Simulación de modelos

Erick Hernandez Navarrete

4 de junio 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 la cinta del alfabeto, donde* $\sqcup \in \lambda$ *y ademas* $\sigma \subseteq \lambda$*,*
4. $\delta : Q \times \lambda \to Q \times \lambda \times \{L, R, S\}$ *es la funcion 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 estan consumiendo son cadenas finitas(infinitas),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. *Decimos que una maquina de Turing acepta una entrada w si existe una secuencia de configuraciones $C_1, \ldots C_n$ tal que:*

1. *C_1 es la configuración inicial*
2. *Cada C_i le sigue C_{i+1} y*
3. *C_k es la configuración de aceptación*

Definition 3. *Sea M una maquina de Turing. La colección de cadenas tales que M las acepta, es un lenguaje decidible por M, y lo denotamos de la siguiente manera: $L(M)$.*

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 tendra una capa de abstracción de comunicacion, asi como su respectiva capa de computación de la siguiente manera.

Capa de comunicación

El modelo de comunicación consiste en un protocolo uno a uno en la red que sera descrita por una grafica conexa, no dirigida $G = (V, E)$, donde los vertices $V = \{v_1, \ldots v_n\}$ representan a los procesos de la red y los aristas representan los canales de comunicación bidireccionales operando entre ellos. Inicialmente consiraremos identificadores unicos asignados a los procesos de la grafica G, Mas concretamente consideraremos a estos identifiadores de un conjunto ordenado de enteros: de la siguiente manera:

$$S = \{s_1, \ldots, s_n\} \; donde: \;\; s_i < s_{i+1} \; \forall i \leq 1 \tag{1.1}$$

Entonces con esta notacion, una ID-asignación es un mapeo: $ID : V \to S$ 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 adjacentes 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 $1 \leq j \leq deg_G(v)$ que le da la semantica de que el canal esta conectado en el canal i del proceso u con el puerto j del proceso v.

Capa de Computacion

Una vez que tenemos la capa de comunicacion,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 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} tal que en cualquier momento dado el proceso p_i esta en el estado q_i de Q_i.*

Por otro lado en el lado de la capa de comunicacion tendremos el siguiente esquema: En cualquier momento dado y en cualquier canal de comunicación $e_i = (u, v)$ esta 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 dirección del canal de comunicación. Vamos a denotar como como la colleccion de todos los posibles mensajes que se pueden enviar 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 $\cup \lambda$, $\overline{q}_{u \leftarrow v} = MSG \in$ significa que ahora el mensaje MSG esta en transicion de u a v, y denotaremos que $\overline{q}_{u \leftarrow v} = \lambda$ para representar el echo de que el canal actual esta vacio en esa dirección. En el inicio del computo, todos los procesos estan en el estado inicial $q_{0,i}$ y todos los canales de comunicación estan vacios. Es decir que sintacticamente: $\overline{q}_{i,0} =< \lambda, \lambda >$

Ejecución de un algorimo 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 afectados. 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 semantica 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. Event $SEND(i, j, MSG)$:El proceso v_i envia de salida un mensaje $MSG \in$ en algun canal de comunicacion link e_l con destino al proceso v_j
3. Event:$DELIVER(i, j, MSG)$: El mensaje $M \in$ 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 4. *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:*

$$q_{0,1}, \ldots, q_{0,n}, \overline{q}_{0,1}, \cdots \overline{q}_{0,m} \tag{1.2}$$

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

Definition 5. *La ejecución de un algoritmo* Π *en una grafica con cirta 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:*

$$\kappa = (C_0, \rho_1, C_1, \rho_2, C_2, \ldots) \tag{1.3}$$

donde cada C_k *es una configuración y cada* ρ_j *es un evento.*

C_0 es la configuración inicial y el mapeo C_{k-1}, ρ_k, C_k tiene una semantica natural, i.e por decir que ρ_k es un evento de naturaleza computacional para un proceso v_i,llamado por el nombre del repertorio de eventos $COMPUTE(i)$ entonces el unico cambio en C_k con respecto a C_{k-1} es en el estado de v_i (es decir es un cambio local), mas concretamente, el estado de v_i en C_k es el resultado de applicar la función de transición de v_i al estado de transición al estado de C_{k-1}. De igual manera los eventos pueden ser de naturaleza en la capa de comunicación: i.e ρ_k podriamos pensar que es $SEND(i, j, MSG)$ entonces los unicos cambios de la configuracion involucra a los estados de las partes que ese evento esta envuelto, es decir cambiara el estado del proceso v_i, del canal de comunicación e_l que conecta con el proceso v_j (en particular el cambio del estaddo v_i reflajara el cambio del estado e_l, $\overline{q}_{v_i \to v_j}$ hara que cambie del estado λ a MSG. Igualmente se pensamos que ρ_k es el evento $DELIVER(i, j, MSG)$ enonces los cambios del estado de v_j se reflejan el estado del canal de comunicación e_l que conecta con el proceso v_i, que es actualizado reflejando el echo de que el el canal es vacio y nuevamente el proceso tiene una entrada en donde conecta con el canal de comunicación e_l que es a saber el mensaje MSG.

Remark. *Podemos ovservar que podemos agregar mas condiciones a la ejecución del algoritmo. En el modelo* ***asincrono****: Podemos decir que una ejecución es legal si cada proceso un numero infinito de eventos del tipo computacional, y ademas cualquiera que sean los eventos de comunicacion de algun canal de comunicación, este cambie de* $MSG \in$ *a* δ *en un tiempo finito. Lo anterior lo podemos pensar:Que estamos haciendo un mapeo 1-1 con cada* $SENT(i, j, MSG)$ *con eventos del tipo* $DELIVER(i, j, MSG)$.

Tendremos por convención que un proceso v_i despues de un evento del tipo $DELIVER(j, i, MSG)$ tendra un evento del tipo computacional. Para el modelo síncrono ademas de tener los requirimientos del modelo de computo asincrono se tendra como requerimiento que los procesos se ejecuten en pasos por bloqueo.Entonces diremos que una ejecución es legal en dicho modelo si ademas de los requerimientos del modelo asincrono, la condición es que los eventos computacionale ocurriran en **rondas**. Una de las condiciones de estas restricciones es que a cada proceso se le permite un evento del tipo computacional por cada ronda. Y ademas cada evento de computo en la ronda r aparecera despues de los eventos de computo de la ronda $r - 1$. Y ademas cada mensaje enviado en la ronda r tendra que ser enviado antes de los eventos de computo de la ronda $r + 1$. Vamos a dotar a este modelo de una diversidad de restricciones con ellos se daran tres tipos de modelos dependiendo la restriccion de el peso de los mensajes, de la ejecución de los eventos,por el momento nos vamos a centrar en las restricción de rondas en los eventos,y no vamos a poner restricción en el tamaño de los mensajes, a este modelo lo vamos a bautizar con el nombre de **LOCAL**.

1.3.2. Decidibilidad

Una vez que tenemos los elementos del modelo distribuido,podemos introducir la noción de **decibilidad**

Definition 6. *Sea una cadena $w = w_1, \ldots, w_n$ que esta alimentada en el algoritmo distribuido a saber $\Pi(w)$, donde de manera distribuida estamos dando inicialmente en la entrada de cada proceso v_j con un pedazo de la cadena w digamos w_k que representa un pedazo de la cadena que es la entrada del proceso anunciado. Entonces decimos que el algoritmo distribuido acepta a la cadena w si para toda ejecución del algoritmo existe un proceso v_t tal que con su entrada local tiene un estado de aceptación.*
Formalmente: Π ***acepta*** *a w si:*

$$\forall_{\Pi(G,w)}\ \exists v_t\ t \in \{1, \ldots, m\} \text{ tal que } q_t = q_{accept}. \tag{1.4}$$

Donde acepta quiere decir que en algun momento de la ejecución del algoritmo, el correspondiete estado $q_t = q_{accept}$ que es la semantica de aceptación formalmente, denotaremos a las cadenas que son aceptadas por el modelo $LOCAL$ para un algoritmo arbitrario Π 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 7. *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 8. *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 ... w_k ← w
Síncronamente
for all r = 1 to n do
   v_j(w_i){Codigo para v_j}
   q_j ← q_0{Poner estando inicial en 0}
   while true do
      call δ(q_j, w_i)
      (q_r, w_r, P) ← δ(q_j, w_i)
   end while
   if q_r = q_accept then
      return q_r
   else
      send(t, MSG ←< 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. □

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. □

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