una generalización de las máquinas de Turing (deterministas), tal como sucede con otras muchas generalizaciones, se demuestra que éstas son, desde el punto de vista de la computabilidad, equivalentes al modelo original.

Palabras y frases claves: Computabilidad, máquinas de Turing, lógica paraconsistente.

Abstract

A method is defined to axiomatize Turing machines, by means of which, given a machine $\mathcal{M}$ and an input n, a theory in classical first order predicate logic is built. This theory allows to represent the behavior of $\mathcal{M}(n)$. When this method is used to axiomatize non-deterministic Turing machines produces inconsistent and therefore trivial theories, because the underlying logic is the classical logic. By substituting the classical logic by the paraconsistent logic $C_1^=$ a new model of computation called paraconsistent Turing machines is built. Although these new Turing machines are a generalization of the (deterministic) Turing machines, it is proved that they are equivalent to the original model from the computability point of view.

Keywords: Computability, Turing machines, paraconsistent logic.

AMSC(2000): Primary: 03D10, Secondary: 03D60

1 Introducción

Para demostrar la indecibilidad de la lógica clásica de predicados de primer orden, Boolos y Jeffrey [3] definen un método para construir teorías axiomáticas que dan cuenta del comportamiento de las máquinas de Turing [16, 12, 7, 9, 10] (se seguirá haciendo referencia a este método como el método B-J). Dada una máquina de Turing $\mathcal{M}$ y una entrada n, mediante el método B-J se construye una teoría[1] $\Delta_{LC}(\mathcal{M}(n))$, en la cual se codifican las instrucciones y la situación inicial de $\mathcal{M}$ (estado inicial, posición sobre la cinta y entrada n). Además, se construye una fórmula H, en la cual se codifican las diferentes situaciones (estado y símbolo que lee) en que $\mathcal{M}$ se detiene, de tal modo que $\Delta_{LC}(\mathcal{M}(n)) \vdash H$ si y sólo si $\mathcal{M}$ con la secuencia de entrada n se detiene.

[1]Se adiciona el subíndice LC a Δ para resaltar el hecho de que la lógica subyacente a la teoría Δ es la lógica clásica.

Como es sabido que el problema de la parada de una máquina de Turing es indecidible [3, 10, 16] se concluye que determinar si $\Delta_{LC}(\mathcal{M}(n)) \vdash H$ también lo es. Por lo tanto, la lógica de predicados de primer orden clásica es indecidible[2].

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 [12, 7, 10] 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 teorías incompletas (sección 3).

Tomando como base el método B-J, se definen nuevos esquemas axiomáticos para que las teorías resultantes sean completas y válidas con respecto al comportamiento de las máquinas de Turing deterministas (MTDs), obteniéndose teorías $\Delta'_{LC}(\mathcal{M}(n))$. Al utilizar el método B-J, con los nuevos esquemas axiomáticos, para axiomatizar máquinas de Turing no deterministas (MTNDs) [8, 14], resultan teorías inconsistentes y por lo tanto triviales, teniendo en cuenta que la lógica subyacente es la lógica clásica [15, 4]. Mediante la sustitución de la lógica subyacente, por la lógica paraconsistente $C_1^{=}$ de da Costa [5, 6], se redefine el comportamiento de las MTNDs, obteniéndose un nuevo modelo de computación al cual se denomina máquinas de Turing paraconsistentes (MTPs). En este nuevo modelo, a diferencia de las MTNDs, una máquina puede ejecutar en forma simultánea múltiples instrucciones, dando lugar a multiplicidad de símbolos sobre las casillas de la cinta, y multiplicidad de estados y posiciones sobre la máquina en un instante de tiempo determinado.

La elección de $C_1^{=}$ como la lógica paraconsistente que se utiliza como lógica subyacente, es en cierto sentido arbitraria. En principio, para realizar un trabajo de esta naturaleza, se requiere de una lógica paraconsistente de predicados de primer orden con igualdad. Se escogió $C_1^{=}$ por ser una lógica que cumple con las condiciones descritas y por ser una de las más conocidas y estudiadas[3].

Las MTPs resultan ser una generalización del modelo original de máquina de Turing que, como las MTNDs y otras generalizaciones del modelo de máquina de Turing (con múltiples cintas, con múltiples cabezas, etc. [14, 11]), es computacionalmente equivalente al modelo original.

---

[2]Sea $\overline{\overline{A}}$ el cardinal del conjunto A. Como $\Delta_{LC}(\mathcal{M}(n))$ es una teoría finita, $\Delta_{LC}(\mathcal{M}(n)) \vdash H$ si y sólo si $\vdash (\wedge_{i=1}^{p}\delta_i) \rightarrow H$, donde δ_i es un axioma de $\Delta_{LC}(\mathcal{M}(n))$ y $p = \overline{\overline{\Delta_{LC}(\mathcal{M}(n))}}$. Por lo tanto, si se supone que la lógica clásica de predicados de primer orden es decidible, se tiene que $\vdash (\wedge_{i=1}^{p}\delta_i) \rightarrow H$ sería decidible y por lo tanto $\Delta_{LC}(\mathcal{M}(n)) \vdash H$ también lo sería. Lo que indicaría que el problema de la parada de una máquina de Turing sería decidible.

[3]La generalización del modelo propuesto utilizando la jerarquía de los *C-Systems* [4] se está realizando por uno de los autores.

2 Método de axiomatización B-J

Antes de describir el método B-J es conveniente aclarar que, en la definición de máquina de Turing utilizada, el conjunto de instrucciones de una máquina de Turing $\mathcal{M}$ es un conjunto finito I, donde cada instrucción $i_j \in I$ es una cuadrupla[4] de alguna de las formas:

$$q_i s_j s_k q_l \quad \text{(I)}$$
$$q_i s_j R q_l \quad \text{(II)}$$
$$q_i s_j L q_l \quad \text{(III)}$$

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.

Para definir el método de axiomatización de máquinas de Turing, Boolos y Jeffrey [3] 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 instrucciones, 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, suponen que la máquina es 'prendida' en el momento 0 y 'apagada' en el primer momento (si existe alguno) después 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[5].

Una vez establecidos los supuestos anteriores, 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 $\Delta_{LC}(\mathcal{M}(n))$, que da cuenta del comportamiento de la máquina de Turing $\mathcal{M}$ con la entrada n. En $\mathcal{L}$, los símbolos Q_i, S_j y $<$ son símbolos de predicado de aridad dos, el símbolo $'$ es un símbolo de función de aridad uno y el símbolo 0 es un símbolo de constante.

[4] En la definición original de Turing [16] 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 [7] para facilitar el proceso de axiomatización.

[5] Estos últimos supuestos no se ven reflejados en el proceso de axiomatización B-J, por lo tanto, en la siguiente sección se definen los esquemas axiomáticos necesarios para asegurar dichos supuestos.

Bajo la interpretación intensional $\mathcal{I}$ de las sentencias 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 axiomas para establecer las propiedades de los símbolos $<$ y $'$, se incluye un nuevo axioma para definir cada una de las instrucciones de $\mathcal{M}$ y un axioma para definir la situación inicial de $\mathcal{M}$.

Los axiomas para establecer las propiedades de los símbolos $<$ y $'$ son:

$$\forall z \exists x (z = x'), \tag{A1}$$

$$\forall z \forall x \forall y (((z = x') \wedge (z = y')) \rightarrow (x = y)), \tag{A2}$$

$$\forall x \forall y \forall z (((x < y) \wedge (y < z)) \rightarrow (x < z)), \tag{A3}$$

$$\forall x (x < x'), \tag{A4}$$

$$\forall x \forall y ((x < y) \rightarrow (x \neq y)). \tag{A5}$$

Por comodidad, $\Delta_{LC}^{-1} = \{A1, A2, A3, A4, A5\}$.

Bajo la convención de que m es el cardinal del alfabeto de entrada-salida de $\mathcal{M}$, los axiomas de las instrucciones $i_j \in I$ de $\mathcal{M}$ se construyen de acuerdo con los esquemas:

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

$$\forall t \forall x \Bigg(\Big(Q_i(t,x) \wedge S_j(t,x) \Big) \rightarrow \Big(Q_l(t',x) \wedge S_k(t',x) \wedge \forall y \Big((y \neq x) \rightarrow \Big(\bigwedge_{i=0}^{m} \big(S_i(t,y) \rightarrow S_i(t',y) \big) \Big) \Big) \Big) \Bigg). \tag{Aj (I)}$$

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

$$\forall t \forall x \Bigg(\Big(Q_i(t,x) \wedge S_j(t,x) \Big) \rightarrow \Big(Q_l(t',x') \wedge \forall y \Big(\bigwedge_{i=0}^{m} \big(S_i(t,y) \rightarrow S_i(t',y) \big) \Big) \Big) \Bigg). \tag{Aj (II)}$$

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

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

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 $n = s_{i,1}s_{i,2}\ldots s_{i,p}$, se construye de acuerdo con el esquema:

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

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

Ejemplo 1. *Sea $\mathcal{M}_1 =< Q, \Sigma, M, I >$ una máquina de Turing con conjunto de estados $Q = \{q_1, q_2, q_3\}$, alfabeto de entrada-salida $\Sigma = \{s_0, s_1\}$, conjunto de instrucciones $I = \{i_1, i_2, i_3\}$, donde $i_1 = q_1 s_0 s_0 q_2$, $i_2 = q_1 s_1 R q_2$ e $i_3 = q_2 s_1 L q_3$ (figura 1), y sea $n_1 = s_1 s_1$ la entrada para $\mathcal{M}_1$.*

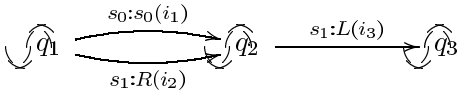

Figura 1: Grafo máquina $\mathcal{M}_1$

Utilizando el método B-J se obtiene la teoría $\Delta_{LC}(\mathcal{M}_1(n_1)) = \Delta_{LC}^{-1} \cup \{A6, A7, A8, A9\}$ donde:

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

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

$$\forall t \forall x \Bigg(\Big(Q_2(t,x') \wedge S_1(t,x') \Big) \rightarrow \Big(Q_3(t',x) \wedge$$

$$\forall y \Big(\big(S_0(t,y) \rightarrow S_0(t',y) \big) \wedge \big(S_1(t,y) \rightarrow S_1(t',y) \big) \Big) \Big) \Bigg), \quad \text{(A8)}$$

$$Q_1(0,0) \wedge S_1(0,0) \wedge S_1(0,0') \wedge \forall y \Big(\big((y \neq 0) \wedge (y \neq 0') \big) \rightarrow S_0(0,y) \Big). \quad \text{(A9)}$$

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

El método de axiomatización B-J produce teorías por medio de las cuales se puede demostrar, para cada instante de tiempo, el estado y posición de la máquina y los símbolos presentes en cada casilla de la cinta. Sin embargo, de acuerdo con la definición de máquina de Turing [16, 10], en cada instante de tiempo la máquina se encuentra en una única posición y estado y con un único símbolo sobre cada casilla de la cinta. Estas unicidades no son establecidas en las teorías construidas mediante el método de axiomatización B-J, lo que impide demostrar ciertas situaciones que se presentan en el proceso de computación de la máquina, como por ejemplo, si la máquina de Turing $\mathcal{M}$, en el instante de tiempo $t = 0$, se encuentra en el estado q_1 y en la posición $x = 0$, entonces $\mathcal{M}$ en ese instante de tiempo no se encuentra en el estado q_2 en ninguna posición, por lo tanto, $\Delta_{LC}(\mathcal{M}(n))$ debería probar la fórmula $\alpha \equiv \neg Q_2(0,0)$, pero de acuerdo a cómo se construye la teoría $\Delta_{LC}(\mathcal{M}(n)) \nvdash \neg Q_2(0,0)$. Por lo tanto, las teorías $\Delta_{LC}(\mathcal{M}(n))$ resultan ser incompletas con respecto al comportamiento de las MTDs.

Para que las teorías obtenidas sean completas, con respecto al comportamiento de las MTDs, se deben adicionar los axiomas para establecer los supuestos de que la máquina se encuentra 'apagada' antes del tiempo $t = 0$ y después del primer instante de tiempo $t = t_d$ (si existe alguno) en el que la máquina se detiene (sección 2), y los axiomas para establecer la unicidad de posición y estado de la máquina y de símbolos sobre cada casilla de la cinta, en cada instante de tiempo, obteniéndose teorías[6] $\Delta'_{LC}(\mathcal{M}(n))$ que son extensiones de las teorías $\Delta_{LC}(\mathcal{M}(n))$.

Bajo la convención de que n es el cardinal del conjunto de estados de $\mathcal{M}$ y m es el cardinal del alfabeto de entrada-salida de $\mathcal{M}$, el axioma para establecer que la máquina se encuentra 'apagada' antes del tiempo $t = 0$ se construye de acuerdo con el esquema:

$$\forall t \forall x \left((t < 0) \rightarrow \left(\left(\bigwedge_{i=1}^{n} \neg Q_i(t,x) \right) \wedge \left(\bigwedge_{j=0}^{m} \neg S_j(t,x) \right) \right) \right). \quad \text{(A(n+1))}$$

[6] La demostración de completitud de las teorías $\Delta'_{LC}(\mathcal{M}(n))$ con respecto al comportamiento de las MTDs se encuentra en [1] (teoremas 1.8, ..., 1.11).

El axioma para establecer que la máquina se encuentra apagada después del primer instante de tiempo $t = t_d$ (si existe alguno) en el que la máquina se detiene se construye de acuerdo con el esquema[7]:

$$\forall t \forall x \left(\left(\bigvee_{q_i s_j \ no\ inic.} \left(Q_i(t,x) \wedge S_j(t,x) \right) \right) \rightarrow \right.$$
$$\left. \forall u \forall y \left(t < u \rightarrow \left(\left(\bigwedge_{i=1}^{n} \neg Q_i(u,y) \right) \wedge \left(\bigwedge_{j=1}^{m} \neg S_j(u,y) \right) \right) \right) \right). \quad \text{(A(n+2))}$$

El axioma para establecer la unicidad de la posición de la máquina, en cada instante de tiempo, se construye de acuerdo con el esquema:

$$\forall t \forall x \left(\bigvee_{i=1}^{n} Q_i(t,x) \rightarrow \forall y \left(x \neq y \rightarrow \bigwedge_{j=1}^{n} \neg Q_j(t,y) \right) \right). \quad \text{(A(n+3))}$$

Para establecer la unicidad del estado de la máquina, en cada instante de tiempo, se debe construir un nuevo axioma por cada uno de los estados de la máquina, de acuerdo con el esquema:

$$\forall t \forall x \left(Q_i(t,x) \rightarrow \bigwedge_{i \neq j} \neg Q_j(t,x) \right). \quad \text{(A(n+4))}$$

Para establecer la unicidad de símbolos sobre cada casilla de la cinta, en cada instante de tiempo, se debe construir un nuevo axioma por cada uno de los símbolos de la máquina, de acuerdo con el esquema:

$$\forall t \forall x \left(S_i(t,x) \rightarrow \bigwedge_{i \neq j} \neg S_j(t,x) \right). \quad \text{(A(n+5))}$$

Ejemplo 2. *Para completar la teoría $\Delta_{LC}(\mathcal{M}_1(n_1))$, del ejemplo 1, se deben adicionar los axiomas $A10, \ldots, A17$, obteniendose la teoría $\Delta'_{LC}(\mathcal{M}_1(n_1)) =$*

[7]El subíndice '$q_i s_j$ *no inic.*' en la disyuntoria indica que sólo se tienen en cuenta las combinaciones de símbolos $q_i s_j$ que no aparezcan como símbolos iniciales en ninguna de las instrucciones de la máquina.

$\Delta_{LC}(\mathcal{M}_1(n_1)) \cup \{A10, \ldots, A17\}$, *donde:*

$$\forall t \forall x \Big((t < 0) \rightarrow \big(\neg Q_1(t,x) \wedge \neg Q_2(t,x) \wedge \neg Q_3(t,x) \wedge \neg S_0(t,x) \wedge \neg S_1(t,x) \big) \Big), \tag{A10}$$

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

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

$$\forall t \forall x (Q_1(t,x) \rightarrow (\neg Q_2(t,x) \wedge \neg Q_3(t,x))), \tag{A13}$$

$$\forall t \forall x (Q_2(t,x) \rightarrow (\neg Q_1(t,x) \wedge \neg Q_3(t,x))), \tag{A14}$$

$$\forall t \forall x (Q_3(t,x) \rightarrow (\neg Q_1(t,x) \wedge \neg Q_2(t,x))), \tag{A15}$$

$$\forall t \forall x \Big(\big(Q_1(t,x) \vee Q_2(t,x) \vee Q_3(t,x) \big) \rightarrow \forall y \Big(x \neq y \rightarrow \big(\neg Q_1(t,y) \wedge \neg Q_2(t,y) \vee \neg Q_3(t,x) \big) \Big) \Big), \tag{A16}$$

$$\forall t \forall x \Bigg(\Bigg(\Big(Q_2(t,x) \wedge S_0(t,x) \Big) \vee \Big(Q_3(t,x) \wedge S_0(t,x) \Big) \vee \Big(Q_3(t,x) \wedge S_1(t,x) \Big) \Bigg) \rightarrow \forall u \forall y \Bigg(t < u \rightarrow \Big(\neg Q_1(u,y) \wedge \neg Q_2(u,y) \wedge \neg Q_3(u,y) \wedge \neg S_0(u,y) \wedge \neg S_1(u,y) \Big) \Bigg) \Bigg). \tag{A17}$$

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

Al axiomatizar MTNDs por medio del método B-J, con los ajustes de la sección 3, se obtienen teorías $\Delta'_{LC}(\mathcal{M}(n))$ inconsistentes. Las inconsistencias en $\Delta'_{LC}(\mathcal{M}(n))$ surgen en cuanto al símbolo presente en una casilla e instante de tiempo determinado, o en cuanto al estado o posición de la máquina en un instante de tiempo determinado. El cuadro 4 resume las inconsistencias generadas de acuerdo con las diferentes combinaciones de instrucciones 'ambiguas', es decir, aquellas instrucciones cuyos dos símbolos iniciales son los mismos.

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

Ejemplo 3. *Sea $\mathcal{M}_2$ la MTND que se obtiene al adicionar a $\mathcal{M}_1$, del ejemplo 1, la instrucción $i_4 = q_1 s_1 s_0 q_1$ (figura 3).*

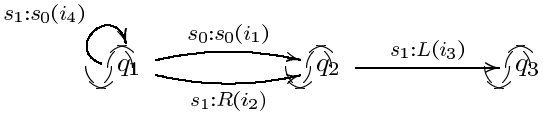

Figura 2: Grafo máquina $\mathcal{M}_1$

La teoría $\Delta'_{LC}(\mathcal{M}_2(n_1))$ se obtiene adicionando $\{A18\}$ a la teoría Δ'_{LC} $(\mathcal{M}_1(n_1))$ (ejemplo 2), donde:

$$\forall t \forall x \Big(\Big(Q_1(t,x) \wedge S_1(t,x) \Big) \rightarrow \Big(Q_1(t',x) \wedge S_0(t',x) \wedge$$

$$\forall y \Big((y \neq x) \rightarrow \big((S_0(t,y) \rightarrow S_0(t',y)) \wedge (S_1(t,y) \rightarrow S_1(t',y)) \big) \Big) \Big) \Big). \quad \text{(A18)}$$

Una de las contradicciones que se demuesta en $\Delta'_{LC}(\mathcal{M}_2(n_1))$ es $Q_1(0',0) \wedge \neg Q_1(0',0)$.

5 Máquinas de Turing paraconsistentes

En el contexto de la discusión de las relaciones entre sistematización cognoscitiva y lógica, Nicolas Rescher considera que "la consistencia es un requisito muy importante de la sistematización cognoscitiva, aunque entendida no tanto como 'principio constitutivo' a nivel de la descripción ontológica del mundo, sino como un 'principio regulativo' a nivel epistemológico" [2, 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" [2, 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" [2, 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 muestra en la sección 4, 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.

Mediante la sustitución de la lógica subyacente a las teorías $\Delta'_{LC}(\mathcal{M}(n))$ por la lógica[8] $C_1^{=}$, a partir de la interpretación de las consecuencias de las nuevas teorías $\Delta'_{C_1^{=}}(\mathcal{M}(n))$, se redefine lo que es una computación en una MTND[9], obteniendose un nuevo modelo de computación, al cual se denomina

[8]La lógica $C_1^{=}$ está basada en la lógica de predicados sin identidad C_1^{*}, la cual a su vez está basada en la lógica proposicional C_1. Para una presentación de esta lógicas se recomienda al lector las referencias [5, 6].

[9]Aunque es conocido que la regla de sustitución de equivalentes no es válida (en general) en $C_1^{=}$, esta característica no impide redefinir el comportamiento de las MTNDs a partir de las teorías $\Delta'_{C_1^{=}}(\mathcal{M}(n))$. Como $C_1^{=}$ se construye tomando como base la lógica positiva intuicionista, por lo menos la regla de sustitución de equivalentes se cumple en los contex-

máquina de Turing paraconsistente (MTP). Como consecuencia de sustituir la lógica subyacente a las teorías $\Delta'_{LC}(\mathcal{M}(n))$, se obtiene que las posibles contradicciones analizadas en la sección anterior también son demostrables en $\Delta'_{C_1^{=}}(\mathcal{M}(n))$, puesto que para demostrarlas sólo se requiere de instanciaciones universales, axiomas y reglas de inferencia pertenecientes al cálculo proposicional positivo clásico, los cuales son válidos en $C_1^{=}$. Sin embargo, como en las teorías $\Delta'_{C_1^{=}}(\mathcal{M}(n))$ en ninguna parte se establece el buen comportamiento[10] de $S_j(t,x)$ ni de $Q_i(t,x)$, las teorías $\Delta'_{C_1^{=}}(\mathcal{M}(n))$, a diferencia de las teorías $\Delta'_{LC}(\mathcal{M}(n))$, no se trivializan ante el surgimiento de estas contradicciones, permitiendo seguir el análisis del comportamiento de $\mathcal{M}(n)$ y definir una nueva noción de computación.

En la nueva noción de computación, al permitir instrucciones 'ambiguas' y no restringir la máquina a que sólo ejecute una única instrucción en cada instante de tiempo (como se hace en las máquinas de Turing no deterministas), la máquina puede ejecutar múltiples instrucciones en forma simultánea, dando lugar a que la máquina se encuentre en múltiples estados o posiciones, o que contenga multiples símbolos sobre algunas casillas de la cinta, en un determinado instante de tiempo. Dichas multiplicidades se ven reflejadas en las teorías $\Delta'_{C_1^{=}}(\mathcal{M}(n))$ por el surgimiento de contradicciones en los símbolos de predicado $Q_i(t,x)$ (estados o posiciones) o $S_j(t,x)$ (símbolos sobre casillas de la cinta)[11].

Ejemplo 4. *Sea $\Delta'_{C_1^{=}}(\mathcal{M}_2(n_1))$ el resultado de sustituir la lógica subyacente a la teoría $\Delta'_{LC}(\mathcal{M}_2(n_1))$, del ejemplo 3, por la lógica $C_1^{=}$. Tomando las consecuencias de la teoría $\Delta'_{C_1^{=}}(\mathcal{M}_2(n_1))$, 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 computación de la máquina paraconsistente $\mathcal{M}_2(n_1)$ por la figura 4.*

---

tos libres de negación, presentándose problemas sólo cuando se tienen negaciones [17, p. 586]. Para redefinir el comportamiento de las MTNDs, a partir de las teorías $\Delta'_{C_1^{=}}(\mathcal{M}(n))$, interesa sólo la interpretación de las fórmulas que puedan ser deducidas (negadas o no), sin importar que fórmulas equivalentes al ser negadas no lo puedan ser.

[10] En la lógica $C_1^{=}$ la expresión A° simboliza el 'buen comportamiento' de la expresión A, lo que indica que la expresión A se comporta 'clásicamente' (no puede ser contradictoria) [5]. En $C_1^{=}$ no se infiere el buen comportamiento de ninguna fórmula y los axiomas de $\Delta'_{C_1^{=}}(\mathcal{M}(n))$ tampoco permiten deducirla.

[11] En [1] se presenta una definición formal de computación en las MTPs, siguiendo la definición de computación por medio de cambios de descripciones instantáneas [13], adecuándola de acuerdo con las nuevas reglas que implica la simultaneidad en la ejecución de las instrucciones y la posibilidad de multiplicidad de estados, posiciones y símbolos sobre las casillas de la cinta.

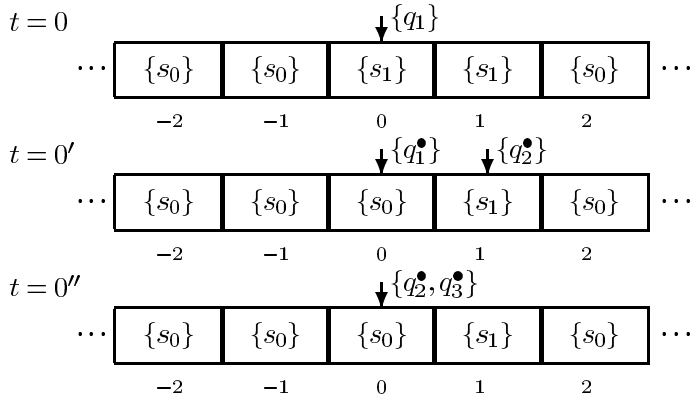

Figura 3: Computación MTP $\mathcal{M}_2(n_1)$

En este ejemplo, la máquina $\mathcal{M}_2$, en el tiempo $t = 0$, ejecuta las instrucciones i_2 e i_4 de forma simultánea, dando lugar a multiplicidad de posiciones ($x = 0$ y $x = 1$, apareciendo una nueva cabeza) y de estados (q_1 y q_2) en el tiempo $t = 0'$, provocando que los estados sean 'inconsistentes'. En en tiempo $t = 0'$, la máquina $\mathcal{M}_2$ ejecuta las instrucciones i_1 e i_3 de forma simultánea, permaneciendo con multiplicidad de estados (q_2 y q_3) en el tiempo $t = 0''$, pero en una sóla posición ($x = 0$, desapareciendo una cabeza).

6 Sobre el poder computacional de la máquina de Turing paraconsistente

Aunque las MTPs son una generalización al modelo original de máquina de Turing, igual que otras generalizaciones (MTNDs, máquinas de Turing multicintas, máquinas de Turing con múltiples cabezas, etc. [14, 11]), las MTPs son desde el punto de vista de la computabilidad equivalentes al modelo original de máquina de Turing, es decir, los conjuntos de funciones computables por ambos modelos son coexistentes.

Puesto que las MTDs son un caso particular de las MTPs, para demostrar la equivalencia entre los dos modelos, sólo es necesario lo siguiente.

Teorema 6. *Para toda MTP $\mathcal{M}$ se puede construir una MTD $\mathcal{M}'$ que simula el comportamiento de $\mathcal{M}$.*

Demostración. En [11] 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 [11, p. 198]. Siguiendo la demostración de equivalencia computacional entre este modelo de máquina de

Turing y el modelo original, se demuestra 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 (Aj (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 (Aj (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 [11, 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 $\mathcal{M}$. □

7 Conclusiones

La axiomatización de las MTDs y la sustitución de la lógica clásica de predicados de primer orden con identidad por la lógica paraconsistente $C_1^{=}$, permitió construir teorías axiomáticas paraconsistentes de máquinas de Turing, a partir de la cual se construyó un nuevo modelo de máquina de Turing, el cual se denominó máquina de Turing paraconsistente (MTP).

El modelo de las MTPs obtenido es una generalización de la definición original de máquina de Turing, en el 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 (conocidos por los autores), asegurando así la 'consistencia' en el proceso de computación.

Aunque en el nuevo modelo de máquina de Turing presentado se elimina

una restricción que parecía inviolable, éste resulta ser desde el punto de vista de la computabilidad, equivalente al modelo original. Esta equivalencia establece el poder computacional de las MTPs y además se constituye en una confirmación adicional de la tesis de Church-Turing, la cual establece que una función es efectivamente calculable si y sólo si es computable por una máquina de Turing.

Agradecimientos Este artículo fue financiado parcialmente por la universidad EAFIT bajo el proyecto de investigación No. 817430. Los autores agradecen a los evaluadores sus comentarios y sugerencias.

Referencias

[1] Agudelo, Juan C. Máquinas de Turing paraconsistentes: algunas posibles definiciones y consecuencias. Proyecto de grado para la Especialización en Lógica y Filosofía, Universidad EAFIT 2003. http://sigma.eafit.edu.co:90/∼asicard/archivos/mtps.ps.gz

[2] Bobenrieth, M. ¿Inconsistencias, por qué no? Santafé de Bogot á: Tercer Mundo Editores, División Gráfica 1996.

[3] Jeffrey, R. and Boolos, G. Computability and logic. Cambridge University Press, 1989.

[4] Carnielli, W. and Marcos, J. Paraconsistency The logical way to the inconsistent. Walter A. Carnielli and Marcello E. Coniglio and Ítala M. L. D'Ottaviano (eds.). capítulo A taxonomy of C-systems, pp. 1–94. New York: Marcel Dekker, 2002. Preprint: ftp://logica.cle.unicamp.br/pub/e-prints/Taxonomy.pdf

[5] Da Costa, N. On the theory of inconsistent formal system. Notre Dame Journal of Formal Logic XV-4, pp. 497–510, 1974.

[6] Da Costa, N. Sistemas formais inconsistentes. Curitaba, Paraná, Brasil: Editora UFPR, 1993.

[7] Davis M. Computability and unsolvability. New York: Dover Publications, Inc. 1982.

[8] de Leeuw et al, Automata Studies (C. E. Shannon and J. McCarthy (eds.)). Computability by probabilistic machines. p. 183–212, Princenton University Press, 1956.

[9] Carnielli, W. and Epstein, R. Computability: computable functions, logic, and the foundations of mathematics. Belmont, CA: Wadsworth/Thomson Learning, 2 edition, 2000.

[10] Sicard, A. y Gómez, R. Informática teórica: elementos propedeúticos. Fondo editorial Universidad EAFIT, Medellìn 2001. Eprint: http://sigma.eafit.edu.co:90/~asicard/archivos/parteInformaticaTeorica.ps.gz

[11] Kelley, D. Teoría de autómatas y lenguajes formales. Prentice-Hall (UK) Ltd. 1995.

[12] Kleene, S. Introducción a la metamatemática. Colección: Estructura y Función. Madrid: Editorial Tecnos, 1974.

[13] Minsky, M. Computation: finite and infinite machines. Englewood Cliffs, N.J.: Prentice-Hall, Inc. 1967.

[14] Odifreddi, P. Classical recursion theory. The theory of functions and sets of natural numbers. Studies inf logic and the foundations of mathematics, volume 125. Amsterdan: North-Holland, 1989.

[15] D'Ottaviano, I. On the development of paraconsistent logic and Da Costa's work. The journal of Non Classical Logic, 7 (1/2), pp. 89–152, 1990.

[16] Turing, A. On computable numbers, with an application to the Entscheidungsproblem. Proc. London Math. Soc., pp. 230–265, 1936. A correction, ibid, vol 43. 1936-1937. pp. 544-546.

[17] Urbas, I. Paraconsistency and the C-Systems of da Costa. Notre Dame Journal of Formal, 30, pp. 583–597, 1989.

Dirección de los autores: Agudelo J., Univ. EAFIT Medellín, jagudelo@eafit.edu.co — Sicard A., Univ. EAFIT Medellín, asicard@eafit.edu.co