Simulación de modelos

Erick Hernandez Navarrete

15 de julio de 2020

Índice general

Capítulo 1

Decidibilidad en los modelos

1.1. Introducción

En este documento expondremos las nociones de decibilidad en el mundo de maquinas de Turing y en el mundo distribuido, asi como formalizaremos la noción de simulación de modelos computacionales, demostrando que el modelo A de la maquina de Turing es equivalente en poder al modelo distribuido B en particular al modelo **LOCAL**

1.2. Decidibilidad en el modelo de maquinas de Turing

1.2.1. Elementos del modelo

Primero enunciaremos los elementos del modelo de maquina de Turing,

Definition 1. *Una maquina de Turing is una 7-tupla,* $(Q, \sigma, \lambda, \delta, q_0, q_{accept}, q_{reject})$ *donde* Q, σ, λ *son conjuntos finitos.*

1. Q *es el conjunto de estados,*
2. Σ *es el alfabeto de entrada que no contiene el simbolo blanco* $\sqcup$
3. Λ *es el alfabeto de la cinta, donde* $\sqcup \in \lambda$ *y ademas* λ*,*
4. $\delta : Q \times \Omega \to Q \times \Omega \times \{L, R, S\}$ *es la función de transición,*
5. q_0 *es el estado de iniciación,*
6. q_{accept} *es el estado de aceptacion*
7. q_{reject} *es el estado de rechazo,donde* $q_{accept} \neq q_{reject}$*,*

Esta definición esta encapsulando los elementos del modelo, asi que una vez que tenemos los elementos, podremos definir la semantica de que un problema se soluble en este clasico modelo,definiendo la noción de decidibilidad.

1.2.2. Decidibilidad

Podemos observar que este modelo, las estructuras de datos que están consumiendo son cadenas a priori finitas, entonces una vez que le damos como entrada a una maquina de Turing una cadena w, podemos decir que esa cadena es aceptada o rechazada, entonces definimos lo siguiente:

Definition 2. *Sean x, y dos cadenas, y q_j con $j \in I$, un estado en el conjunto de estados Q, estos elementos del modelo Turing conformaran una semántica: xq_jy donde la posición del estado representa la localidad del cabezal en ese momento del computo. Para esa triada x, q_j, y definira la noción de configuración, que la denotaremos de la siguiente manera C_j.*

Una vez que tenemos la definición formal de **Configuración**, nos movemos a conectar esta idea con la noción de computo.

Definition 3. *Sean C_1, C_2 configuraciones:*
vamos a decir que C_1 produce a C_2 legalmente en un solo paso si semánticamente pasa lo siguiente: Sea $a, byc \in \Omega$ y tomamos también u, v cadenas y tenemos el siguiente contexto: uaq_ibv produce uq_jacv si $\delta(q_i, b) = (q_j, c, L)$ donde R, L, S son para denotar la semantica del movimiento del cabezal, derecha, izquierda o en su defecto no moverse.
En el caso de un movimiento a la derecha es: uaq_ibv produce $uacq_jv$ si $\delta(q_i, b) = (q_j, c, R)$.

La definición anterior formalizara la noción de la secuencias de configuraciones, que en esencia esta formalizando la noción de computo.

Definition 4. *Decimos que tenemos una configuración inicial para M una máquina de Turing y w con entrada una cadena w a la configuración: C_0 escrita de la siguinte manera: q_0w. El cual indica que la máquina esta en su estado inicial q_0 con su cabezal en el última localidad del lado izquierdo.*

Definition 5.

- *En una configuración de aceptación tenemos el estado q_{accept}*
- *En una configuración de rechazo tenemos el estado q_{reject}.*

Remark. *Podemos clasificar a las configuracione de la siguiente manera:*

- *Configuraciones de detención*
- *Y las que no son.*

*Dichas configuraciones tienen la propiedad de no producir otra configuración C_k, en otras palabras son configuraciones finales. Las configuraciones que son de dicha naturelaza, son la configuración de **aceptacioń** y la configuración de **rechazo**.*
Mas aún podriamos complicar la función δ tomando $Q^* = Q - \{q_{accept}, q_{reject}\}$
y redefinimos a delta como:
$\delta : Q^* \times \Omega \leftarrow Q \times \Omega \times \{L, R, S\}$

Definition 6. *Decimos que una máquina de Turing **acepta** una entrada w si existe una secuencia de configuraciones $C_1, \ldots, C_k$ tal que:*

1. *C_1 es la configuración inicial*
2. *Cada C_i produce C_{i+1} y*

3. *C_k es la configuración de* ***aceptación****.*

Definition 7. *Sea M una máquina de Turing. La colección de cadenas tales que M las acepta, es un* ***lenguaje*** *que acepta ó que es reconocible por M, y lo denotamos de la siguiente manera: $L(M)$.*

Mas aun podemos definir la siguiente nocion:

Definition 8. *Llamamos a un lenguaje* ***Turing-Reconocible*** *si existe una máquina de Turing que lo reconozca.*

Remark. *Cuando iniciamos una máquina de Turing con entrada w, pueden pasar tres escenarios:*

- *acepta,*
- *rechaza,*
- *nunca se detiene.*

En este caso, tomaremos solamente máquinas de Turing que se detienen para todas sus entradas, por lo tanto dichas máquinas siempre se dentendrán en algún momento de la ejecución de M para $\forall w$ entrada. Formalizaremos lo anterior con una definición:

Definition 9. *Sea M máquina de Turing tal que w entrada se detiene en algún momento de la ejecución, entonces decimos que dicha máquina de Turing tiene el atributo de ser* ***decidible***
Es decir, siempre entrán en su estado de ***aceptación*** *o en su estado de* ***rechazo*** *en algún momento de la ejecución.*

Y por el lado de la noción del lenguaje definimos lo siguiente:

Definition 10. *Decimos que un lenguaje es* ***decidible*** *si existe un máquina de Turing que lo* ***decide****.*

Ejemplo de lenguajes decidibles

A continuación describiremos un ejemplo de una máquina de Turing, describiendo cáda una de sus partes que la definen, así como el principio de funcionamiento de la misma.

Example 1. *Tomamos el siguiente lenguaje:*

$$A = \{0^{2n} : n >= 0\} \tag{1.1}$$

Es el lenguaje que consta de cadenas de 0's tal que su longitud es una potencia de 2.

Entonces afirmamos que M_2 es un lenguaje decidible. Entonces podemos hacer las instrucciones (alto nivel) para M_2 como sigue:

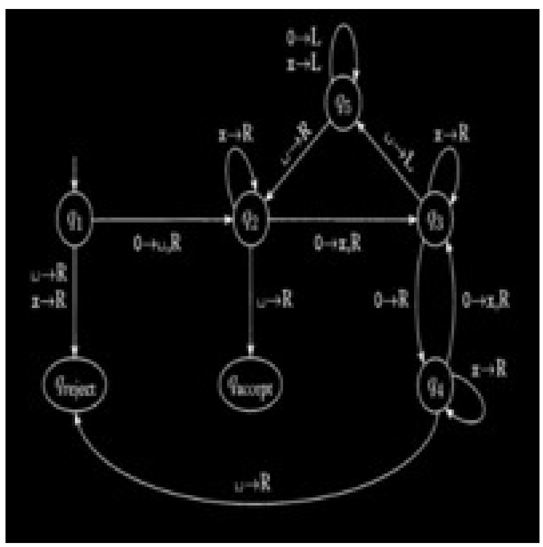
1. recorrer de izquierda a derecha a través de la cinta, tachando uno de cada dos ceros.
2. Si en la etapa 1, la cinta contenia un solo 0,entonces **acepta**.
3. Si en la etapa 1, la cinta contenía mas de un 0 y ademas contenía un numero impar, mayor a 1, **rechaza**.

4. Regresa el cabezal al lado izquierdo al final de la cinta.
5. Ve a la etapa 1.

Una vez que tenemos la descripción de la máquina de Turing a un alto nivel, podemos hacer un puente a un bajo nivel, que es formalmente describir la máquina con sus elementos que la definen para una entrada w, el cual nuestro puente será en describir la función δ vía un diagrama de estados.

En la etapa 1 de M_2 corta el número de ceros a la mitad.Mientras la máquina barre a tráves de la cinta en la etapa 1, lleva la cuenta de 0's vistos si es par o impar. Si el número es impar mayor a 1,entonces el número original de 0's en la entrada no puede ser una potencia de 2. Entonces la máquina **rechaza** en esa instancia. Como sea, si el número de 0's visto es 1, entonces el número original tiene que ser una potencia de 2, entonces en ese caso la máquina **acepta**. Sea $M_2 = (Q, \Sigma, \Lambda, \delta, q_1, q_{accept}, q_{reject})$:

1. $Q = \{q_1, q_2, \ldots, q_5, q_{accept}, q_{reject},$
2. $\Sigma = \{0\}$ and
3. $\Lambda = \{0, x, \sqcup\}$.
4. Describiremos a δ con un diagrama de estado.
5. Los estados de inicio, aceptación y rechazo son $q_1, q_{accept}, q_{reject}$ respectivamente.



En este diagrama, la etiqueta $0 \to \sqcup, R$ que aparecé en la transición del estado q_1 al estado q_2, tiene la semántica de la definición de la función δ, que en otras palabras se describe como si: en el estado q_1 con el cabezal leyendo 0, la máquina va al estado q_2, escribe $\sqcup$ y mueve el cabezal a la derecha. Lo cúal en términos formales significa que: $\delta(q_1, 0) = (q_2, \sqcup, R)$. Otra escritura que usaremos para hacer la notación en este ejemplo mas compacta, es denotar: $0 \leftarrow R$, que la semántica es que se hace la transición del estado q_3 al estado q_4, que es la etiqueta que se aprecia en el diagrama de estados, y ademas significa que la máquina se mueve a la derecha en el momento en el que lee un 0 en el estado q_3, pero que no altera la cinta(hace una operación escritura), entonces $\delta(q_3), 0) = (q_4, 0, R)$ en términos del mapeo que se esta generando. Esta máquina inicia escribiendo un simbolo blanco sobre el ulti|mo 0 de lado izquierdo. En particular esto sirve para delimitar el final de la cinta con el simbolo blanco $\sqcup$ en esta particular máquina de Turing. Ahora vamos a hacer una ejecución de Turing para la entrada $w = 0000$, en el cual escribiremos la función transición δ con la semántica de las configuraciones.

La siguiente secuencia es la ejecución de M_2 con la entrada en particular $w = 0000$, Se lee hacía abajo de las columnas de izquierda a derecha.

$$\begin{array}{ccc}
q_1 0000 & \sqcup q_5 x 0 x \sqcup & \sqcup x q_5 x x \sqcup \\
\sqcup q_2 000 & q_5 \sqcup x 0 x \sqcup & \sqcup q_5 x x x \sqcup \\
\sqcup x q_3 00 & \sqcup q_2 x 0 x \sqcup & q_5 \sqcup x x x \sqcup \\
\sqcup x 0 q_4 0 & \sqcup x q_2 0 x \sqcup & \sqcup q_2 x x x \sqcup \\
\sqcup x 0 x q_3 \sqcup & \sqcup x x q_3 x \sqcup & \sqcup x q_2 x x \sqcup \\
\sqcup x 0 q_5 x \sqcup & \sqcup x x x q_3 \sqcup & \sqcup x x q_2 x \sqcup \\
\sqcup x q_5 0 x \sqcup & \sqcup x x q_5 x \sqcup & \sqcup x x x q_2 \sqcup \\
 & & \sqcup x x x \sqcup q_{accept}
\end{array}$$

1.3. Decidibilidad en el modelo de computo distribuido LOCAL

Ahora lo que haremos es definir el modelo de computo distribuido de manera formal, y tomar un modelo en particular en la que enfocaremos nuestro estudio, para posteriormente estudiar la noción de decibilidad en dicho modelo.

1.3.1. Presentación del modelo de computo distribuido

Presentación del protocolo de comunicación

Este modelo tendrá una capa de abstracción de comunicación, asi como su respectiva capa de computación de la siguiente manera.

Capa de comunicación

El modelo de comunicación consiste en una red de comunicación 1-1 que será descrita en términos formales por una gráfica conexa, no dirigida $G = (V, E)$, donde los vertices $V = \{v_1, \ldots v_n\}$. operando entre ellos. Inicialmente, consideraremos identificadores unicos asignados a los procesos de la gráfica G. Má||s concretamente consideraremos a estos identificadores de un conjunto ordenado de enteros: de la siguiente manera:

$$S = \{s_1, \ldots, s_n\} \; donde: \;\; s_i < s_{i+1} \; \forall i \leq 1 \tag{1.2}$$

Entonces con esta notación, una ID-asignación es un mapeo: $ID : V \rightarrow S$ entonce nos referiremos a su identificador como: $ID(v)$.
Entonces la comunicación se llevara acabo de la siguiente manera:
Cada vertice tendra asociado el numero de puertos, como el $deg_G(v)$, entonces en este sentido, el conjunto de aristas adyacentes al vertice contiene exactamente $deg_G(v)$, donde cada arista esta

conectado en un puerto de v.
Podemos denotar que a cada arista (u, v) le corresponde la pareja $((u, i), (v, j))$ donde $1 \leq i \leq deg_G(u)$ y $1 \leq j \leq deg_G(v)$, con semántica: **un canal de comunicación conectadose en el puerto i de u con el puerto j de v.**
El vertice u envía un mensaje a sus vecinos v cargando el mensaje en puerto apropiado, digamos i. Este mensaje es recivido por v a tráves del puerto j.

Capa de Computación

Una vez que tenemos la capa de comunicación, presentaremos la capa formal del modelo de computo. El modelo estara governado por un algoritmo Π que estara compuesto de protocolos $\Pi_1, \ldots, \Pi_n$ donde cada Π_i residira en el vertice v_i.

Remark. *Hasta ahora hemos hablado a secas de los vertices,pero podemos darle una semantica de computo en términos de* ***procesos****, el cual significa que es una entidad de computo, y entonces a cada v_i lo nombraremos como el proceso p_i.*
Con esta convecion podemos decir que para Π_i residira en su respectivo proceso: p_i.

Remark. *Podemos observar que podemos modelar a cada Π_i como una maquina de estado para $\forall i$ con su correspodiente conjunto de estados estado Q_i conteniendo su estado inicial q_{0i} y sus estados de aceptación y rechazo: $q_{accept}, q_{reject} \in Q_i$ respectivamente tal que en cualquier momento dado el proceso p_i esta en el estado q_i de Q_i. Mas aún podemos pensar en cada Π_i como en una maquina de Turing con operaciones de envio y recepción de* ***mensajes****.*

Por otro lado en la capa de comunicación tendremos el siguiente esquema:

Definition 11. *Vámos a definir un mensaje MSG como la información local que sera enviada del proceso v al proceso u, por medio del canal $((v, i), (u, j))$, donde el proceso u tiene el atributo de la operacion send, para el cual cada proceso tiene un puerto de entrada por cada canal de comunicación adyacente, en el cual es añadida la información MSG, y el cual el proceso hace una operación de lectura para dicha entrada.*
Podemos decir que el tamaño de la información del mensaje MSG, es $(\log n)$ bits.

En cualquier momento dado y en cualquier canal de comunicación $e_i = (u, v)$ está en algun estado $\overline{q}_i$ del conjunto de estados $\overline{Q}_i$ el estado $\overline{q}_i$ esta compuesto de dos componentes denotadas de la siguiente manera: $\overline{q}_{u \leftarrow v}$ y $\overline{q}_{v \leftarrow v}$ una por cada direccion del canal de comunicación. Vamos a denotar como M cómo la colección de todos los posibles mensajes que se pueden enviar de un proceso a otro en toda ejecucion del algoritmo, cada uno de los dos componentes $\overline{q}_{u \leftarrow v}$ es un elemento de $M \cup \lambda$, $\overline{q}_{u \leftarrow v} = MSG \in M$ significa que ahora el mensaje MSG esta en transicion de u a v, y denotaremos que $\overline{q}_{u \leftarrow v} = \lambda$ para representar una semantica de que el canal actual esta vacio en esa dirección. En el inicio del computo, todos los procesos estan en el estado inicial $q_{0,i}$ $\forall i$ y todos los canales de comunicación estan vacios. Es decir que sintacticamente: $\overline{q}_{i,0} =< \lambda, \lambda >$

Ejecución de un algoritmo en este modelo

La ejecución del algoritmo en este ambiente consistira de **Eventos** ocurriendo en diversos lugares de la red y afectando a los procesos involucrados. Diremos que un **paso computacional** es una operación como máquina de Turing del proceso p. Los eventos puede ser del tipo:

- Computacional:representando un paso en un procesador
- Comunicación: representando la entrega o la recepción de un mensaje.

Donde cada evento de comunicación tiene una semántica de: $SEND(i, j, MSG)$ o $DELIVER(i, j, MSG)$ para algún MSG. Entonces a manera de reportorio de eventos tenemos:

1. Evento $COMPUTE(i)$:El proceso v_i ejecuta una operación interna, basado en su estado local, y posiblemente cambie su estado local.
2. Evento $SEND(i, j, MSG)$:El proceso v_i envia de salida un mensaje MSG en algún canal de comunicación link e_l con destino al proceso v_j
3. Evento :$DELIVER(i, j, MSG)$: El mensaje MSG originado de un proceso v_i que es enviado por el canal de comunicación e_l es entregado en la entrada del destino v_j

Entonces la computación en un sistema distribuido lo podemos pensar de la siguiente manera: Como una secuencia de configuraciones, capturando el estado actual de los procesos y los canales de comunicación.
Cada evento cambia de estado para algun procesador v_i, y posiblemente también para un canal de comunicación y eso cambiara la configuracion del sistema. En terminos formales lo podemos pensar de la siguiente manera:

Definition 12. *Una configuracion es una tupla $(q_1, \ldots, q_n, \overline{q}_1, \ldots, \overline{q}_m)$ donde $q_i, \overline{q}_j$ es el estado del procesador p_i y del canal de comunicación e_j respectivamente y la configuración inicial es:*

$$q_{0,1}, \ldots, q_{0,n}, \overline{q}_{0,1}, \cdots \overline{q}_{0,m} \tag{1.3}$$

Remark. *Como estamos pensando de manera intuitiva a cada uno de los procesos como una máquina de Turing, entonces una vez que uno de los procesos entra en alguno de los estados: $_{accept}, q_{reject}$, el proceso ya no cambía de estado, pero sus operaciones de envio y recepción de mensajes siguen activos.*

Entonces modelaremos la computación del algoritmo como una (posible) infinita secuencia de configuraciones alternadamente con eventos.

Definition 13. *La ejecución de un algoritmo Π en una grafica con cierta topologia G con una entrada inicial I en los procesos es denotado como $\kappa_{\Pi(G,I)}$. Formalmente, una **ejecución** es una secuencia de la forma:*

$$\kappa = (C_0, \rho_1, C_1, \rho_2, C_2, \ldots) \tag{1.4}$$

donde cada C_k es una configuración y cada ρ_j es un evento, y en particular C_0 denota la configuración inicial.

Podemos imponer ciertas restricciones en las ejecuciones del algoritmo, pero por el momento podemos definir formalmente el concepto en terminos de ejecuciónes para algun algoritmo Π
Entonces lo anterior nos permite definir lo siguiente:

Definition 14. *Diremos que un modelo es un subconjunto de esas posibles ejecuciones*

Con la definición anterior, nos basaremos en un modelo en particular con una propiedad en particular, a saber el que tiene una estructura de rondas.

Definition 15. *Diremos que un* ***modelo*** *tiene el atributo de rondas si:*

1. *Cada proceso p ejecuta* ***send*** *a todos sus vecinos,* ***deliver*** *de todos sus vecinos y finalmente* ***compute****,*

2. *Cada proceso ejecuta su r-ésima ronda si todos los procesos ejecutaron su $r-1$ ronda.*

La definición anterior nos permitira definir el siguiente modelo:

Definition 16. *Llamaremos $LOCAL$ al modelo que posea la estructura de rondas.*

Y podemos observar que el punto 2 de la definición de rondas, es lo que da el cáracter de síncronia en la ejecución.

Ejemplos

1.3.2. Decidibilidad

Una vez que tenemos los elementos del modelo distribuido, podemos introducir la noción de **decibilidad**, en este modelo de computo formal.

Definition 17. *Sea w una cadena, podemos escribir a la cadena como $w_0, \ldots, w_n$, la cual sera la entrada al algoritmo $\Pi(w)$, donde de manera distribuida, tendremos como inicialización, que cada proceso p tendra como entrada un caracter de la cadena w, digamos $p_j(w_k)$.*
Sea $\kappa_{\Pi(w,G)}$ una ejecución del algoritmo Π con entrada w en la grafica G, entonces diremos que la entrada w es aceptada, si existe una configuración en la ejecución $\kappa_{\Pi(w,G)}$, digamos C_k tal que existe un estado q_a en C_k, de su correspondiente proceso p_a, tal que $q_a = q_{accept}$.
Es decir, que el estado en esa configuracion es exactamente el estado q_{accept}.
En simbolos:

$$\forall \kappa_{\Pi(w,G)},\ \exists C_K \ \mid \exists p_a \ \mid q_a = q_{accept}. \tag{1.5}$$

Lo anterior, nos permitira definir lo siguiente:

Definition 18. *Al conjunto de cadenas que acepta un algoritmo distribuido Π es el lenguaje de Π, o el lenguaje que decide Π, y lo denotaremos como $L(\Pi)$.*

Capítulo 2

Simulación de modelos Maquina de Turing y LOCAL

2.1. Noción de simulación en modelos

Una vez que tenemos estos dos modelos de computo formal, a nivel logico podemos decir que:

Definition 19. *Decimos que un modelo T simula un modelo S si:*

$$\forall x \in L(S) \text{ entonces } x \in L(T) \tag{2.1}$$

Mas aún decimos que son modelos equivalentes (computacionalmente) si:

$$\forall x \in L(T) \iff x \in L(S) \tag{2.2}$$

Entonces enunciaremos nuestro teorema de la siguiente manera:

Theorem 1. *Sea TM una maquina de Turing, entonces existe un Π algoritmo distribuido que simula a TM,con la semántica de **simulación** con base a la definición anterior.*

Una vez esto nos podemos adentrar en el diseño de un algoritmo en el que burdamente le daremos en la entrada una rebanada de la cadena que es aceptada por una maquina de Turing en Abstracto y que es aceptada por dicho algoritmo.

2.2. Diseño del algoritmo

Remark. *Trivialmente podemos pensar que al darle la entrada la cadena que es aceptada por una maquina de Turing, es consumida, tal que cada proceso la tiene enteramente como entrada, i.e $\Pi(w)$ entonces de manera local se da que $v_j(w)$ entonces esta en particular que en algun momento de la ejecucion el estado de ese proceso localmente, $q_i \leftarrow q_{accept}$ entonces $w \in L(\Pi)$, pero podemos observar que la forma de la entrada en el algoritmo Π es de manera no distribuida.*

Entonces no es verdaderamente un diseño de un algoritmo distribuido,por lo tanto procederemos a diseñar de manera distribuida el siguiente algoritmo: Daremos la distribución de la información a manera de contricante de la siguente manera: Sea $w \in L(TM)$ para una maquina de Turing TM arbitraria, entonces decimos que una rebanada de la cadena $w[i]$ tiene localidad i. Esta semantica nos va a permitir definir lo siguiente:

Definition 20. *Sea un v_k un proceso del modelo en particular LOCAL diremos que dicho proceso tendra como entrada a una rebanada w_i de la cadena w denotada tambien como w_j, de localidad j*

En general podremos alimentar a cada proceso con una familia de rebanadas de cierta localidad. Esto nos da la noción del control externo de las entradas, que por el momento esto nos esta generando una cierta familiaridad del papel a un alto nivel de la visión del contrincante como podremos observar esta es la contraparte del algoritmo que esta gobernando computacionalmente (o por la capa de computo) por el algoritmo. Entonces nos propondremos el siguiente diseño del algoritmo que nos dara a priori la solución del problema que estamos atacando.

Algorithm 1 $Simula\_Algo\_TM(w)$

$w_1 \ldots w_k \leftarrow w$
Síncronamente
for all $r = 1$ **to** n **do**
  $v_j(w_i)$\{Codigo para v_j\}
  $q_j \leftarrow q_0$\{Poner estando inicial en 0\}
  while true do
    call $\delta(q_j, w_i)$
    $(q_r, w_r, P) \leftarrow \delta(q_j, w_i)$
  end while
  if $q_r = q_{accept}$ **then**
    return q_r
  else
    send$(t, MSG \leftarrow< q_r, w_r, P >)$
  end if
end for

2.3. Descripción del algoritmo

Entonces lo que podemos observar que en esencia estamos delegando con la función δ que es parte de la información local del proceso v_j pero tendremos un control en el que de manera implicita por la parte que esta teniendo la visión del contrincante,que es la naturaleza de la distribución de la información, en las entradas del buffer de cada $v_j \in V(G)$, donde la naturaleza topologica de G es abstracta a priori. Pero la logica del token es que esta llamada sera iterada hasta que la rebanada de la cadena que nos da la invocacioń de δ sea una rebanada de localidad correspondiente al actual proceso que esta teniendo el evento $COMPUTE(j)$. Asi cuando esto sea falso, tendra una logica de aceptación o en su defecto de iniciar un evento del tipo de comunicación: $SEND(j, t, MSG)$ donde el MSG es lo que nos arroja el ultimo llamado de δ, donde t, denota sin perdida de generalidad el indice de uno de sus vecinos. Asi que enunciaremos los siguientes afirmaciones a manera de teorema del cual se desprendera la simulación de TM maquina de Turing via el modelo $LOCAL$.

2.4. Demostración del procedimiento $Simula\_Algo\_TM$

Theorem 2. *El algoritmo Simula_Algo_TM es correcto.*

Demostración. Sea r una ronda de la ejecución del algoritmo $\Pi = Simula\_Algo\_TM$, al incio de esa ronda se estara iniciando un evento del tipo $COMPUTE(k)$, de nuestro repertorio de eventos para el proceso v_k, por la naturaleza de la distrubución de la información llegara un momento de la iteración en la que se de una estructura de dato $msg \leftarrow < q_r, w_r, P >$ arrojada por el llamado iterativo de δ, ya que la localidad de w_r no esta asignada a v_k sin perdida de generalidad. Entonces siguiendo el codigo, observamos que tenemos una logica para la estructura de dato: si $q_r = q_{accept}$ entonces $w \in L(\Pi)$ y se acabaria la ejecución en dicha ronda. Si no, entonces se activa el evento $Send(t, MSG)$,donde sin perdida de generalidad t representa el indice de uno de los vecinos del proceso k, por otro lado como $w \in L(TM)$ entonces $\exists v_l$ en la ronda $r + 1$ tal que al final dicha ronda $\exists q_l$ estado tal que $q_l = q_{accept}$.
$\therefore w \in L(\pi)$, por lo tanto el algoritmo es correcto.

$\square$

Una vez que tenemos la corrección del algoritmo se desprende a manera de corolario la simulación de TM en $LOCAL$.

Corrollary 1. *Sea TM una maquina de Turing, entonces:*

$$\forall w \ \in L(TM) \ \exists \Pi \ algoritmo \ en \ LOCAL \ t.q \ w \ \in L(\Pi) \tag{2.3}$$

Demostración. Sean $w \in L(TM)$ para una maquina de Turing y $\Pi = Simula\_Algo\_TM$, $\Pi(w)$ como dicho algoritmo es correcto por el teorema 2, entonces ya tenemos un algoritmo en $LOCAL$ que hace que $w \in L(w)$ $\forall w \in L(TM)$, que semanticamente se reduce a que Π **simula** a TM con TM en abstracto. $\square$

Entonces una vez que tenemos un algoritmo que es correcto a nivel semantico, la siguiente pregunta es la complejidad asociada a la ejecución de $\Pi \leftarrow Simula\_Algo\_TM$ tanto espacial,de comunicación asi como temporal.

2.4.1. Complejidad del algoritmo