Segunda Parte del avance de Tesis.

Resumen

Expondremos las nociones y atributos de los conceptos que nos permita hacer la simulación entre el mundo distribuido y el mundo de las maquinas de Turing.

Definiciones

En esta parte del documento vamos a formalizar las nociones, de las cuales se desprenderá casi de manera inmediata la subrutina que dara el control en la estructura recursiva.
Entonces definamos las nociones de la siguiente manera:
Una vez que tenemos la maquina de Turing como modelo formal en Computación, entonces con los elementos que tiene la maquina de Turing y el modelo particular $LOCAL$ de computo Distribuido haremos lo siguiente, definamos la noción de localidad de la siguiente manera:

Definition 0.1. *Sea $v \in VG$ un proceso en el modelo Formal **LOCAL**, vamos a denotar lo siguiente: (w_i, p) que va a denotar que el elemento de la cadena w_i tiene localidad i, o en otros términos, que a v le pertenece el elemento de localidad i.*

De aquí se desprende la relación que existe entre los elementos de la cadena y los procesos del modelo **LOCAL**

Definition 0.2. *Sean $v \in VG$ y $w_i \in w$ un proceso y un elemento de la cadena que es aceptada por TM entonces decimos que están relacionados si y solo si (w_i, v), i.e si y solo si w_i tiene localidad v*

Remark. *Una vez que tenemos esas nociones podemos construir un conjunto para cada proceso v en $LOCAL$:*
$LOC(P) = \{w_i \in w : (w_i, P)\}$ que es generada por la relación definida anteriormente, este conjunto esta encapsulando las nociones anteriormente descritas.

Entonces una vez que tenemos esto bien definido, podemos hacer lo siguiente:
De manera local sea (w_i, v) entonces en un sentido computacional y de entrada como un sistema computacional se traduce a $v(w_i)$.

Descripción del Diseño del Algoritmo

Una vez que tenemos la pareja entonces tenemos de manera computacional lo siguiente $v(w_i)$ que es el procedimiento computacional que hemos de desarrollar tal que solucione nuestro problema con la entrada de localidad asignada, entonces lo que haremos es definir primero una instancia de δ de tal manera que cuando hagamos la llamada a delta esta llamada sera iterativa hasta que la localidad de la cadena que nos de la subrutina delta nos de una localidad distinta al que tiene el proceso actual entonces tendremos un mensaje M o en su defecto acabara su computo en esa ronda, pero si no lo que hará es enviar es el estado que produjo y la cadena que nos genero en esa ronda r el proceso p.
Pero para ello construiremos una subrutina Booleana que lo que hará es un token en las entradas via nuestra noción de localidad. Una vez que tenemos las estructuras bien definidas hacemos lo siguiente:

Algorithm 1 $Func\_Boolean(w_i, p)$

```
if (w_i, p) then
    return true
else
    return false
end if
```

Básicamente lo que esta haciendo es dar una estructura booleana que sera la afirmación que dara el control en el siguiente algoritmo. Una vez que tenemos la subrutina de tipo Booleana, diseñemos el algoritmo principal del cual demostrara que la cadena w es una cadena aceptada por el modelo **LOCAL**, y esto a un alto nivel demostrara nuestro teorema principal.

Algorithm 2 $Simulate\_Algo\_TM(w, C, TM)$

```
w ← (w_1, ..., w_n)
Síncronamente
for all r to to m do
    p(w_i, δ())
    q ← q_0
    while func_boolean(w_i, p) do
        call δ(q, w_i) ← (q_R, w_R, p)
    end while
    if q_R = q_accept then
        return q_accept
    else
        M =< q_R, w_R, Pos >
        send M
    end if
end for
```

Ejemplo

Podemos exponer de manera concreta la siguiente situación:
tomar una linea de procesos, de tal manera que el pariente de i es $i+1$, para i en el conjunto de indices de los procesos de $V(G)$, y mas aun el pariente de n es 1.
Entonces lo que podemos observa es que tenemos computacionalmente es que la subrutina $func\_boolean()$ es el que da el control del computo que tienen los procesos del modelo $LOCAL$, asi que este ejemplo nos da una noción natural de lo que esta ocurriendo por que en particular se tiene:
(w_i, p_j) si y solo si $i = j$, esta asignación es por construcción de la topología de la gráfica, esto nos da una noción geométrica de lo que computacionalmente esta ocurriendo en este algoritmo distribuido.
Lo que prosigue es la demostración de la corrección del algoritmo.

Theorem 0.1. *El algoritmo $Simulate\_Algo\_TM$ es correcto.*

Demostración. Sea $i = r$ una roda en la ejecución del algoritmo.
En esa ronda tenemos la siguiente estructura de dato $M =< w_R, q_R, P >$ donde $P \in LEFT, RIGHT, STAY$ para esa ronda tenemos que el w_R que es el resultado de la llamada recursiva de δ, podemos concluir que $func - boolean(w_r, p) = false$ por lo tanto no es de la localidad de p, entonces podemos tomar las siguientes decisiones: como $w_R \in w$ es decidible en el sentido de la maquina de Turing, entonces q_R puede que sea el estado de aceptación, en ese caso el algoritmo toma la decision regresar q_R en su defecto toma la decision de enviar el mensaje M a sus vecinos, para el cual como TM es tal que w es decidible entonces sin perdida de generalidad en la ronda $r = i + 1$ se tendrá el estado de aceptación para un p proceso p con el ambiente de $LOCAL$.
Por lo tanto existe un proceso p tal que regresa un estado de aceptación.
$\therefore w \in L(Simulate\_Algo\_TM)$.
$\therefore Simulate\_algo\_TM()$ es correcto. □

Corollary. *$\forall w \in L(TM)$ $\exists \pi$ algoritmo en el modelo $LOCAL$ tal que $w \in L(\pi)$.*

Demostración. Por el teorema anterior hemos construido un algoritmo en $LOCAL$ tal que hace decidible a w, i.e $w \in L(Simulate\_Algo\_TM)$(En el sentido distribuido).
Mas aún como w es tal que $w \in L(TM)$ entonces podemos concluir que $\pi = Simulate\_Algo\_TM$.
$\therefore \forall w \in L(TM), w \in L(\pi)$. □