UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

EL PROBLEMA DE SINCRONIZACION DE RELOJES EN SISTEMAS DE COMPUTO DISTRIBUIDOS DESDE EL PUNTO DE VISTA FORMAL DEL CONOCIMIENTO

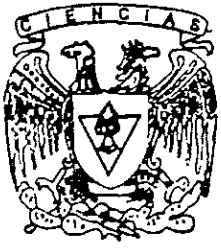
T E S I S

QUE PARA OBTENER EL TITULO DE:

LIC. EN CIENCIAS DE LA COMPUTACION

P R E S E N T A:

MANUEL ALBERTO SUGAWARA MURO



FACULTAD DE CIENCIAS
UNAM

DIRECTOR DE TESIS:

DR. SERGIO RAJSBAUM GORODEZKY

2000

MAT. MARGARITA ELVIRA CHÁVEZ CANO
Jefa de la División de Estudios Profesionales
P r e s e n t e

Comunicamos a usted que hemos revisado el trabajo de Tesis.

"El Problema de Sincronización de Relojes en Sistemas de Cómputo Distribuidos desde el Punto de Vista Formal del Conocimiento "

realizado por Manuel Alberto Sugawara Muro

Con número de cuenta 9561784-4 , pasante de la carrera de Ciencias de la Computación

Dicho trabajo cuenta con nuestro voto aprobatorio.

A t e n t a m e n t e

Director de tesis Propietario	Dr. Sergio Rajsbaum Gorodezky
Propietario	M. en C. Elisa Viso Gurovich
Propietario	Fís. Miguel Angel Rodríguez Sosa
Suplente	Dra. Atocha Aliseda Llera
Suplente	M. en C. José Alfredo Amor Montaño

Consejo Departamental de MATEMATICAS

M. EN C. MA. GUADALUPE ELENA IBARGUENGOITIA GONZALEZ

Alguien contó los días,
Alguien ya sabe la hora
JORGE LUIS BORGES

Agradecimientos

Me gustaría agradecer a todas las personas que han contribuido en mi formación profesional y en el presente trabajo. Desgraciadamente la finitud del tiempo y del espacio me impiden hacer un listado detallado; sin embargo, no quisiera dejar de mencionar a algunas personas. Espero que cualquier omisión de mi parte pueda ser comprendida y disculpada.

A mi asesor, Sergio Rajsbaum, de quien aprendí el estudio formal de los sistemas distribuídos y el *apasionante mundo del tiempo* así como el complejo reto que ha implicado para el hombre su medición e incluso su comprensión. El trabajo previo de Sergio en el tema es, sin lugar a dudas, el corazón de la presente tesis y un firme pilar del mundo académico.

A la maestra Elisa Viso, quien de forma tan determinante ha contribuido en mi formación profesional y personal. Elisa ha sido para mi, una maestra, una fuente de inspiración, pero sobre todo una amiga. Gracias por todo.

A mis sinodales, Atocha, José Alfredo y Miguel Ángel. En particular, Atocha y José Alfredo de quienes tuve el enorme placer de ser *alumno. Con ellos* aprendí la magia de la lógica y de las matemáticas

Por último agradezco infinitamente a mi familia, en particular a mis papas Lucero y Masae, a mi tía Herlinda, a mi abuelita Susana, a mis hermanas Addy y Mariana y a la pequeña Camila. A ellos debo lo que soy y lo que he sido. A ellos quisiera dedicar, con mucho cariño, el presente trabajo.

Índice General

Capítulo 1

Introducción

Los sistemas de cómputo distribuidos son ampliamente usados hoy en día, dada la basta cantidad de aplicaciones inherentemente distribuidas o por cuestiones de confiabilidad, rendimiento y economía. Una parte fundamental del trabajo de construir sistemas distribuidos, es el diseño, implementación y análisis de protocolos distribuidos.

Razonar acerca del conocimiento de los procesadores en un sistema de cómputo distribuido, es una valiosa herramienta para el análisis y la comprensión de protocolos distribuidos. El interés principal de este trabajo es desarrollar el *análisis* desde el punto de vista del conocimiento de los procesadores acerca de la sincronía relativa de los eventos de un sistema y su aplicación directa al problema de sincronización de relojes en los sistemas de cómputo distribuidos. En general, el objetivo de la sincronización de relojes, físicos o lógicos, dependiendo de la aplicación, es lograr que procesadores físicamente dispersos adquieran una noción común del tiempo, donde "tiempo" puede significar desde una precisa aproximación del tiempo real hasta un sencillo contador entero. La granularidad del tiempo se define en términos de la aplicación particular.

En un artículo clásico [Lam78] acerca de problemas de sincronización en sistemas distribuidos, Lamport introduce una sencilla noción de reloj lógico, la cual se basa en las relaciones causales de los eventos del sistema distribuido. Intuitivamente, la definición de Lamport nos dice que un evento e afecta causalmente a un evento e' si e debe suceder *necesariamente antes* que el evento e'. Un sencillo ejemplo de relación causal entre eventos es la comunicación. En todos los casos, el receptor recibe el mensaje *después* de la emisión del mismo. En este caso, decimos que la emisión del mensaje *afecta causalmente* su correspondiente recepción.

En [Lam78], Lamport define formalmente la noción de reloj lógico para sistemas distri-

buidos y deriva un sencillo algoritmo distribuido de sincronización. Intuitivamente, un reloj lógico se puede ver como una función C, de los eventos observados por el procesador, en los números naturales, que satisface la siguiente condición: si e afecta causalmente a e', entonces $C(e) < C(e')$. Por medio de los relojes lógicos de Lamport, es posible resolver una gran cantidad de los problemas de sincronización que aparecen comúnmente en los sistemas distribuidos. Sin embargo, como se ejemplifica en [Lam78], existen problemas de sincronización que dependen de eventos externos al sistema, y dichos eventos pueden depender a su vez del tiempo real. Dos claros ejemplos son los sistemas bancarios y de reservación. Para estos casos, Lamport propone en [Lam78] un algoritmo genérico de sincronización de relojes físicos. Existe una gran cantidad de aplicaciones prácticas para relojes físicos sincronizados con muy alta precisión; véase [Lis93].

Como se ha notado con anterioridad [PSR94], la dificultad básica en la sincronización de relojes radica en el deterioro que sufre la información acerca del tiempo sobre los ejes espacial y temporal. Es decir, cuando la frecuencia de los relojes locales no se conoce con exactitud, la sincronización pierde "calidad" conforme el tiempo pasa. Aunado a esto, existe cierta incertidumbre inherente acerca del tiempo de envío y recepción de los distintos mensajes del sistema.

Desde el punto de vista práctico los relojes y las líneas de comunicación perfectas no existen, sin embargo, en muchas ocasiones existen algunas garantías *a priori* del sistema acerca de su comportamiento con respecto al tiempo real, derivadas, por ejemplo, de garantías que hacen los fabricantes del equipo. Normalmente, se conocen cotas en el posible desplazamiento de los relojes locales con respecto al tiempo real. Por ejemplo, el error relativo de relojes controlados por cristales de cuarzo con una frecuencia de $60Hz$ es generalmente inferior a 10^{-5} pulsos por hora. Dichas cotas se conocen como *cotas de desplazamiento*. Usualmente, es conocen también cotas en el tiempo necesario para transmitir un mensaje por una línea de comunicación. Dichas cotas se conocen como *cotas de retardo de los mensajes*. Intuitivamente y como se hace notar en [PSR94], la esencia de todos los problemas de sincronización de relojes radica en hacer uso óptimo de dichas garantías.

Una sencilla variante del problema es cuando un procesador tiene que obtener cotas en la lectura de su reloj local, cuando un evento remoto ocurra en la ejecución. En el ejemplo 1.1 se introducen la ideas principales de esta sencilla variante.

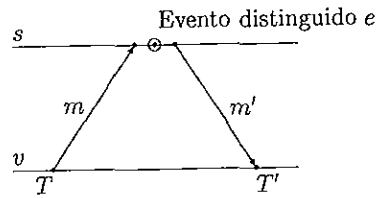


Figura 1.1: El evento distinguido e sucede en s entre los tiempos locales T y T' del procesador v

Ejemplo 1.1.
Considérese un sistema que consiste de dos procesadores s y v, conectados por medio de una línea bidireccional de comunicación. Véase el diagrama de tiempo espacio de la figura 1.1 (en el apéndice A se definen los diagramas de tiempo espacio). Supóngase que el procesador v envía un mensaje m a s cuando el reloj local de v muestra T; el procesador s responde enviando un mensaje m' a v el cual es recibido en v en su tiempo local T'. Supóngase además que el evento distinguido ocurre en el procesador s *después de que* m *es recibido y antes de* que m' sea enviado.

Si deontamos por $e \rightarrow e'$, el hecho de que el evento e afecta causalmente al evento e', podemos deducir que $send(m) \rightarrow receive(m)$; es decir, el evento del envío el mensaje m desde v *afecta causalmente* su correspondiente recepción en s. Similarmente podemos ver que se cumplen las siguientes relaciones:

1. $\text{receive}(m) \rightarrow e$
2. $e \rightarrow \text{send}(m')$
3. $send(m') \rightarrow receive(m')$

Como el evento de envío del mensaje m afecta causalmente al evento distinguido e, el evento e *no pudo ocurrir antes* que el reloj del procesador v marque el tiempo T. Similarmente, como el evento e afecta causalmente al evento de recepción del mensaje m', el evento e *no puede ocurrir después* de que el reloj de v marque T'. En ausencia de más información, el procesador v únicamente puede deducir que el evento e ocurrió en algun instante del intervalo $[T, T']$; en este sentido la calidad de la sincronización se puede medir en términos de

la "distancia" entre los puntos T' y T sobre la recta real. El real $|T' - T|$ se conoce como la *estrechez* de la sincronización. ∎

En general, se distinguen dos variantes del problema. La *sincronización externa* y la *sincronización interna*.

En el problema de *sincronización externa*, existe un procesador distinguido llamado la *fuente*. El trabajo del resto de los procesadores, consiste en obtener, en cada instante del tiempo, el intervalo de tamaño mínimo $[a, b]$ de tal forma que la lectura del reloj de la fuente se encuentre $[a, b]$.

En el problema de *sincronización interna*, todos los procesadores del sistema tratan de mantener sus relojes internos lo más cerca posible el uno del otro.

1.1 Antecedentes

Existe mucho trabajo en la literatura en torno a la sincronización de relojes en sistemas distribuidos. Algunos de los resultados más estudiados y conocidos son revisados en [SWL88]. Tal cantidad de trabajo se justifica por la amplia cantidad de aplicaciones.

Una de las variantes más estudiada del problema desde el punto de vista teórico, es el problema de sincronización interna, en el caso específico donde todos los relojes del sistema trabajan al ritmo del tiempo real. Dichos relojes se conocen como *libres de tendencia*. En [WL84], se estudia el caso en donde existe una línea de comunicación entre cualesquiera dos procesadores del sistema y las cotas en el retardo de transmisión de mensajes son idénticas para todas las líneas. Para este caso, se presenta un algoritmo de sincronización que obtiene la estrechez óptima en el escenario del peor caso permitido por las especificaciones del sistema.

El trabajo presentado en [HMM85], se generalizan los resultados expuestos en [WL84] a redes cuya topología es arbitraria y donde las cotas en el retardo de transmisión de los mensajes pueden ser distintas en cada línea. La idea principal en el análisis de [HMM85] es resolver el problema de sincronización por medio del uso de herramientas de programación lineal. Sin embargo, y como se observa en [AHR96], el algoritmo presentado en [HMM85] siempre da la estrechez en el escenario del peor caso, aun cuando la ejecución real suceda en condiciones más favorables para la sincronización. Esta observación motiva el trabajo presentado en [AHR96], donde se generalizan los resultados de [HMM85], derivando algoritmos

que son óptimos en cada una de las posibles ejecuciones del sistema, en vez de sólo en el escenario del peor caso.

Los trabajos anteriormente citados se centran es en obtener cotas usando un enfoque fuera de línea y centralizado. Típicamente, los algoritmos se pueden ver en dos etapas. En la primera etapa, todos los procesadores intercambian información acerca del tiempo. En la segunda, todos los procesadores envían su información a un solo procesador, el cual realiza los cómputos necesarios y distribuye los resultados de regreso a cada procesador. Sólo entonces cada uno de los procesadores puede ajustar su reloj interno.

Normalmente, el trabajo práctico se enfoca más en algoritmos distribuidos en línea. Usualmente, sistemas débilmente acoplados utilizan sincronización externa y sistemas fuertemente acoplados utilizan sincronización interna. Un importante ejemplo es NTP [Mil92], el protocolo de sincronización externa de INTERNET.

El trabajo presentado en [PSR94] presenta un marco teórico para el análisis de protocolos de sincronización en línea para las dos variantes clásicas del problema. Su trabajo se basa en un novedoso concepto llamado *gráficas de sincronización*, por medio del cual, se reduce el problema de sincronización de relojes, al cómputo de caminos de peso mínimo en gráficas, simplificando considerablemente el análisis del problema. Usando esta herramienta teórica, en [PSR94] se deriva la cota inferior en la posible estrechez de la sincronización y un algoritmo óptimo para el problema de sincronización externa. Para el problema de sincronización interna se deriva una nueva cota inferior en la posible estrechez de la sincronización.

El trabajo presentado en [MB94] se encuentra estrechamente relacionado con el trabajo presentado en [PSR94]. Sin embargo, el análisis de [MB94] se enfoca desde el punto de vista teórico del conocimiento en sistemas distribuidos. El análisis en [MB94] se basa en una generalización del orden parcial de Lamport, el cual llaman *precedencia sincronizada*. Intuitivamente, se usa la notación $e \xrightarrow{\alpha} e'$, para denotar el hecho que el evento e debe preceder al evento e' por *al menos* α unidades de tiempo real. La misma notación tiene sentido para valores negativos de α. Intuitivamente, interpretamos el enunciado $e \xrightarrow{-|\beta|} e'$ como el evento e' precede al evento e por *a lo más* β unidades de tiempo real.

Intuitivamente, la cercana relación de los trabajos presentados en [PSR94] y [MB94] se debe a que cada enunciado de precedencia sincronizada se puede ver como una arista en las gráficas de sincronización.

1.2 Organización del Trabajo

En el capítulo 2 se presenta el modelo formal de sistema distribuido que se considera en este trabajo. Intuitivamente, el modelo presentado es una formalización de lo que se denomina sistemas parcialmente síncronos de paso de mensajes [Lyn97].

En el capítulo 3 presentan las principales herramientas teóricas usadas en el resto del trabajo. Empezamos por definir el concepto de garantía de sistema, abstracción que nos permite modelar las especificaciones del sistema de manera uniforme y sencilla. A continuación se desarrolla y estudian las gráficas de sincronización, por último se define el concepto de enunciados de precedencia sincronizada en términos de las gráficas de sincronización.

En el capítulo 4 es una breve introducción al modelo formal de conocimiento en sistemas distribuidos y se introducen algunas definiciones necesarias en el desarrollo de los capítulos posteriores.

En el capítulo 5, se estudia el conocimiento en los sistemas parcialmente síncronos. En este capítulo hacemos uso de las teoría desarrollada en el capítulo 3 para caracterizar el conocimiento que tienen los procesadores acerca del comportamiento temporal del sistema.

En el capítulo 6 se presentan algunos de los resultados presentados en [PSR94] desde el punto de vista formal del conocimiento, para el problema de sincronización externa. Dicho capítulo concluye con un pequeño ejemplo y con un breve análisis de NTP, el protocolo de sincronización externa usado en INTERNET.

El presente trabajo conluye con un breve capítulo de conclusiones.

Capítulo 2

Modelo Formal de Sistema Distribuido

2.1 Sistemas de Paso de Mensajes

Modelamos la parte física de un sistema distribuido de paso de mensajes por medio de una gráfica dirigida cuyos nodos representan a los procesadores y cuyos arcos representan líneas unidireccionales de comunicación. Dicha gráfica se conoce como la *gráfica subyacente del sistema*. En este modelo, consideramos a los procesadores cómo máquinas de estado equipadas con relojes que se comunican unas con otras por medio del envío y recepción de mensajes a través de las líneas unidireccionales de comunicación en la gráfica subyacente.

Asumimos que existe un conjunto fijo de mensajes por medio del cual se comunican los procesadores; denotamos dicho conjunto por *MSG*. En este modelo, los mensajes son tripletas de la forma (i, m, j), donde i representa el procesador que envía el mensaje m, mientras j representa el procesador destinatario. Asumimos también que para toda tripleta de la forma (i, m, j) en el conjunto de mensajes del sistema *MSG* existe un arco (una línea unidireccional de comunicación) (i, j) en la gráfica subyacente del sistema.

Una suposición crucial de este modelo es que en cualquier instante del tiempo el sistema se encuentra en un *estado global* bien definido. Éste, se una ver como una fotografía instantánea del sistema en un tiempo dado. Dicho estado global incluye el *estado local* de cada uno de los procesadores y el *estado local del ambiente*. En el estado local del ambiente codificamos todos los aspectos relevantes del sistema e inaccesibles a los procesadores, como el estado de las líneas de comunicación Nótese que lo que se considera "relevante" se define en términos de la aplicación específica.

Para modelar el estado local de un procesador, asumimos la existencia de un conjunto fijo

INT_i de acciones internas para cada procesador i. La unión de dichos conjuntos se denota por INT. Para cada mensaje (i, m, j), distinguimos la acción interna $\text{send}(m, j) \in INT_i$ y una acción especial $\text{receive}(i, m) \notin INT$, las cuales representan, respectivamente, el envío y recepción del mensaje m. Distinguimos también una acción especial de paso de tiempo $\nu \notin INT$. Asumimos además que para cada procesador i existe una acción inicial $\mathsf{start}_i \in INT_i$. Bajo este marco, un evento e de un procesador i es una tripleta de la forma (i, k, a), donde $\mathsf{a} \in INT_i \cup \{\text{receive}(j, m) | (j, m, i) \in MSG\}$ y que representa la k-ésima ocurrencia de la acción a en el procesador i.

En este trabajo asumimos que los procesadores conservan en su estado local la secuencia de todos los eventos observados a través de una ejecución, así como el tiempo local en que sucedieron. Modelamos esta idea codificando dicha secuencia por medio de una función de historia. Formalmente:

Definición 2.1.1 (Historia Local). *La* historia local *de un procesador i, es una función $h : \mathbb{R}^+ \to INT_i \cup \{receive(j, m) | (j, m, i) \in MSG\} \cup \{\nu\}$ tal que h satisface que existe un conjunto finito no vacío $E = \{T_0, \ldots, T_k\} \subset \mathbb{R}^+$ con $T_0 = 0$, y $T_0 < T_1 < \cdots < T_k$ tal que*

(i) $h(T_0) = \mathsf{start}_i$

(ii) $h(T) \in INT_i \cup \{receive(j, m) | (j, m, i) \in MSG\}$ *para todo* $T \in E$.

(iii) Para toda $T, T' \in E$, *si* $h(T) = h(T')$ *entonces* $T = T'$.

(iv) Para toda $x \in \mathbb{R}^+ \setminus E$, $h(x) = \nu$.

La función de historia para el tiempo local $T \geq 0$ *es la restricción de* h *al intervalo* $[0, T]$.

Denotamos a los eventos en la historia de un procesador i por e_i^T, donde $h(T) = e$. Diremos que una función de historia h' en un tiempo local T' *extiende* a una función de historia h en un tiempo local T si $T < T'$ y para toda $x \in [0, T]$ se cumple que $h'(x) = h(x)$. La función vacía se extiende a sí misma y es la única función que extiende a una función de historia para el tiempo local 0.

Para cada procesador i, definimos el conjunto de *estados locales* L_i como un conjunto con tripletas de la forma (i, T, h) donde i es el identificador del procesador, $T \in \mathbb{R}^+$ representa la lectura de su reloj local y h es una función de historia del procesador i para el tiempo local T. Para cada procesador i asumimos que el evento local vacío, definido por $(i, 0, \emptyset)$, esta en L_i. En este trabajo, por simplicidad hacemos de lado el estado local del ambiente.

A continuación definimos el concepto de ejecución del sistema; intuitivamente, una ejecución del sistema es una descripción del mismo a través del tiempo, un conjunto ordenado de "fotografías" que van definiendo al sistema en cada instante. Dicha idea intuitiva se modela por medio una función que define el estado global del sistema en cada instante del *tiempo*. Formalmente definimos el conjunto de estados globales del sistema, denotado por $\mathcal{G}$, dados conjuntos de estados locales $L_1, \ldots, L_n$ para n procesadores, como un subconjunto de $L_1 \times \cdots \times L_n$. A continuación se define formalmente el concepto de ejecución en este modelo.

Definición 2.1.2 (Ejecución). *Dado un conjunto* $\mathcal{G} \subseteq L_1 \times \cdots \times L_n$ *de* estados globales, *una* ejecución r *es una función* $r : \mathbb{R}^+ \to \mathcal{G}$. *Si* $r(t) = (s_1, \ldots, s_n)$, *denotamos al estado local* s_i *por* $r_i(t)$ *para* $i = 1, \ldots, n$.

La definición de ejecución anterior satisface que en cada instante del tiempo t, el sistema se encuentra en un estado global bien definido, precisamente $r(t)$; nótese sin embargo que la definición anterior es demasiado general. Sin restricciónes adicionales, el conjunto de *fotografías* se podría contradecir con la realidad física del mundo. Por ejemplo, podemos ver primero la recepción de un mensaje y después su correspondiente envío. Más adelante se imponen condiciones adicionales en la definición de sistema.

Se asume también que en cada ejecución el instante en que *despierta* cada procesador se encuentra bien definido. Formalmente, para cada ejecución r y para cada procesador i, existe un real $0 < t_{\text{start}}(i, r) \leq \infty$ tal que para toda $x \in [0, t_{\text{start}}(i, r))$, $r_i(x) = (i, 0, \emptyset)$ y $r_i(t_{\text{start}}(i, r)) = (i, 0, h)$, donde h es una función de historia para el procesador i en el tiempo local 0. Diremos que un estado local (i, T, h) *extiende* a un estado local (i, T', h') si la historia local h extiende a la historia local h'.

Dentro de este marco, un *sistema* es un conjunto no vacío $\mathcal{R}$ de ejecuciones, sobre un conjunto dado de estados globales $\mathcal{G}$. En vez de tratar de definir un sistema directamente, ésta definición modela los posibles comportamientos del sistema. El requerimiento de que un sistema sea un conjunto no vacío modela la idea de que el sistema tiene al menos un posible comportamiento.

La definición general de sistema presentada requiere de algunas condiciones adicionales de correctud para modelar la idea intuitiva de sistema de paso de mensajes. Dichas condiciones pueden variar dependiendo de la aplicación específica, aunque podemos imponer en primer lugar condiciones mínimas y generales que capturan de manera global a los sistemas de paso de mensajes. Formalmente.

Definición 2.1.3 (Sistema de Paso de Mensajes). *Dado un sistema $\mathcal{R}$ de n procesadores diremos que $\mathcal{R}$ es un* sistema de paso de mensajes donde los procesadores están equipados con relojes locales *si y sólo si toda ejecución $r, r' \in \mathcal{R}$ satisface las siguientes condiciones:*

MP1. Para todo procesador i y para toda pareja de reales $0 \leq t < t'$ se cumple que $r_i(t')$ extiende a $r_i(t)$.

MP2. Si $r_i(t) = (i, T, h)$ con $h(T) = receive(j, m)$ entonces existe t' con $0 \leq t' \leq t$ tal que $r_j(t') = (j, T', h')$ y $h'(T') = send(i, m)$.

MP3. Si $r_i(t) = (i, T, h)$ y $r_i'(t') = (i, T', h')$, $h(T) = h'(T')$ entonces $h = h'$.

La condición MP1 garantiza dos importantes condiciones de los procesadores del sistema. Primero, los procesadores *no olvidan*, es decir, cada procesador registra en su función de historia local cada uno de los eventos observados por dicho procesador. Es fácil ver que estamos asumiendo implícitamente que cada procesador cuenta con una cantidad no acotada de memoria. La condición MP1 garantiza además que todo avance en el tiempo real requiere de un avance en cada reloj local; es decir, los relojes locales se pueden ver como funciones no decrecientes del tiempo real.

La condición MP2 impone una propiedad un poco restrictiva del sistema de entrega de mensajes. Los mensajes no se pueden corromper, sin embargo, permitimos que el sistema de entrega pierda los mensajes, los duplique o que retarde su entrega arbitrariamente. La condición MP3 garantiza que los eventos se definen en términos de la historia de los procesadores.

Una pareja (r, t), donde r es una ejecución y $t \in \mathbb{R}^+$ un real se conoce como un *punto*. Hacemos referencia a puntos de un sistema $\mathcal{R}$ usando la notación $(r, t) \in \mathcal{R}$. Adicionalmente, diremos que un evento e_i^T está en un estado local $r_i(t)$ y lo denotamos por $e_i^T \in r_i(t)$ si $r_i(t) = (i, T', h)$ y $e = h(T)$ para alguna $T \leq T'$. Diremos que un evento esta en un punto y lo denotamos por $e_i^T \in (r, t)$ si $e_i^T \in r_i(t)$ para algún procesador i. Definimos también el conjunto de eventos de un punto, denotado por $\mathcal{E}(r, t)$ como $\{e \mid e \in (r, t)\}$.

A continuación definimos la noción de tiempo local para cada procesador.

Definición 2.1.4 (Tiempo local). *Dados un sistema $\mathcal{R}$ de paso de mensajes donde los procesadores están equipados con relojes locales, un procesador i y un punto $(r, t) \in \mathcal{R}$, se define el tiempo local del procesador i en el punto (r, t), denotado por $\tau(i, (r, t))$ como T siempre que $r_i(t) = (i, T, h)$. Si $t < t_{\text{start}}(i, r)$ entonces el tiempo local del procesador i se encuentra indefinido.*

Nótese que las parejas $(t, \tau(i,(r,t)))$, forman una función no decreciente del tiempo real, para cada ejecución r y para cada procesador i.

Definimos también para cada ejecución, una función que nos permite recuperar el tiempo real de ocurrencia de los eventos. Dicha noción se encuentra únicamente disponible para el análisis del sistema. Formalmente:

Definición 2.1.5 (Función de tiempo real). *Dado un sistema $\mathcal{R}$ de paso de mensajes donde los procesadores están equipados con relojes locales, se define la función de tiempo real $real\_t_r(e_i^T)$ para una ejecución r y un evento e como t si $r_i(t) = (i, T, h)$ y $h(T) = e_i^T$. Si $e \notin r$, la función $real\_t_r(e)$ se encuentra indefinida.*

Una importante propiedad dentro en este modelo es la capacidad de los procesadores de recunstruir la secuencia de eventos que han observado a través de la ejecución; dicha propiedad se conoce como *memoria perfecta.* Definimos primero de manera formal dicha propiedad:

Definición 2.1.6. *Dado un sistema $\mathcal{R}$ y dos puntos $(r,t),(r',t') \in \mathcal{R}$, diremos que en $\mathcal{R}$ los procesadores tienen* memoria perfecta, *si para todo procesador i, si $r_i(t) = r'_i(t')$ entonces existe una función continua y biyectiva $f : [0,t] \to [0,t']$, tal que $r_i(x) = r'_i(f(x))$ para toda $x \in [0,t]$.*

Una consecuencia inmediata de las definiciones anteriores es que los sistemas de paso de mensajes antes definidos tienen memoria perfecta, como el siguiente lema propone:

Lema 2.1.1. *Sea $\mathcal{R}$ un sistema de paso de mensajes donde los procesadores están equipados con relojes locales. Para toda pareja de puntos $(r,t),(r',t') \in \mathcal{R}$, con t y t' distintos de 0, si $r_i(t) = r'_i(t')$ entonces existe una función continua y biyectiva $f : [0,t] \to [0,t']$, tal que $r_i(x) = r'_i(f(x))$ para toda $x \in [0,t]$.*

Demostración. La demostración se basa en la construcción de la función f arriba descrita. Si la historia local del procesador i es vacía, para toda $x \in [0,t]$ definimos $f(x) = x \cdot \frac{t'}{t}$ que se encuentra bien definida ya que $t \neq 0$ y que claramente cumple las propiedades del lema. Supongamos entonces que la historia local de i en el punto (r,t) es no vacía. Como todos los eventos junto con el tiempo local en que suceden están definidos por medio de la historia, en la construcción de la función f solamente hay que tener cuidado con acomodar correctamente el tiempo real en que suceden los eventos. Por definición de función de historia, existe un

conjunto finito no vacío $E_i = \{T_1, T_2, \ldots, T_k\}$ tal que, $T_1 < T_2 < \ldots T_k$; y para todo $e \in h$ con $e \neq \nu$, existe $T_j \in E_i$ con $h(T_j) = e$. Para cada $T_j \in E_i$ con $h(T_j) = e$, definimos a t_j como $real\_t_r(e)$ y a t'_j como $real\_t_{r'}(e)$. Nótese que por definición $t_1 = t_{\mathsf{start}}(i, r)$ y $t'_1 = t_{\mathsf{start}}(i, r')$. Similarmente, por definición de ejecución, se cumple que:

$$t_1 < t_2 < \ldots < t_k \tag{2.1}$$

$$t'_1 < t'_2 < \ldots < t'_k \tag{2.2}$$

Para cada $j = 1, \ldots, k-2$ y para toda $x \in [t_j, t_{j+1})$, definimos la función $f_j : [t_j, t_{j+1}) \to [t'_j, t'_{j+1})$ de manera explícita por:

$$f_j(x) = t'_j + \left((x - t_j) \cdot \frac{t'_{j+1} - t'_j}{t_{j+1} - t_j}\right)$$

Definimos también la función $f_0 : [0, t_1) \to [0, t'_1)$ y la función $f_k : [t_{k-1}, t_k] \to [t_{k-1}, t_k]$ para toda $y \in [0, t_1)$ y para toda $x \in [t_{k-1}, t_k]$ explícitamente por las siguientes reglas:

$$\begin{aligned} f_0(y) &= y \cdot \frac{t'_1}{t_1} \\ f_k(x) &= t_{k-1} + \left((x - t_k) \cdot \frac{t'_k - t'_{k-1}}{t_k - t_{k-1}}\right) \end{aligned}$$

Nótese que las condiciones 2.1 y 2.2 implica que para cada $j = 2, \ldots, k$, cada $t_{j-1} - t_j \neq 0$ y $t'_{j-1} - t'_j \neq 0$, de donde cada f_j se encuentra bien definida, es continua y tiene inversa. Por último, la función f se define como la unión de las funciones arriba definidas. Formalmente, $f = \cup_{j=0}^{k} f_k$. La función f así definida es continua e invertible. Por último, por construcción, f satisface que $r_i(x) = r'_i(f(x))$, para todo $x \in [0, t]$ lo cual concluye la demostración del lema. ∎

La demostración del lema 2.1.1 muestra en un sentido preciso que la definición de sistema de paso de mensajes es muy débil para el estudio de eventos que dependen del tiempo real del sistema. Intuitivamente, podemos ver a cada una de las funciones f_j descritas en la demostración, como una transformación del tiempo real de una ejecución en otra; sin embargo, dicha transformación puede deformar el tiempo real prácticamente de cualquier forma, lo cual impide a los procesadores recuperar la más mínima información acerca del tiempo real. En la siguiente sección se imponen condiciones adicionales al sistema por medio de las cuales se captura una mejor representación para problemas que dependen del tiempo real.

2.2 Sistemas Parcialmente Síncronos

Los sistemas parcialmente síncronos se pueden definir informalmente como un sistema de paso de mensajes donde existen garantías *a priori* no triviales acerca de su comportamiento con respecto al tiempo real. Dichas garantías se dividen en dos componentes básicas: el comportamiento de los relojes locales y el comportamiento de las líneas de comunicación. Nótese que la definición de sistema de paso de mensajes nos impide acotar el avance de los relojes locales con respecto al tiempo real. Intuitivamente, la condición MP1 en la definición de un sistema de paso de mensajes permite modelar relojes que deforman el tiempo casi de cualquier forma; por ejemplo, la función de reloj local puede permanecer por debajo de una cota finita mientras el tiempo real se va hacia infinito (véase la figura 2.1).

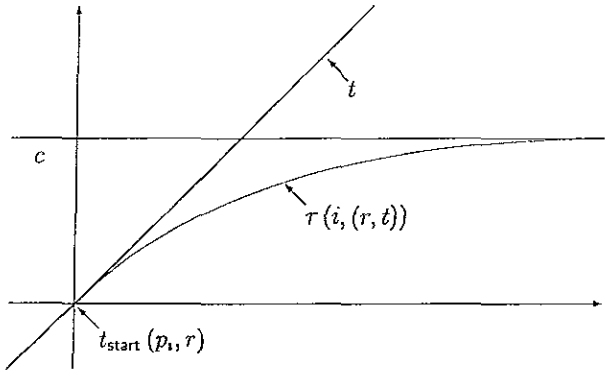


Figura 2.1: El reloj de p_i permanece por debajo de c mientras t se va a infinito.

Para modelar relojes físicos asumimos la existencia de una "envoltura lineal" que acota la posible diferencia de un reloj con respecto al tiempo real. Para representar dicha envoltura lineal es común asumir que existe una cota superior denotada por $\rho \geq 0$ en el desplazamiento *del reloj. Supongamos que* t, t' *son reales positivos, dos condiciones muy similares que* definen dicha envoltura lineal son:

(i) $(t - t')/(1 + \rho) \leq \tau(t) - \tau(t') \leq (t - t')(1 + \rho)$

(ii) $1/(1 + \rho) \leq d\tau(t)/dt \leq 1 + \rho$

Donde $d\tau(t)/dt$ representa la derivada de la función de reloj local con respecto al tiempo

real (véase [SWL88]). En este trabajo se usa la misma condición que en [PSR94] que es una generalización de las condiciones arriba presentadas. Formalmente:

Definición 2.2.1. *Sea i un procesador. Si para toda $r \in \mathcal{R}$ toda pareja de reales $0 \leq t \leq t' < \infty$ con $t_{\text{start}}(i,r) \leq t$ existen $\underline{\varrho}, \overline{\varrho}$ con $0 < \overline{\varrho} \leq \underline{\varrho} < \infty$ tal que*

$$\frac{t-t'}{\underline{\varrho}} \leq \tau(i,(r,t)) - \tau(i,(r,t')) \leq \frac{t-t'}{\overline{\varrho}}$$

entonces, el procesador i se dice que tiene un $(\underline{\varrho}, \overline{\varrho})$-reloj en $\mathcal{R}$. El reloj de un procesador i se dice que tiene tendencia acotada *en $\mathcal{R}$ si es un $(\underline{\varrho}, \overline{\varrho})$-reloj en $\mathcal{R}$. Un $(1,1)$-reloj se dice* libre de tendencia. *Diremos que un sistema $\mathcal{R}$* está acotado en la tendencia de los relojes *si todos los procesadores tienen un reloj de tendencia acotada en $\mathcal{R}$.*

Ejemplo 2.1.
Supongamos que el procesador i tiene un reloj cuyo fabricante garantiza que la desviación del mismo con respecto al tiempo real es de ± 1 cada 500 unidades de tiempo real. Modelamos esta garantía por medio de la definición anterior tomando a $\underline{\varrho}$ como $\frac{500}{499}$ y a $\overline{\varrho}$ como $\frac{500}{501}$. Si graficamos a $\frac{\underline{\varrho}}{t}$ y a $\frac{\overline{\varrho}}{t}$ (véase figura 2.2), podemos ver la envoltura lineal que acota al reloj en cuestión. En este sentido, siempre se cumple la desigualdad:

$$\frac{t}{\underline{\varrho}} \leq \tau(i,(t)) \leq \frac{t}{\overline{\varrho}}$$

Supongamos que en una ejecución r el la historia local del procesador i contiene a los eventos e^T y $e^{T'}$; (véase la figura 2.3). Supongamos además que dichos eventos suceden en los tiempos locales T y T' respectivamente. En ausencia de más información, el procesador i no puede distinguir cuanto tiempo real ha pasado exactamente en r entre el evento e y el evento e'. Nótese sin embargo, que por la garantía que ofrece el fabricante del reloj de i, dicho procesador podría deducir al menos que

$$\frac{T'-T}{\underline{\varrho}} \leq real\_t_r\left(e^{T'}\right) - real\_t_r\left(e^{T}\right) \leq \frac{T'-T}{\overline{\varrho}}$$

Suponiendo que dicho procesador tuviera acceso a la información provista por la garantía de su reloj, lo cual es una suposición natural y común. ∎

En este trabajo asumimos también que existen cotas en el retardo de los mensajes del sistema. Para modelar esta idea, en [HM90] se dice que un sistema $\mathcal{R}$ tiene *retardo incierto*

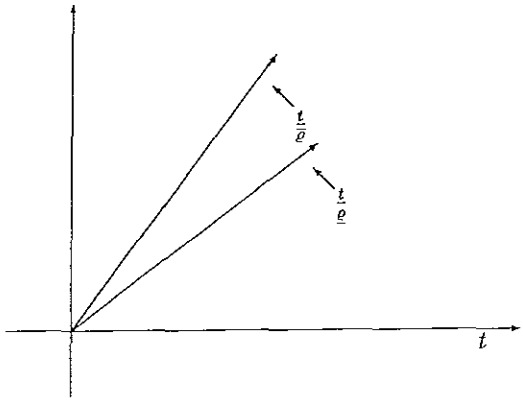


Figura 2.2: Envoltura lineal que acota a un reloj $(\underline{\varrho}, \overline{\varrho})$.

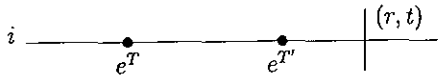


Figura 2.3: Diagrama de tiempo espacio para un punto (r,t) con un solo procesador i con dos eventos en su historia local.

pero acotado en la entrega de los mensajes si para toda línea de comunicación l en la gráfica subyacente existen cotas $L_l < H_l$ tal que el retardo de todos los mensajes transmitidos a través de la línea l está en el intervalo (L_l, H_l). Esta definición trata esencialmente igual a todos los mensajes que se transmiten a través de una misma línea. En este trabajo, siguiendo a [PSR94], supondremos que existen cotas en el retardo de cada mensaje, permitiendo tomar en cuenta la estructura interna de los mismos, como su tamaño o el tiempo local del procesador emisor. Formalmente:

Definición 2.2.2. *Sea $\mathcal{R}$ un sistema de paso de mensajes y sea e el evento de recepción de un mensaje m y e' su correspondiente envío. Si existen $L(m), H(m)$ con $0 \leq L(m) < H(m) \leq \infty$ tales que, para toda $r \in \mathcal{R}$ se cumple que $L(m) \leq real\_t_r(e) - real\_t_r(e') \leq H(m)$ entonces diremos que el mensaje m* está acotado en su retardo *en $\mathcal{R}$, por $L(m), H(m)$. Diremos que un sistema $\mathcal{R}$ está acotado en los mensajes si todo mensaje m está acotado en $\mathcal{R}$.*

Ejemplo 2.2.
Sea R un sistema con dos procesadores llamados s y v. Supongamos que el procesador v envía un mensaje m a s en un punto (r, t) y denotemos dicho evento por e (véase el diagrama de tiempo espacio de la figura 2.4). Supongamos además que el mensaje m está acotado en el retardo de su entrega por la pareja $(L(m), H(m))$ y que es recibido por s en su evento e'.

Dada las garantía de que el tiempo de tránsito de mensaje siempre es mayor que $L(m)$ y menor que $H(m)$, el procesador v puede deducir que, si el mensaje m fue recibido, entonces fue recibido en *en al menos* $L(m)$ unidades de tiempo real y en *a lo más* $H(m)$ unidades de tiempo real después de que es enviado; es decir, el mensaje m tuvo que ser recibido por s en el intervalo de tiempo real:

$$[real\_t_r(e) + L(m), real\_t_r(e) + H(m)]$$

En ausencia de más información, el procesador v no tiene forma deducir más acerca del tiempo que realmente tomó al mensaje m arribar a su destino en la ejecución r. ∎

Haciendo uso de las ideas anteriores , a continuación se define de manera formal el concepto de sistema parcialmente síncrono dentro de este modelo.

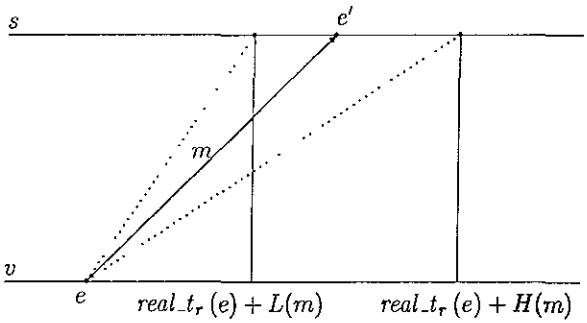


Figura 2.4: Las cotas provistas por las garantías de sistema aseguran que el tiempo total de tránsito del mensaje m es siempre mayor o igual que $real\_t_r\,(e) + L(m)$. Similarmente, el tiempo total de tránsito del mensaje m es siempre menor o igual que $real\_t_r\,(e) + H(m)$.

Definición 2.2.3 (Sistema parcialmente síncrono). *Un sistema $\mathcal{R}$ se dice parcialmente síncrono si y sólo si satisface*

PS1 $\mathcal{R}$ es un sistema de paso de mensajes donde los procesadores están equipados con relojes locales,

PS2 $\mathcal{R}$ está acotado en la tendencia de los relojes y

PS3 $\mathcal{R}$ está acotado en los mensajes.

Nótese que las cotas en el desplazamiento de los relojes así como las cotas en retardo de los mensajes, normalmente se conocen *a priori* y se definen en términos de las especificaciones de los componentes del sistema. Formalizamos esta idea en la siguiente capítulo.

Como se nota en [FHMV95], en cualquier sistema de paso de mensajes, los procesadores *conocen*[1], al menos, su historia local. En el caso particular de los sistemas parcialmente síncronos asumimos también que los procesadores conocen las garantías *a priori* del sistema,

[1] Usamos la palabra "conocen" de manera intuitiva. Una definición formal de conocimiento se da en el capítulo 4

sin embargo, los procesadores pueden conocer más cosas que les permitirían deducir información acerca del estado de los demás procesadores. Para eliminar cualquier conocimiento extra procedemos como en [FHMV95], consideramos posibles *todas* las ejecuciones consistentes con las condiciones PS1, PS2 y PS3. Formalmente, diremos que un conjunto V de historias locales es *cerrado bajo prefijos* si cada prefijo de h distinto de la función vacía se encuentra también en V. Sean $V_1, \ldots, V_n$ conjuntos de historias locales cerrados bajo prefijo para los procesadores $1, \ldots, n$ respectivamente. Definimos a $\mathcal{R}(V_1, \ldots, V_n)$ como todas las ejecuciones que satisfacen las condiciones PS1, PS2 y PS3 tales que la historia local de cada procesador i esta en V_i. En el resto de este trabajo, supondremos que los sistemas parcialmente síncronos son de la forma $\mathcal{R}(V_1, \ldots, V_n)$ para alguna elección de $V_1, \ldots, V_n$. Nótese que la elección se hace en términos del problema que se esta analizando.

2.3 Vistas

Para facilitar el estudio de los sistemas parcialmentre síncronos, abstraemos la información accesible a los procesadores en una ejecución por medio de gráficas dirigidas. Empezamos por definir la relación de *adyacencia* entre eventos de un sistema. Formalmente:

Definición 2.3.1. *Sea (r,t) un punto y $e, e' \in (r,t)$ dos eventos. Diremos que el evento e es* adyacente *en r al evento e' si:*

(i) e es el evento antecesor del evento e' en un mismo procesador.

(ii) e es un evento de envío de un mensaje y e' es su correspondiente recepción.

Diremos que el evento e es adyacente en un sistema $\mathcal{R}$ al evento e' si e es adyacente a e' para alguna ejecución $r \in \mathcal{R}$.

La cerradura transitiva de la relación de adyacencia entre eventos es equivalente a la relación *sucede antes* definida por L. Lamport en [Lam78]. Capturamos la misma idea con la noción de *alcanzabilidad* en una gráfica dirigida conocida como *vista* y definida para cada punto del sistema. Formalmente:

Definición 2.3.2 (Vista). *Sea $\mathcal{R}$ un sistema parcialmente síncrono y $(r,t) \in \mathcal{R}$ un punto. La* vista *del punto (r,t), denotada por $\mathcal{V}(r,t)$ es una gráfica dirigida (V,E) tal que existe*

una biyección $f_{\mathcal{V}(r,t)} : V \to \mathcal{E}(r,t)$ *y para todo* $(p,q) \in E$ *se cumple que el evento* $f_{\mathcal{V}(r,t)}(p)$ *es adyacente en* r *al evento* $f_{\mathcal{V}(r,t)}(q)$.

Como cada evento codifica el lugar y tiempo local de su ocurrencia y dada la definición de vista, asumimos, sin perder generalidad, que las vistas también codifican también dicha información Abusando un poco de la notación, hablaremos de eventos en vez de vértices cuando hablamos de vistas.

La definición de vista permite abstraer información temporal del sistema disponible para los procesadores. Sin embargo, como una vista está definida en términos de una porción completa de una ejecución, ésta puede contener información inaccesible a los procesadores. Esta idea motiva la definición de vista local, la cual, como se ve más adelante, caracteriza de manera precisa la porción de la vista a la cual tienen acceso los distintos procesadores del sistema. Formalmente:

Definición 2.3.3 (Vista Local). *Sea* $\mathcal{V}(r,t) = (V,E)$ *una vista de un punto* (r,t) *y sea* i *un procesador del sistema. Sea* e *el último evento en la historia local del procesador* i *el punto* (r,t). *La* vista local *del procesador* i *en la vista* $\mathcal{V}(r,t)$, *denotada por* $\mathcal{V}_i(r,t)$ *es la restricción de la vista* $\mathcal{V}(r,t)$ *a los eventos* $e' \in V$ *tales que el evento* e *es alcanzable desde* e' *en* $\mathcal{V}(r,t)$.

Ejemplo 2.3.
Para ejemplificar los conceptos de vista y vista local usaremos un sencillo sistema de tres procesadores, denotados por i, j y k, conectados dos a dos por una línea bidireccional de comunicación. En la figura 2.5 se muestra el diagrama espacio para un punto (r,t) de ejemplo. En este diagrama denotamos por i_n al evento n del procesador i (similarmente para j y k).

En la figura 2.6, se muestra la vista del punto (r,t), $\mathcal{V}(r,t)$ (nótese la cercana relación entre los conceptos de diagrama de tiempo espacio y vista de un punto). En la figura 2.7 se muestra la vista local $\mathcal{V}_i(r,t)$, del procesador i en el punto (r,t). ∎

Dados dos eventos e, e' y una ejecución r con $e, e' \in r$, si el evento e' es alcanzable en la vista $\mathcal{V}(r,t)$ para alguna t, lo denotamos por $e \to_r e'$, lo cual, en el sentido de la relación sucede antes se lee como *el evento* e *sucede antes que el evento* e'

Una importante consecuencia de las definiciones anteriores es que la vista local de un procesador puede ser la vista completa de la ejecución. Formalizamos esta idea por medio del siguiente teorema:

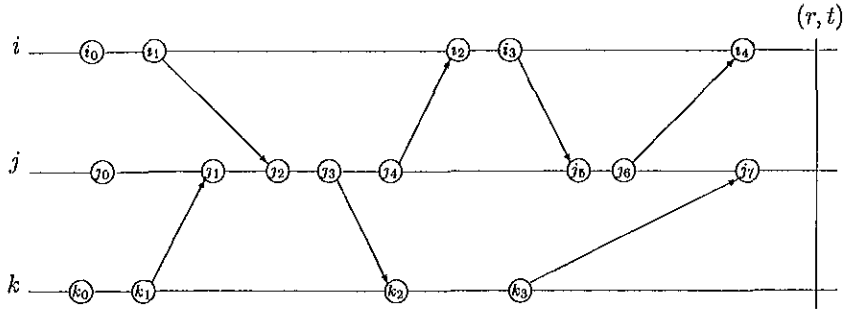


Figura 2.5: Diagrama de tiempo espacio para el punto (r, t) del ejemplo 2.3.

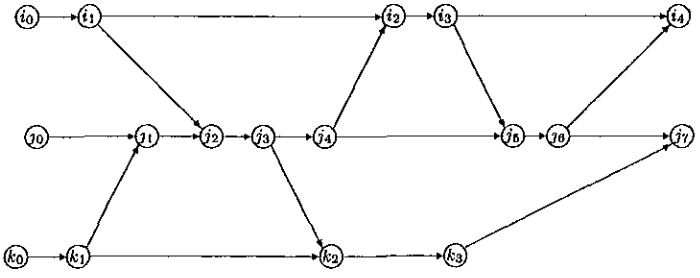


Figura 2.6: Vista del punto (r, t) mostrado en la figura 2.5.

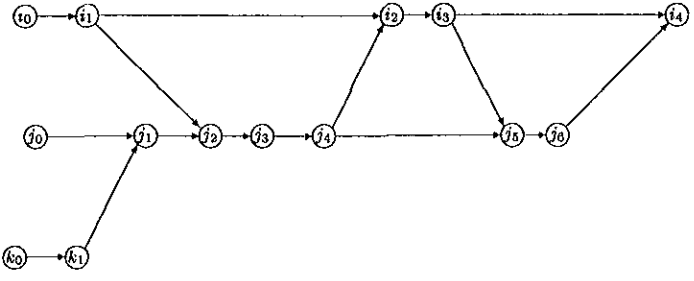


Figura 2.7: Vista local del procesador i en el punto (r, t) mostrado en la figura 2.5.

Teorema 2.3.1. *Sea $\mathcal{R}$ un sistema parcialmente síncrono. Sea i un procesador y $(r,t) \in \mathcal{R}$ un punto con $r_i(t) = (i, T, h)$. Supongamos que $h \neq \emptyset$.*

(a) Para todo punto $(r', t') \in \mathcal{R}$, con $r_i(t) = r'_i(t')$ se cumple que $\mathcal{V}_i(r,t) = \mathcal{V}_i(r', t')$.

(b) Existe un punto $(r', t') \in \mathcal{R}$, tal que $r_i(t) = r'_i(t')$ y $\mathcal{V}_i(r,t) = \mathcal{V}(r', t')$.

Demostración La parte (a) es una consecuencia directa de las definición de ejecución y sistema. Probaremos la parte (b) construyendo la ejecución r' con las propiedades que enuncia el teorema. En la figura 2.9 se muestra la misma construcción para el escenario mostrado en el diagrama de tiempo espacio de la figura 2.8. Sea $\mathcal{V}_i(r,t) = (V, E)$ la vista local del procesador i en el punto (r,t) y sea G el conjunto de todos los procesadores del sistema. Sea e el último evento en la historia local del procesador i en el punto (r,t). Para toda $j \in G$ definimos el conjunto $L_j = \{e' \in \mathcal{E}(r,t) \mid e' \rightarrow_r e\}$. Nótese que por definición, $\cup_{j \in G} L_j = f_{\mathcal{V}_i(r,t)}[V]$[2], donde $f_{\mathcal{V}_i(r,t)}$ es la biyección existente, por definición, entre los vértices de la gráfica y los eventos del sistema. Definimos para cada $j \in G \backslash \{i\}$ a t_j como 0 si $L_j = \emptyset$ y como $t_j = \max\{t'_j \mid e' \in L_j \text{ y } t'_j = real\_t_r(e')\}$. Para el procesador i, definimos a t_i como t Definimos la ejecución r' para cada $j \in G$ formalmente por:

(i) Para toda $x \in [0, t_j]$, $r'_j(x) = r_j(x)$ y

(ii) Para toda $y > t_j$ con $r_j(y) = (i, T, h_j)$, $r'_j(y) = (i, T, h'_j)$, con $h'_j(T) = \nu$ si $h_j \neq \emptyset$. Si $h_j = \emptyset$ entonces $h'_j = \emptyset$

La ejecución r' así definida satisface que $\cup_{j \in G} L_j = \mathcal{E}(r', t)$ y como $\cup_{j \in G} L_j = f_{\mathcal{V}_i(r,t)}$, directamente obtenemos que $\mathcal{V}(r', t) = \mathcal{V}_i(r,t)$. Veamos por último que la ejecución r' así definida satisface también las condiciones en la definición de ejecución de sistema parcialmente síncrono. Como todos los eventos en r' conservan el mismo tiempo local y real de r, y similarmente, como los relojes locales de los procesadores funcionan igual en r y en r', la ejecución r' es consistente con las condiciones PS2 y PS3 por suponer a $r \in \mathcal{R}$. Similarmente, por construcción, r' cumple la condición MP1 en la definición de sistema de paso de mensajes. Veamos por último que r' satisface la condición MP2. Sea $e' \in (r', t)$ un evento de recepción de mensaje. Como $e' \in (r', t)$, por definición de r' se sigue que $e' \rightarrow_r e$ y como $r \in \mathcal{R}$, por la condición MP2, existe un evento $e'' \in (r, t)$ tal que e'' es el envío del mensaje

[2]Usamos la notación $f[V]$ para denotar el conjunto generado al aplicar la función f a cada uno de los elementos en V.

recibido en el evento e'. Por definición de la relación de adyacencia entre eventos se sigue que e'' es adyacente a e' en r y por lo tanto e es alcanzable desde e'' en r. Por la definición de r', obtenemos que $e'' \in (r', t)$, de donde podemos concluir que $r' \in \mathcal{R}$, con lo cual se concluye la demostración del teorema. ∎

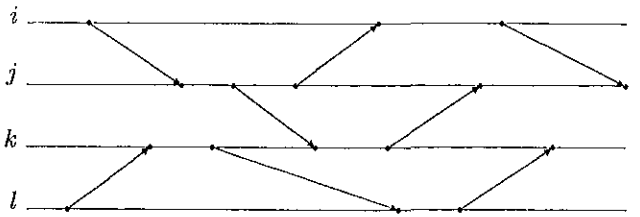


Figura 2.8: Escenario de ejemplo en la demostración del teorema 2.3.1.

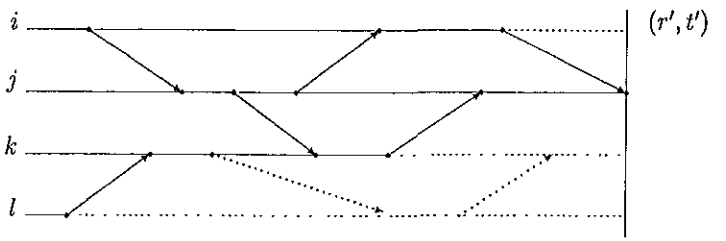


Figura 2.9: Construcción de la ejecución r' para el escenario de la figura 2.8. Las líneas punteadas horizontales representan el intervalo que se "elimina" de la ejecución r en la ejecución r'. Los mensajes representados por líneas punteadas no son recibidos en la ejecución r' y en algunos casos ni siquiera son enviados.

El teorema 2.3.1 implica una importante propiedad del sistema: las acciones de un procesador en cada punto del sistema dependen *únicamente* de la vista local del mismo en dicho punto. Esta idea se explora con mayor profundidad en el capítulo 5, por lo pronto hacemos la siguiente observación misma que es usada en dicho también en el capítulo 5:

Observación 2.3.2. *Sea $\mathcal{R}$ un sistema parcialmente síncrono y $(r, t), (r', t') \in \mathcal{R}$ dos puntos. Las siguientes afirmaciones son equivalentes:*

(i) $\mathcal{V}_i(r,t) = \mathcal{V}_i(r',t')$

(ii) $r_i(t) = r_i'(t')$

Capítulo 3

Gráficas de Sincronización

3.1 Representación de las Especificaciones de Tiempo Real

Modelamos las especificaciones de tiempo real dadas en la definición de sistemas parcialmente síncronos por medio de una función la cual se denomina garantía de sistema. Intuitivamente las garantías de sistemas acotan, haciendo uso de la especificación "temporal" del sistema, los eventos adyacentes para cada ejecución. Formalmente:

Definición 3.1.1 (Garantía de Sistema). *Una* garantía de sistema *para un sistema parcialmente síncrono (p.s) $\mathcal{R}$ es una función $\Sigma : \mathcal{E} \times \mathcal{E} \to \mathbb{R}$ tal que todo par de eventos e, e', la pareja $(e, e') \in \mathrm{dom}(\Sigma)$ si y sólo si e es adyacente en $\mathcal{R}$ a e' o si e' es adyacente en $\mathcal{R}$ a e. Una ejecución τ se dice* consistente *con Σ si y sólo si $real\_t_\tau(e) + \alpha \leq real\_t_\tau(e')$ para cada pareja de eventos e, e' adyacentes en τ para los cuales $\Sigma(e, e') = \alpha$.*

La definición anterior no se encuentra necesariamente relacionada con las especificaciones de tiempo real de un sistema parcialmente síncrono. La conexión se hace por medio de las garantías de sistema *estándar*, las cuales se definen en términos de las especificaciones dadas para un sistema. A continuación se presenta la definición formal.

Definición 3.1.2 (Garantía de Sistema Estándar). *Sea Σ una garantía de sistema para un sistema $\mathcal{R}$. Diremos que Σ es la* garantía de sistema estándar *de $\mathcal{R}$ si y sólo si se cumple que, para toda pareja de eventos e, e'*

(i) Si $e = e_i^T$ *y* $e' = e_i^{T'}$ *son dos eventos de un mismo procesador con un* $(\underline{\varrho}, \overline{\varrho})$*-reloj, y adyacentes en* $\mathcal{R}$*, se tiene que* $\Sigma(e, e') = \left(\frac{T'-T}{\overline{\varrho}}\right)$ *y* $\Sigma(e', e) = -\left(\frac{T-T'}{\underline{\varrho}}\right)$.

(ii) Si $e = (i, T, send(j, m))$ *y* $e' = (j, T', receive(i, m))$ *con* $L(m), H(m)$ *las cotas en el retardo del mensaje* m *entonces* $\Sigma(e, e') = L(m)$ *y* $\Sigma(e', e) = -H(m)$

Las garantías de sistema nos permiten modelar de manera sencilla y uniforme las especificaciones de tiempo real de un sistema parcialmente síncrono y tienen la interesante propiedad de estar definidas en términos de información localmente disponible a los procesadores, como son los eventos y las cotas del sistema. Una consecuencia de la definición de garantía de sistema es que dados dos eventos $(e, e') \in dom(\Sigma)$, con $\Sigma(e, e') \geq 0$, podemos decir que el evento e' sucede *al menos* $\Sigma(e, e')$ unidades de tiempo real después que el evento e. En el sentido contrario, si $\Sigma(e', e) \leq 0$, podemos decir que el evento e' sucede *a lo más* $-\Sigma(e', e)$ unidades de tiempo real después que el evento e. Es decir, dada una garantía de sistema Σ y una ejecución r consistente con Σ, para toda pareja de eventos $e, e' \in r$ con $(e, e') \in \text{dom}(\Sigma)$ se cumple que $real\_t_r(e) - real\_t_r(e') \in [\Sigma(e, e'), -\Sigma(e', e)]$.

Ejemplo 3.1.

Supongamos que tenemos un sencillo sistema p.s. de dos procesadores llamados s y v el cual satisface a una garantía de sistema Σ. Supongamos también que el procesador s tiene un reloj libre de tendencia y que v tiene un $(\underline{\varrho}, \overline{\varrho})$–reloj. Supongamos que tenemos un punto (r, t) como el que se muestra en el diagrama de tiempo espacio de la figura 3.1.

Nótese que como los eventos $e_v^{T_1}$ y $e_v^{T_4}$ son adyacentes debemos tener que la pareja $(e_v^{T_1}, e_v^{T_4}) \in \text{dom}(\Sigma)$. Como el reloj del procesador v tiene cotas definidas por la pareja $(\underline{\varrho}, \overline{\varrho})$ tenemos que $\Sigma(e_v^{T_1}, e_v^{T_4}) = \frac{T_4-T_1}{\underline{\varrho}}$; similarmente se cumple que $\Sigma(e_v^{T_4}, e_v^{T_1}) = -\frac{T_1-T_4}{\overline{\varrho}}$, de donde podemos deducir que:

$$real\_t_r\left(e_v^{T_1}\right) + \frac{T_4 - T_1}{\underline{\varrho}} \leq real\_t_r\left(e_v^{T_4}\right)$$

$$real\_t_r\left(e_v^{T_4}\right) - \frac{T_1 - T_4}{\overline{\varrho}} \leq real\_t_r\left(e_v^{T_1}\right)$$

De las dos desigualdades anteriores podemos concluir que:

$$\frac{T_4 - T_1}{\underline{\varrho}} \leq real\_t_r\left(e_v^{T_4}\right) - real\_t_r\left(e_v^{T_1}\right) \leq \frac{T_4 - T_1}{\overline{\varrho}}$$

En la siguiente sección continuamos el desarrollo del presente ejemplo. ∎

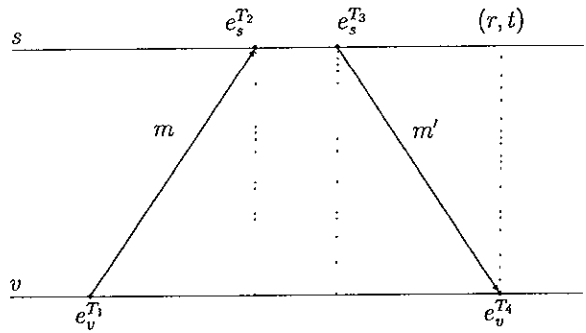


Figura 3 1: Diagrama de tiempo espacio para el punto (r,t) del ejemplo 3.1.

El siguiente lema enuncia una propiedad directa de la definición de garantía de sistema estándar.

Lema 3.1.1. *Sea $\mathcal{R}$ un sistema parcialmente síncrono y Σ su garantía de sistema estándar. Toda ejecución $r \in \mathcal{R}$ es consistente con Σ.*

Demostración. Directa de las definiciones ∎

Dado un sistema parcialmente síncrono $\mathcal{R}$ y su garantía de sistema estándar Σ, diremos que la pareja $(\mathcal{R}, \Sigma)$ es un *sistema garantizado*. Dada una garantía de sistema Σ, diremos que el sistema $\mathcal{R}$ que *representa* a Σ es el conjunto de todas las ejecuciones consistentes con Σ Una garantía de sistema se dice *satisfacible* si el sistema $\mathcal{R}$ que la representa es no vacío e *insatisfacible* de otra forma.

3.2 Gráficas de Sincronización

En esta sección relacionaremos los conceptos de garantía de sistema con la vista de un punto mediante el concepto de *gráfica de sincronización* [PSR94], lo cual nos permite generalizar de manera formal las cotas que proveen las garantías de sistema. Empezamos con la definición:

Definición 3.2.1. *Sea $(\mathcal{R}, \Sigma)$ un sistema garantizado. Sea $(r,t) \in \mathcal{R}$ un punto y $\mathcal{V}(r,t) =$*

(V, E) su vista. La gráfica de sincronización del punto (r, t), denotada por $\Gamma(r, t)$ es una pareja $(\widehat{G}, w)$, donde $\widehat{G} = (\widehat{V}, \widehat{E})$ es una gráfica dirigida y $w : V \to \mathbb{R}$ es una función de pesos, donde $\widehat{V} = V$ y el conjunto de arcos $\widehat{E}$ es tal que $(p, q) \in \widehat{E}$ si y sólo si $(p, q) \in E$ o $(q, p) \in E$. La función de pesos se define por $w(p, q) = \Sigma\left(f_{\widehat{G}}(p), f_{\widehat{G}}(q)\right)$.

Definida de esta forma, una gráfica de sincronización se puede ver como la versión con pesos de la vista con la cual se define.

Ejemplo 3.2 (Continuación del ejemplo 3.1).
Continuando con el ejemplo 3.1, supongamos además que el mensaje m cuyo envío se muestra en el diagrama de tiempo espacio de la figura 3.1, esta acotado en su retardo por $L(m), H(m)$. Similarmente, supongamos que el mensaje m' se encuentra acotado por la pareja $L(m'), H(m')$. La gráfica de sincronización del punto (r, t) se muestra en la figura 3.2. ∎

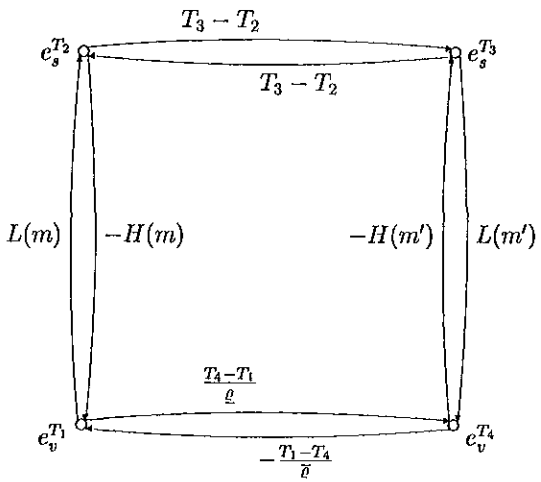


Figura 3.2: Gráfica de sincronización para el punto (r, t) mostrado en el diagrama de tiempo espacio de la figura 3.1.

El siguiente lema se puede ver como una generalización de la definición de consistencia de una ejecución y es una primera e importante propiedad de las gráficas de sincronización.

Lema 3.2.1. *Sea $(\mathcal{R}, \Sigma)$ un sistema garantizado. Sea $(r, t) \in \mathcal{R}$ un punto y $\Gamma(r, t)$ su gráfica*

de sincronización. Para toda pareja de eventos e, e', si existe camino dirigido con peso α de e a e' entonces $real\_t_r(e) + \alpha \leq real\_t_r(e')$.

Demostración. Sea $r \in \mathcal{R}$ una ejecución y supongamos que r es consistente con Σ. Sea $t \in \mathbb{R}^+$. Probaremos por inducción en la cardinalidad de c, un camino de peso α de e a e' en la gráfica de sincronización $\Gamma(r,t)$, que $real\_t_r(e) + \alpha \leq real\_t_r(e')$.

Si $|c| = 1$ el lema se sigue de suponer a r consistente con Σ. Sea $n > 1$ y supongamos por inducción que para *todo camino* c' de e a e' *con peso* $\alpha_{|c'|}$ y con $|c'| < n$ se cumple que $real\_t_r(e) + \alpha_{|c'|} \leq real\_t_r(e')$. Sea c un camino de e a e' con $|c| = n$ y supongamos que $c = \langle e_0, e_1, \ldots, e_n \rangle$ con $e_0 = e$ y $e_n = e'$. Sea $c_0 = \langle e_0, e_1, \ldots, e_{n-1} \rangle$ y sea $c_1 = \langle e_{n-1}, e_n \rangle$. Sea $\beta = \sum_{i=0}^{n-2} w(e_i, e_{i+1})$. Como $|c_0| = n - 1 < n$, por hipótesis de inducción se cumple:

$$real\_t_r(e_0) + \beta \leq real\_t_r(e_{n-1}) \tag{3.1}$$

Similarmente, como $|c_1| = 1$ y como r es consistente con Σ se cumple:

$$real\_t_r(e_{n-1}) + w(e_{n-1}, e_n) \leq real\_t_r(e_n) \tag{3.2}$$

Juntando (3.1) y (3.2) obtenemos que $real\_t_r(e_0) + \alpha_{|c|} \leq real\_t_r(e_n)$ con $\alpha_{|c|} = \beta + w(e_{n-1}, e_n)$, como se quería. ∎

El lema anterior implica que si existen dos caminos con pesos α, β de un evento e a un evento e' en una gráfica de sincronización, entonces $real\_t_r(e) + \gamma \leq real\_t_r(e')$, con $\gamma = \max\{\alpha, \beta\}$. Veamos que el camino de peso máximo se encuentran bien definido, para cada pareja de eventos conectados en las gráficas de sincronización, notando primero que los ciclos en dichas gráfica no tienen peso positivo, como a continuación se prueba:

Proposición 3.2.2. *Sea $(\mathcal{R}, \Sigma)$ un sistema garantizado. Para todo punto (r,t), su gráfica de sincronización no contiene ciclos de peso positivo.*

Demostración. Sea $(r,t) \in \mathcal{R}$ un punto y sea $c = \langle e_0, \ldots, e_k = e_0 \rangle$ un ciclo dirigido en la

gráfica de sincronización $\Gamma(r,t)$. De la definición de peso se sigue que:

$$\begin{aligned} w(c) &= \sum_{i=0}^{k-1} w(e_i, e_{i+1}) \quad \text{(Por definición)} \\ &\leq \sum_{i=0}^{k-1} (real\_t_r(e_{i+1}) - real\_t_r(e_i)) \quad \text{(Por el lema 3.2.1)} \\ &= real\_t_r(e_0) - real\_t_r(e_0) \\ &= 0 \ \blacksquare \end{aligned}$$

Definimos la distancia entre dos eventos en las gráficas de sincronización denotada por $d_{\Gamma(r,t)}$, como el peso de un *camino de peso máximo* o como $-\infty$ si no existe camino entre dichos eventos. Veamos que la distancia en las gráficas de sincronización caracteriza de manera precisa las cotas que el sistema especifica en términos del tiempo real para cualesquiera dos eventos conectados en la gráfica. Formalmente:

Teorema 3.2.3. *Sea Σ una garantía de sistema satisfacible (GSS). La ejecución r es consistente con Σ si y sólo si para toda $t \in \mathbb{R}^+$ y para toda pareja de eventos $e, e' \in (r,t)$ se cumple que $real\_t_r(e) + d_{\Gamma(r,t)}(e,e') \leq real\_t_r(e')$.*

Demostración. Sea Σ una garantía de sistema satisfacible y r una ejecución. Supongamos primero que r es consistente con Σ y sean $e, e' \in (r,t)$ dos eventos. Si no existe un camino entre e y e', tenemos que $d_{\Gamma(r,t)}(e,e') = -\infty$ y la afirmación se sigue de manera trivial. De otra forma, sea $c = e_0, \ldots, e_m$ con $e_0 = e$ y $e_m = e'$ un camino de peso máximo de e a e', entonces tenemos que

$$\begin{aligned} d_{\Gamma(r,t)}(e,e') &= \sum_{k=0}^{m} w(e_{k-1}, e_k) \\ &\leq \sum_{k=0}^{m} real\_t_r(e_k) - real\_t_r(e_{k-1}) \\ &= real\_t_r(e_m) - real\_t_r(e_0) \end{aligned}$$

con lo cual queda demostrada la parte del "sólo si".

Para la parte del "si", supongamos que para todo t y para todo par de eventos $e, e' \in (r,t)$, $real\_t_r(e) + d_{\Gamma(r,t)}(e,e') \leq real\_t_r(e')$ y sean $e, e' \in (r,t)$ tal que $(e,e') \in \mathrm{dom}(\Sigma)$. Por definición de la función de pesos y distancia, tenemos que $\Sigma(e,e') = w(e,e') \leq d_{\Gamma(r,t)}(e,e')$ y como por suposición $d_{\Gamma(r,t)}(e,e') \leq real\_t_r(e') - real\_t_r(e)$, obtenemos que $real\_t_r(e) + \Sigma(e,e') \leq real\_t_r(e')$, como se quería. $\blacksquare$

Para poder usar cotas provistas por la definición de distancia en las gráficas de sincronización tenemos que ver que estas son estrechas, es decir, que existen ejecuciones consistentes con su garantía de sistema que alcanzan dichas cotas. A continuación probaremos este importante resultado para la optimalidad de protocolos de sincronización. Primero introducimos notación adicional para manejar las distancias no finitas en la gráfica de sincronización, de manera similar a como se hace en [PSR94].

Definición 3.2.2. *Sea* $(\mathcal{R}, \Sigma)$ *un sistema garantizado. Sea* $(r,t) \in \mathcal{R}$ *un punto,* $e \in (r,t)$ *un evento y sea* $N \in \mathbb{R}$.

(i) Una ejecución r' *se dice una* N ejecución extrema hacia e *si para todo* $e' \in (r,t)$, *si* $d_{\Gamma(r,t)}(e',e) > -\infty$, *entonces* $real\_t_{r'}(e') + d_{\Gamma(r,t)}(e',e) = real\_t_{r'}(e)$; *de otra forma,* $real\_t_{r'}(e') + N > real\_t_{r'}(e)$.

(ii) Una ejecución r' *se dice una* N ejecución extrema desde e *si para todo* $e' \in (r,t)$, *si* $d_{\Gamma(r,t)}(e,e') > -\infty$, *entonces* $real\_t_{r'}(e) + d_{\Gamma(r,t)}(e,e') = real\_t_{r'}(e')$; *de otra forma,* $real\_t_{r'}(e) + N > real\_t_{r'}(e')$.

Una N ejecución extrema hacia un evento e es donde *la distancia en términos de tiempo real* de cada evento e' conectado con e en la gráfica de sincronización por medio de un camino, se define exactamente como la distancia en la misma gráfica. Para los eventos que no se encuentran conectados con e, la distancia en términos de tiempo real es mayor que N. La definición de una N ejecución extrema desde e es para el caso de los eventos e' con que e se conecta por medio de un camino.

Ejemplo 3.3.
Para ejemplificar el concepto de N ejecución extrema, pensemos en un sistema p.s. con cuatro procesadores llamados s_1, v_1, s_2, v_2 En el diagrama de tiempo espacio de la figura 3.3 muestra el escenario del ejemplo.

Nótese que no existen eventos que se relacionen causalmente entre el conjunto de eventos de los procesadores s_1, v_1 y el conjunto de eventos de los procesadores s_2, v_2. Por la definición de gráfica de sincronización, la observación anterior se reduce a que no existe un camino entre los eventos de los procesadores s_2, v_2 y, en particular, el evento $e_{v_1}^{T_3}$.

Sea $N > 0$ y sea M tal que $M > N$. En el diagrama de tiempo espacio de la figura 3.4 se muestra una ejecución N extrema hacia el evento $e_{v_1}^{T_3}$. Como sugiere la figura, los eventos de los procesadores s_1, v_1 se han acercado *lo más posible* al evento $e_{v_1}^{T_3}$. Similarmente, como no

existe una relación causal entre ellos, podemos separar en a los eventos de los procesadores s_2, v_2 por una cantidad *arbitraria* de tiempo real del evento $e_{v_1}^{T_3}$ sin que este cambio nos lleve una inconsistencia. ∎

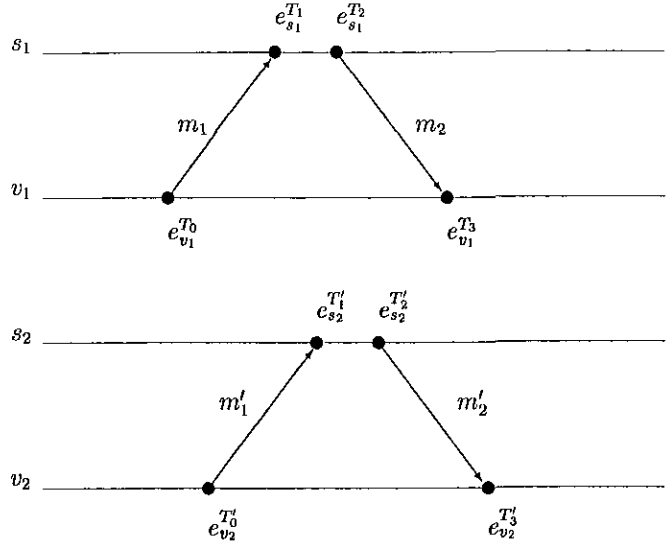


Figura 3.3: Escenario para el sistema del ejemplo 3.3.

A continuación se presenta un importante resultado para los problemas de sincronización. El siguiente teorema, que es en esencia el teorema 3.6 de [PSR94], propone que las ejecuciones N extremas no solo existen, sino que también representan ejecuciones válidas y consistentes con su correspondiente garantía de sistema. Formalmente:

Teorema 3.2.4. *Sea $(\mathcal{R}, \Sigma)$ un sistema garantizado. Para todo punto $(r,t) \in \mathcal{R}$ para todo evento $e \in (r,t)$ y para toda $N \in \mathbb{R}$, existen ejecuciones $r^0, r^1 \in \mathcal{R}$, y reales positivos t^0, t^1, tal que $\mathcal{V}(r,t) = \mathcal{V}(r^0,t^0) = \mathcal{V}(r^1,t^1)$ y tal que r^0 es una N ejecución extrema hacia e y r^1 es una N ejecución extrema desde e.*

Demostración. Sea $(\mathcal{R}, \Sigma)$ un sistema garantizado, sea $(r,t) \in \mathcal{R}$ un punto y $e \in (r,t)$ un evento y sea $N \in \mathbb{R}$. La demostración del teorema se basa en la construcción de las ejecuciones mencionadas. Aquí hacemos la construcción de r^0; la construcción de r^1 se puede

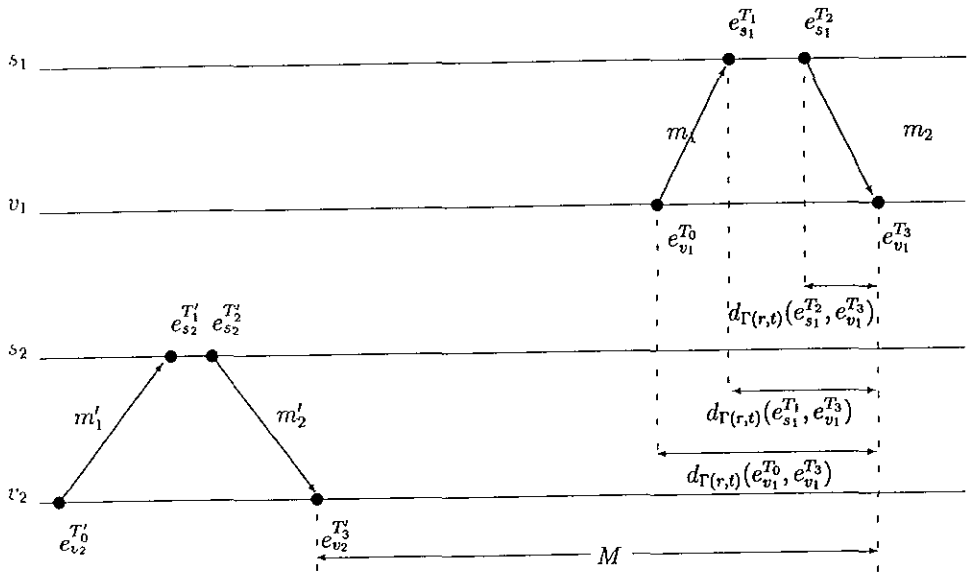


Figura 3.4: *Ejecución N extrema* para el escenario de la figura 3.3, donde $0 < N < M$.

hacer de manera similar. Sea $\Gamma(r,t) = ((V,E),w)$ la gráfica de sincronización del punto (r,t). Definimos una función $\mathcal{T}_0' : \mathcal{E}(r,t) \to \mathbb{R}$ explícitamente por:

$$\mathcal{T}_0'(e') = \begin{cases} real\_t_r\,(e) - d_{\Gamma(r,t)}(e',e) & \text{Si } d_{\Gamma(r,t)}(e,e') > -\infty \\ real\_t_r\,(e) + real\_t_r\,(e') - N & \text{de otra forma} \end{cases}$$

Sea $L_0 \geq 0$ tal que $L_0 > \min\{\mathcal{T}_0'(e') \mid e' \in \mathcal{E}(r,t)\}$; definimos la función $\mathcal{T}_0 : \mathcal{E}(r,t) \to \mathbb{R}^+$ como $\mathcal{T}_0(e) = \mathcal{T}_0'(e) + L_0$. Intuitivamente, la función $\mathcal{T}_0$ es el mapeo de tiempo real en la ejecución r^0 para cada evento del punto (r,t). Veamos primero que la función $\mathcal{T}_0$ arriba definida cumple las condiciones necesarias para ser una función de mapeo de tiempo real como el teorema requiere:

Afirmación 1. *La función $\mathcal{T}_0$ arriba satisface que para todo $e' \in \mathcal{E}(r,t)$*

(i) $\mathcal{T}_0(e') \geq 0$

(ii) Si $d_{\Gamma(r,t)}(e',e) > -\infty$ *entonces* $\mathcal{T}_0(e') + d_{\Gamma(r,t)}(e',e) = \mathcal{T}_0(e)$

(iii) Si $d_{\Gamma(r,t)}(e',e) = -\infty$ *entonces* $\mathcal{T}_0(e') + N > \mathcal{T}_0(e)$

La parte (i) se cumple por la suma del término L_0 en la definición de $\mathcal{T}_0$. Para la parte (ii), como $d_{\Gamma(r,t)}(e',e) > -\infty$, por definición $\mathcal{T}_0(e') = \mathcal{T}_0(e) - d_{\Gamma(r,t)}(e',e)$, de donde, $\mathcal{T}_0(e') + d_{\Gamma(r,t)}(e',e) = \mathcal{T}_0(e)$; por último, para la parte (iii) tenemos que $\mathcal{T}_0(e') = \mathcal{T}_0(e) + real\_t_r\,(e') - N$ de donde $\mathcal{T}_0(e') + N = \mathcal{T}_0(e) + real\_t_r\,(e')$. Por la definición de la función de tiempo real, $real\_t_r\,(e') > 0$, por lo cual $\mathcal{T}_0(e) + real\_t_r\,(e') > \mathcal{T}_0(e)$, por lo tanto se cumple que: $\mathcal{T}_0(e') + N > \mathcal{T}_0(e)$; como se quería.

El siguiente paso en la demostración es usar a la función $\mathcal{T}_0$ para mapear a la ejecución r en la ejecución r^0. Para este fin, definimos una transformación $m : \mathbb{R}^+ \times \mathcal{E}(r,t) \times \mathcal{E}(r,t) \to \mathbb{R}^+$ de manera explícita por:

$$m(x,e',e'') = real\_t_r\,(e') + (x - \mathcal{T}_0(e')) \cdot \frac{real\_t_r\,(e'') - real\_t_r\,(e')}{\mathcal{T}_0(e'') - \mathcal{T}_0(e')}$$

Para cada procesador $i = 1,\ldots,n$, definimos $E_i = \langle \mathsf{start}_i, e_{i,1}, \ldots, e_{i,k_i} \rangle$ como la secuencia ordenada (posiblemente vacía) de los eventos de cada procesador i del sistema en el punto

(r,t). Definimos la ejecución r^0 en cada procesador i con $E_i \neq \emptyset$ explícitamente por:

$$r_i^0(x) = \begin{cases} (i,0,\emptyset) & \text{si } x \in [0, \mathcal{T}_0(\mathsf{start}_i)) \\ r_i(m(x, \mathsf{start}_i, e_{i,1})) & \text{si } x \in [\mathcal{T}_0(\mathsf{start}_i), \mathcal{T}_0(e_{i,1})) \\ r_i(m(x, e_{i,1}, e_{i,2})) & \text{si } x \in [\mathcal{T}_0(e_{i,1}), \mathcal{T}_0(e_{i,2})) \\ \vdots & \\ r_i(m(x, e_{i,n_i-1}, e_{i,n_i})) & \text{si } x \in [\mathcal{T}_0(e_{i,n_i-1}), \mathcal{T}_0(e_{i,n_i})] \\ (i,T,h) & \text{si } x \in (\mathcal{T}_0(e_{i,k_i}), \infty) \text{ y } r_i(x) = (i,T,h') \text{ con } h(T) = \nu \end{cases}$$

Si $E_i = \emptyset$ entonces definimos $r_i^0(t) = (i,0,\emptyset)$, para toda $t \in \mathbb{R}^+$. La ejecución r^0 así definida satisface que para todo $t' > \max\{\mathcal{T}_0(e') \mid e' \in (r,t)\}$, la vista $\mathcal{V}(r^0,t')$ es igual a la vista $\mathcal{V}(r,t)$.

Veamos por último que la ejecución r^0 satisface a Σ. Sean $e', e'' \in (r,t)$ y supongamos que $d_{\Gamma(r,t)}(e',e'') = \alpha$ con $\alpha > -\infty$. Por el teorema 3.2.3, basta ver que $real\_t_{r^0}(e') + d_{\Gamma(r,t)}(e',e'') \leq real\_t_{r^0}(e'')$. Dividimos la prueba en dos casos; primero, cuando $d_{\Gamma(r,t)}(e',e) \neq -\infty$ y $d_{\Gamma(r,t)}(e'',e) \neq -\infty$. Sea c un camino de peso máximo de e' a e''. Por definición de la función $\mathcal{T}_0$ se cumple que:

$$\begin{aligned} d_{\Gamma(r,t)}(e',e) &= real\_t_r(e) - \mathcal{T}_0(e') \\ d_{\Gamma(r,t)}(e'',e) &= real\_t_r(e) - \mathcal{T}_0(e'') \end{aligned}$$

Y por definición de distancia se tiene que: $d_{\Gamma(r,t)}(e',e'') + d_{\Gamma(r,t)}(e'',e) \leq d_{\Gamma(r,t)}(e',e)$, de donde:

$$\begin{aligned} d_{\Gamma(r,t)}(e',e'') + real\_t_r(e) - \mathcal{T}_0(e'') &\leq real\_t_r(e) - \mathcal{T}_0(e') \\ \mathcal{T}_0(e') + d_{\Gamma(r,t)}(e',e'') &\leq \mathcal{T}_0(e'') \end{aligned}$$

El segundo caso es donde $d_{\Gamma(r,t)}(e',e) \neq -\infty$ y por lo tanto $d_{\Gamma(r,t)}(e'',e) \neq -\infty$. Por definición de $\mathcal{T}_0$ tenemos que:

$$\begin{aligned} \mathcal{T}_0(e') &= \mathcal{T}_0(e) + real\_t_r(e') - N \\ \mathcal{T}_0(e'') &= \mathcal{T}_0(e) + real\_t_r(e'') - N \end{aligned}$$

de donde obtenemos que $\mathcal{T}_0(e') - real\_t_r(e') = \mathcal{T}_0(e'') - real\_t_r(e'')$ *(i)*. Como r es consistente con Σ se cumple que

$$real\_t_r(e') + d_{\Gamma(r,t)}(e',e'') \leq real\_t_r(e'') \quad (ii)$$

Usando *(i)* y *(ii)*, directamente obtenemos que $\mathcal{T}_0(e') + d_{\Gamma(r,t)}(e', e'') \leq \mathcal{T}_0(e'')$ y por el teorema 3.2.3 y la definición de sistema parcialmente síncrono, se cumple que $r^0 \in \mathcal{R}$, lo cual concluye la demostración del teorema. ∎

Intuitivamente, el teorema 3.2.4 muestra que los extremos de las cotas provistas por las garantías de sistema para un punto se pueden alcanzar por ejecuciones del sistema, mismas que son indistinguibles para los procesadores, en el sentido de que dichas ejecuciones comparten la misma vista.

3.3 Definición Formal de los Enunciados de Precedencia Sincronizada

Usando la teoría desarrollada en las secciones anteriores, formalizamos la idea de enunciados de precedencia sincronizada de manera directa. Intuitivamente, usamos la notación $e \xrightarrow{\alpha} e'$ para denotar el hecho de que el evento e' sucede *al menos* α unidades de tiempo real después del e.

La notación $e \xrightarrow{\alpha} e'$ tiene sentido también para valores negativos de α. Intuitivamente, la interpretación del enunciado $e \xrightarrow{-|\beta|} e'$ es que el evento e *no puede suceder más de* $|\beta|$ unidades de tiempo real después de que e' lo hace. De esta forma, podemos usar los enunciados de precedencia sincronizada para especificar cotas de tiempo superiores e inferiores. Formalmente:

Definición 3.3.1 (Enunciados de Precedencia Sincronizada). *Sea $(\mathcal{R}, \Sigma)$ un sistema garantizado, sea $(r,t) \in \mathcal{R}$ un punto y e, e' una pareja de eventos. Usamos la notación*

$$\mathcal{V}(r,t) \vdash_\Sigma e \xrightarrow{\alpha} e'$$

siempre que $e, e' \in (r,t)$ y $d_{\Gamma(r,t)}(e, e') = \alpha$.

Por los teoremas 3.2.3 y 3.2.4, los enunciados de precedencia sincronizada así definidos satisfacen las propiedades arriba descritas. Adicionalmente, usamos la notación

$$\mathcal{V}(r,t) \vdash_\Sigma e \overset{\alpha}{\rightsquigarrow} e'$$

para denotar el hecho de que existe un camino de peso α, del evento e al evento e' en la gráfica de sincronización.

Capítulo 4

Modelo Formal de Conocimiento

4.1 Definición de Conocimiento

Para hablar del conocimiento de los procesadores de un sistema, asumimos la existencia de un lenguaje básico generador. Más formalmente, asociamos con un sistema $\mathcal{R}$ de n procesadores un conjunto Φ de *proposiciones básicas*. Estas incluyen afirmaciones acerca del estado del sistema. Por ejemplo, "la variable x del procesador i vale 3" o "el reloj local del procesador j marca T" pueden ser parte del conjunto Φ. Las proposiciones básicas se definen en términos de la aplicación específica.

Se extiende el lenguaje generador Φ tomando su mínima cerradura bajo los conectivos Booleanos estándar de el cálculo proposicional y mediante los operadores modales de conocimiento K_i para cada procesador i y E_G,C_G y D_G para cada *subconjunto no* vacío de procesadores G. A este nuevo lenguaje lo denotamos por $\mathcal{L}_n(\Phi)$.

Un *sistema interpretado* $\mathcal{I}$ es una pareja $(\mathcal{R}, \pi)$ donde $\mathcal{R}$ es un sistema y π es una *función de interpretación* sobre un conjunto de proposiciones básicas Φ. Formalmente, $\pi : \mathcal{G} \times \Phi \rightarrow \{true, false\}$.

Dado un sistema interpretado $\mathcal{I} = (\mathcal{R}, \pi)$, se define de la relación de satisfacibilidad "$\models$" que existe entre las fórmulas $\varphi \in \mathcal{L}_n(\Phi)$ y los puntos $(r, t) \in \mathcal{R}$ por inducción en la

complejidad de la fórmula. Formalmente:

$$\begin{array}{lll} (\mathcal{I},r,t) \models p \text{ (para } p \in \Phi) & \text{syss} & \pi(r(t),p) = true \\ (\mathcal{I},r,t) \models \varphi \wedge \psi & \text{syss} & (\mathcal{I},r,t) \models \varphi \text{ y } (\mathcal{I},r,t) \models \psi \\ (\mathcal{I},r,t) \models \neg\varphi & \text{syss} & (\mathcal{I},r,t) \not\models \varphi \\ (\mathcal{I},r,t) \models K_i\varphi & \text{syss} & (\mathcal{I},r',t') \models \varphi \text{ para todo } (r',t') \text{ tal que } r_i(t) = r_i'(t') \end{array}$$

Dado un sistema interpretado $\mathcal{I} = (\mathcal{R}, \pi)$, diremos que el procesador i *conoce* una fórmula $\varphi \in \mathcal{L}_n(\Phi)$ en un punto $(r,t) \in \mathcal{I}$ siempre que $(\mathcal{I},r,t) \models K_i\varphi$. La presente adscripción de conocimiento a los procesadores del sistema es una noción externa y no requerimos que el procesador pueda recuperar dicho conocimiento. Nótese sin embargo que dicha noción de conocimiento, tiene la interesante propiedad de estar definida únicamente en términos de información disponible al procesador, como lo es su estado local.

Una extensión del conocimiento de un procesador, es el conocimiento de un subconjunto no vacío G de procesadores. Pensemos primero la noción de un conjunto de procesadores en el cual todos conocen una fórmula φ, lo cual se denota por $E_G\varphi$ y se define directamente por

$$(\mathcal{I},r,t) \models E_G\varphi \quad \text{syss} \quad (\mathcal{I},r,t) \models K_i\varphi \text{ para todo } i \in G$$

Otra de las variantes del conocimiento de un grupo de procesadores es el conocimiento común. Intuitivamente el *conocimiento común* de una fórmula φ se puede ver como una conjunción infinita de enunciados del tipo "todos en G saben φ y todos en G saben que todos en G saben φ y todos ... ". Siguiendo esta idea, se define el operador de conocimiento común por

$$(\mathcal{I},r,t) \models C_G\varphi \quad \text{syss} \quad (\mathcal{I},r,t) \models E_G^k\varphi \text{ para toda } k \geq 1$$

donde el operador $E_G^k\varphi$ se define por inducción en k como:

$$\begin{array}{rll} (\mathcal{I},r,t) \models E_G^1\varphi & \text{syss} & (\mathcal{I},r,t) \models E_G\varphi \\ (\mathcal{I},r,t) \models E_G^{k+1}\varphi & \text{syss} & (\mathcal{I},r,t) \models E_G^k E_G\varphi \end{array}$$

El último de los operadores estándar es el *conocimiento distribuido* el cual se denota por D_G. La definición del conocimiento distribuido de un grupo de procesadores es una generalización directa de la definición del conocimiento para un procesador. Formalmente:

$$\begin{array}{lll} (\mathcal{I},r,t) \models D_G\varphi & \text{syss} & (\mathcal{I},r',t') \models \varphi \text{ para todo punto}(r',t') \in \mathcal{I} \text{ que cumple que} \\ & & r_i(t) = r_i'(t') \text{ para todo procesador } i \in G \end{array}$$

Intuitivamente, el conocimiento distribuido de un hecho φ entre los miembros de un grupo de procesadores G, es el mismo conocimiento que se tendría si se juntase el conocimiento de todos y cada uno de los miembros de G. Por ejemplo, si un procesador sabe φ y otro sabe que $\varphi \Rightarrow \psi$, entonces, entre los dos tienen conocimiento distribuido de ψ.

Diremos que una fórmula es válida en un sistema $\mathcal{I}$ y se denota por $\mathcal{I} \models \varphi$ siempre que $(r,t) \models \varphi$ para cada punto $(r,t) \in \mathcal{I}$. Diremos que una fórmula φ es válida y se denota por $\models \varphi$ siempre que $\mathcal{I} \models \varphi$ para todo sistema interpretado $\mathcal{I}$. En la siguiente sección hacemos una breve descripción de las propiedades formales del conocimiento.

4.2 Propiedades Formales del Conocimiento

Dentro de la teoría clásica de la lógica modal, dado un operador modal M, los esquemas de axioma que usualmente se consideran para el operador modal M son:

A1 $M\varphi \Rightarrow \varphi$

A2. $(M\varphi \wedge M(\varphi \Rightarrow \psi)) \Rightarrow M\psi$

A3. $M\varphi \Rightarrow MM\varphi$

A4. $\neg M\varphi \Rightarrow M\neg M\varphi$

junto con las reglas de inferencia

R1. De φ y $\varphi \Rightarrow \psi$ se infiere ψ (modus ponens)

R2. De φ se infiere $M\varphi$ (generalización)

Desde el punto de vista del conocimiento, el esquema A1 se conoce como *axioma del conocimiento* Intuitivamente, este esquema implica que sólo hechos verdaderos pueden conocerse; este es el esquema que distingue la noción de conocimiento de la de creencia. El esquema A2 se conoce como el *axioma de distribución* ya que permite distribuir un operador modal sobre la implicación. El esquema A3 se conoce como el *axioma de introspección positiva*, e implica que los procesadores saben que es lo que saben. Por último, el esquema A4 se conoce como el *axioma de introspección negativa*, e implica que los procesadores saben que es lo que no saben.

Desde el punto de vista clásico (véase [Che93]) y para el caso de un solo procesador, el sistema compuesto por el esquema A2 junto con las reglas de inferencia R1 y R2 se conoce como K. El sistema K+A1 se conoce como T, T+A3, es conocido como S4, mientras S4+A4 es llamado S5. En el caso de n procesadores con $n > 1$, estos sistemas se conocen por K_n, T_n, $S4_n$ y $S5_n$, respectivamente (véase [HM92, FHMV95]).

Dada la definición del operador K_i, éste satisface las propiedades del sistema de lógica modal $S5_n$, como el siguiente teorema propone:

Teorema 4.2.1. *Para todo sistema interpretado $\mathcal{I}$, para todo par de fórmulas φ y ψ, y para todo procesador i se cumple:*

(a) $\mathcal{I} \models (K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi$

(b) Si $\mathcal{I} \models \varphi$ *entones* $\mathcal{I} \models K_i\varphi$

(c) $\mathcal{I} \models K_i\varphi \Rightarrow \varphi$

(d) $\mathcal{I} \models K_i\varphi \Rightarrow K_iK_i\varphi$

(e) $\mathcal{I} \models \neg K_i\varphi \Rightarrow K_i\neg K_i\varphi$

Demostración. Sea $\mathcal{I} = (\pi, \mathcal{R})$ un sistema interpretado y sea $(r, t) \in \mathcal{R}$ un punto.

(a). Supongamos que $(\mathcal{I}, r, t) \models K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)$ y sea $(r', t') \in \mathcal{R}$ tal que $r'_i(t') = r_i(t)$. Por la definición del operador K_i, en (r', t') se cumplen que $(\mathcal{I}, r', t') \models \varphi$ y $(\mathcal{I}, r', t') \models (\varphi \Rightarrow \psi)$ de donde, directamente obtenemos $(\mathcal{I}, r', t') \models \psi$. Por la definición del operador de conocimiento, directamente se sigue $(\mathcal{I}, r, t) \models K_i\psi$, como se quería.

(b). Supongamos que $\mathcal{I} \models \varphi$ y sea $(r, t) \in \mathcal{R}$ un punto y i un procesador. Como $\mathcal{I} \models \varphi$, en particular, $(\mathcal{I}, r', t') \models \varphi$ para todo $(r', t') \in \mathcal{R}$ tal que $r_i(t) = r'_i(t')$, de donde, por la definición de conocimiento se sigue que $(\mathcal{I}, r, t) \models K_i\varphi$. Como (r, t) fue cualquiera, directamente obtenemos $(\mathcal{I}, r, t) \models K_i\varphi$.

(c). Supongamos que $(\mathcal{I}, r, t) \models K_i\varphi$. Por la definición del operador de conocimiento, para todo $(r', t') \in \mathcal{R}$ tal que $r_i(t) = r'_i(t')$ se cumple $(\mathcal{I}, r', t') \models \varphi$. Entonces, en particular para (r, t) tenemos que $(\mathcal{I}, r, t) \models \varphi$.

(d). Supongamos que $(\mathcal{I}, r, t) \models K_i\varphi$ y sea $(r', t') \in \mathcal{R}$ tal que $r_i(t) = r'_i(t')$. Por la definición del operador de conocimiento directamente obtenemos que $(\mathcal{I}, r', t') \models K_i\varphi$. Como (r', t') fue cualquiera, por la definición de conocimiento se sigue que $(\mathcal{I}, r, t) \models K_iK_i\varphi$.

(e). Supongamos que $(\mathcal{I}, r, t) \models \neg K_i\varphi$. Por definición del operador K_i existe un punto $(r'', t'') \in \mathcal{R}$ con $r_i''(t'') = r_i(t)$ y tal que $(\mathcal{I}, r', t') \models \neg\varphi$. Sea $(r', t') \in \mathcal{R}$ con $r_i'(t') = r_i(t)$. Como $r_i'(t') = r_i''(t'')$ y de la definición del operador K_i obtenemos que $(\mathcal{I}, r', t') \models \neg K_i\varphi$, y como (r', t') es cualquiera que satisface $r_i'(t') = r_i(t)$, concluimos que $(\mathcal{I}, r, t) \models K_i\neg K_i\varphi$, como se quería ∎

Para el caso del operador de conocimiento K_i, se puede mostrar que los axiomas A1–A4 junto con la regla de inferencia R2 proporcionan una axiomatización completa del sistema (véase [HM92, FHMV95]).

Los operadores E_G, C_G y D_G también cumplen las propiedades del *sistema modal* $S5_n$ y, adicionalmente, el conocimiento común satisface dos axiomas adicionales:

C1 El *axioma de punto fijo*: $C_G\varphi \equiv E_G(\varphi \wedge C_G\varphi)$.

C2 La *regla de inducción*: De $\varphi \Rightarrow E_G(\varphi \wedge \psi)$ se deduce $\varphi \Rightarrow C_G\psi$.

Un caso importante de la regla de inducción es cuando se toma ψ como φ. Como $E_G(\varphi \wedge \varphi)$ es equivalente a $E_G\varphi$ se obtiene que de $\varphi \Rightarrow E_G\varphi$ se puede inferir $\varphi \Rightarrow C_G\varphi$. Es interesante notar que este caso muestra que el conocimiento común requiere de sincronía en un sentido muy fuerte. Intuitivamente, para que una fórmula sea conocimiento común, se requiere que una vez que esta es veradera, todos los procesadores, simultáneamente y al mismo tiempo, tienen que saber la misma formula es verdadera. Para muchas fórmulas no triviales lo anterior es simplemente imposible. Para una discusión con más detalle de las propiedades del conocimiento común véase [HM90, FHMV95].

Claramente, las nociones de conocimiento en grupo introducidas anteriormente conforman una jerarquía, donde

$$C_G\varphi \Rightarrow \cdots \Rightarrow E_G^{k+1}\varphi \Rightarrow \cdots \Rightarrow E_G\varphi \Rightarrow D_G\varphi \Rightarrow \varphi$$

Para el caso de un procesador podemos ver que la jerarquía anterior se colapsa, de tal forma que $C\varphi \equiv E^k\varphi \equiv E\varphi \equiv D\varphi$. Sin embargo, para el caso de $n \geq 2$ procesadores sin memoria compartida, la jerarquía de arriba es estricta.

4.3 Operadores del Tiempo

Adicionalmente a los operadores de conocimiento anteriormente descritos, enriquecemos nuestro lenguaje con *operadores temporales*, lo cual nos permite modelar algunas sencillas propie-

dades temporales de los sistemas que estemos estudiando.

En este trabajo haremos uso de un solo operador temporal, denotado por $\Box$ ("siempre"), aunque muchas veces resulta útil definir algunos operadores adicionales como el dual de "siempre", denotado por $\Diamond$ ("eventualmente"). Definimos de manera formal el operador $\Box$ por:

$$(\mathcal{I}, r, t) \models \Box\varphi \quad \text{syss} \quad (\mathcal{I}, r, t') \models \varphi \text{ para toda } t' \geq t$$

A continuación, haciendo uso del operador $\Box$, definimos una importante propiedad en las fórmulas; propiedad que se estudia más a fondo en el siguiente capítulo.

Definición 4.3.1 (Fórmula estable). *Una fórmula φ se dice* estable *en un sistema interpretado $\mathcal{I}$ siempre que $\mathcal{I} \models \varphi \Rightarrow \Box\varphi$.*

En otras palabras, una fórmula estable una vez que es verdadera, permanece así por el resto de la ejecución. Nótese que cualquier tautología es una fórmula estable; existen sin embargo un sin fin de ejemplos no triviales como la proposición: *el reloj local de i ya marcó la lectura T*. Nótese que a partir del momento en que el reloj local de i marca T la proposición anterior es verdadera y permanece así por el resto de la ejecución. Es fácil ver también que si φ_1 y φ_2 son fórmulas estables para un sistema interpretado $\mathcal{I}$, también lo son $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$. Adicionalmente, si suponemos que el sistema tiene memoria perfecta, se puede probar que $K_i\varphi_1$ y $C_G\varphi_1$ son estables. Para el caso del operador de conocimiento distribuido D_G esto mismo no es necesariamente cierto, en particular no es cierto en los sitemas parcialmente síncronos. En el siguiente capítulo se ahonda en esta "deficiencia" del operador D_G.

Capítulo 5

Conocimiento en Sistemas Parcialmente Síncronos

5.1 Conocimiento Inherente

Muchas de las variantes del los problemas de sincronización, sobre todo aquellas que no dependen de información temporal exterior al sistema, se pueden resolver por medio de *relojes lógicos*. Para estos casos, es posible simplificar la noción de tiempo real, usando, por ejemplo, a los números naturales y pensando el reloj de cada procesador i como una función C_i que mapea eventos en números naturales y que satisface que para toda pareja de eventos e_i, e_j, si $e_i \rightarrow e_j$ entonces $C_i(e_i) < C_j(e_j)$. Existen algoritmos de sincronización sencillos y elegantes para relojes lógicos (véase [Lam78, AW98]), que usan el orden parcial generado por la relación "sucede antes". En [FHMV95], se prueba que para el caso de los sistemas completamente asíncronos, la mejor información que los procesadores pueden recuperar acerca de el orden relativo de los eventos, es la mediada por la relación $\rightarrow_\tau$. *Esto es*, aún *combinando el conocimiento de todos los procesadores* de una ejecución, los procesadores no podrían recuperar mayor información acerca del orden de los eventos que la implicada por la relación sucede antes. La idea de conocimiento combinado se captura, en este, caso por medio del operador de conocimiento distribuido D_G.

La definición de los sistemas parcialmente síncronos nos permite modelar información acerca de la sincronía relativa de los eventos. Por medio de los enunciados de precedencia sincronizada, es posible acotar la ocurrencia de los eventos en un punto del sistema. Para este caso, probaremos que la mejor información que los procesadores pueden recuperar acerca

de la sincronía relativa de los eventos, es la mediada por los enunciados de precedencia sincronizada; sin embargo, el conocimiento distribuido no es un alternativa adecuada para el análisis de este caso ya que su definición se basa en la ocurrencia *simultánea* de el estado local de un grupo de procesadores. Si los procesadores pudieran hacer un uso total de su conocimiento distribuido, entonces, podrían sincronizar sus relojes perfectamente, lo cual, como se sabe, es imposible, *sin importar la cantidad de información que los procesadores intercambien.* En la siguiente sección se presenta una noción alternativa al conocimiento distribuido, propuesta en [MB94], y que es adecuada para razonar acerca del conocimiento combinado de un conjunto de procesadores, mismo que puede ser usado por los procesadores.

5.1.1 Definición Formal y Propiedades

Pensemos que se quiere considerar la noción de "conocimiento combinado" de los procesadores; parece razonable entonces que el estado local de cada uno de ellos implique dicho conocimiento, sin embargo, queremos evitar que el conocimiento combinado implique una simple conjunción de enunciados como "el estado local del procesador i es s_i y el estado local del procesador j es s_j", asumiendo implícitamente la ocurrencia simultánea de dichos eventos. En [MB94], los autores definen una noción alternativa al conocimiento distribuido, conocida como conocimiento inherente, que permite razonar acerca del conocimiento combinado de los procesadores en sistemas con incertidumbre temporal. Formalmente:

Definición 5.1.1 (Conocimiento inherente). *Sea $\mathcal{I}$ un sistema interpretado, $G = \{1, 2, \ldots, n\}$ un conjunto de procesadores y φ una fórmula. Diremos que el conjunto G de procesadores tienen conocimiento inherente de la fórmula φ en el punto $(r, t) \in \mathcal{I}$ y lo denotamos por $I_G\varphi$ siempre que:*

$$(\mathcal{I}, r, t) \models I_G\varphi \quad syss \quad (\mathcal{I}, r', t') \models \varphi \text{ se cumple para todo } (r', t') \in \mathcal{I} \text{ tal que}$$
$$(\forall j \in G)(\exists t_j \leq t') \;\; r_j(t) = r'_j(t_j)$$

Es decir, una fórmula φ es concimiento inherente de un grupo de procesadores precisamente cuando la misma fórmula se cumple en todos los puntos en los cuales, en algún momento anterior, cada procesador i *tuvo* el mismo estado local que en el punto (r, t). Intuitivamente, el conocimiento inherente de un grupo de procesadores hace uso del estado local de cada procesador, sin asumir que los mismos suceden simultáneamente. Esta definición, restringe un poco las fórmulas que pueden ser conocimiento inherente, como el siguiente lema propone:

Lema 5.1.1. *Sea $\mathcal{I}$ un sistema interpretado. Para toda fórmula φ y para todo conjunto no vacío de procesadores G se cumple:*

$$\models I_G\varphi \Rightarrow \Box\varphi$$

Demostración. Sea $\mathcal{I}$ un sistema interpretado y G un conjunto de procesadores. Sea $(r_0, t_0) \in \mathcal{I}$ y supongamos que $(\mathcal{I}, r_0, t_0) \models I_G\varphi$. Queremos ver que, para toda $t \geq t_0$ se cumple: $(\mathcal{I}, r, t') \models \varphi$. Sea $t \geq t_0$. Nótese primero que $(\forall j \in G)(\exists t_j \leq t)$ tal que $r_j(t_0) = r_j(t_j)$ ya que podemos tomar $t_j = t_0$ y como $t_0 = t_j \leq t$, por la definición de conocimiento inherente y como por susposición $(\mathcal{I}, r_0, t_0) \models I_G\varphi$, tenemos que $(I, r_0, t) \models \varphi$. Como t es cualquiera, para toda $t \geq t_0$ se cumple $(I, r_0, t) \models \varphi$, como se quería. ∎

El lema 5.1.1 implica que ciertas fórmulas pueden ser conocidas por un procesador sin ser conocimiento inherente de cualquier conjunto G de procesadores, aun cuando G contiene a dicho procesador. Por ejemplo, el procesador puede saber que *la lectura de su reloj local es c*, sin embargo, el conocimiento inherente será solamente que *su reloj ya marcó la lectura c.* Entre otras cosas, el ejemplo anterior muestra una de las diferencias principales del conocimiento inherente con el conocimiento distribuido: para el conocimiento distribuido se tiene que $\models K_i\varphi \Leftrightarrow D_{\{i\}}\varphi$; es decir, el conocimiento distribuido del conjunto unitario $\{i\}$ coincide con el conocimiento del procesador i. En general, esto no se cumple para el conocimiento inherente. Nótese sin embargo que, $\mathcal{I} \models K_i\varphi \Leftrightarrow I_{\{i\}}\varphi$ se cumple para fórmulas φ que son estables en $\mathcal{I}$ como el siguiente lema propone:

Lema 5.1.2. *Sea $\mathcal{I}$ un sistema interpretado. Para toda fórmula φ estable en $\mathcal{I}$ y para todo procesador i se cumple:*

$$\mathcal{I} \models K_i\varphi \Leftrightarrow I_{\{i\}}\varphi$$

Demostración. Sea $\mathcal{I}$ un sistema interpretado y sea $(r, t) \in I$. Sea φ una fórmula que es estable en $\mathcal{I}$. Para el sólo si, supongamos que $(\mathcal{I}, r, t) \models K_i\varphi$. Sea $(r', t') \in \mathcal{I}$ que cumple $(\exists t, \leq t')$ tal que $r'_i(t_i) = r_i(t)$. Por la definición de $I_{\{i\}}\varphi$, basta ver que $(\mathcal{I}, r', t') \models \varphi$. Como supusimos que $(\mathcal{I}, r, t) \models K_i\varphi$ y como $r'_i(t_i) = r_i(t)$, entonces se cumple que $(\mathcal{I}, r', t_i) \models \varphi$ y como φ es estable en $\mathcal{I}$, para toda $t'' \geq t_i$, se cumple que $(\mathcal{I}, r', t'') \models \varphi$, en particular, como $t' \geq t_i$, se cumple que $(\mathcal{I}, r', t') \models \varphi$. Como (r', t') es cualquiera, por la definición de conocimiento inherente tenemos que $(\mathcal{I}, r, t) \models I_i\varphi$.

Para el regreso, supongamos que $(\mathcal{I}, r, t) \models I_{\{i\}}\varphi$. Sea $(r', t') \in \mathcal{I}$ tal que $r_i(t) = r'_i(t')$. Por definición de concimiento inherente y como $t' \leq t'$ tenemos que $(\mathcal{I}, r', t') \models \varphi$. Como el punto (r', t') es cualquiera podemos concluir que $(\mathcal{I}, r, t) \models K_i\varphi$, como se quería. ∎

El lema 5.1.1 nos dice que si una fórmula es conocimiento inherente de un grupo de procesadores G, entonces, dicha fórmula tiene que ser estable. Como consecuencia, el conocimiento inherente puede ser muy útil en el estudio de dichas fórmulas. La cercana relación que existe entre el conocimiento inherente y estabilidad es más evidente en los sistemas con memoria perfecta, donde el conocimiento inherente es a su vez estable. Formalmente:

Lema 5.1.3. *Sea $\mathcal{I}$ un sistema interpretado con memoria perfecta. Para todo conjunto no vacío de procesadores G se cumple:*

$$\mathcal{I} \models I_G\varphi \Rightarrow \Box I_G\varphi$$

Demostración. Sea $\mathcal{I}$ un sistema interpretado con memoria perfecta y sean $(r, t_0) \in \mathcal{I}$, φ una fórmula y $t \geq t_0$. Supongamos que $(\mathcal{I}, r, t_0) \models I_G\varphi$. Por la definición del operador $\Box$ basta ver que $(\mathcal{I}, r, t) \models I_G\varphi$. Sea $(r', t') \in \mathcal{I}$ que satisface que $(\forall j \in G)(\exists t_j \leq t)$ tal que $r'_j(t_j) = r_j(t)$. Por la definición de conocimiento inherente, basta ver que $(\mathcal{I}, r', t') \models \varphi$.

Como supusimos que el sistema tiene memoria perfecta, entonces, para cada $j \in G$, existe una biyección $f_j : [real\_t_{r'}(\mathsf{start}_j), t_j] \to [real\_t_r(\mathsf{start}_j), t]$ tal que para toda $x \in [real\_t_{r'}(\mathsf{start}_j), t_j]$, se cumple que $r'_j(x) = r_j(f_j(x))$. Como $t_0 \leq t$ y cada f_j es biyectiva, para cada j, existe una $x_j \in [real\_t_{r'}(\mathsf{start}_j), t_j]$ tal que $f_j(x_j) = t_0$, es decir, cada x_j satisface que $r'_j(x_j) = r_j(f(x_j))$. Por la definición de conocimiento inherente y como supusimos que $(\mathcal{I}, r, t_0) \models I_G\varphi$, tenemos que $(r', t') \models \varphi$, como se quería. ∎

El lema 5.1.3 se cumple para fórmulas φ arbitrarias. Como se sabe, si φ es una fórmula estable en un sistema $\mathcal{I}$ con memoria perfecta, entonces, ambas, $K_i\varphi$ y $C_G\varphi$ (véase [FHMV95]), son estables en $\mathcal{I}$. El conocimiento distribuido puede fallar en cumplir ésto, en sistemas que no son síncronos[1], aun para sistemas con memoria perfecta y fórmulas φ estables en dicho sistema. El lema 5.1.3 muestra que esta "deficiencia" del conocimiento distribuido es superada por el conocimiento inherente.

[1]Un sistema $\mathcal{I}$ se dice síncrono si para todo par de puntos $(r, t), (r', t') \in \mathcal{I}$, se cumple que si $r(t) = r'(t')$, entonces $t = t'$. Intuitivamente, en los sistemas síncronos, los procesadores conocen el tiempo real del sistema.

Ejemplo 5.1.
En el presente ejemplo construimos un contraejemplo a la proposición $\models D_G\varphi \Rightarrow \Box D_G\varphi$. El ejemplo es un poco rebuscado, sin embargo ayuda a ver la incertidumbre temporal de los sistemas parcialmente síncronos, aun cuando existen condiciones que se pueden considerar como favorables.

Pensemos en un sistema p.s. $\mathcal{R}$ de dos procesadores llamados s, v. Supongamos que el procesador s tiene un reloj libre de tendencia y el procesador v tiene un $(\underline{\varrho}, \overline{\varrho})$-reloj con $\overline{\varrho} < 1 < \underline{\varrho}$ Sea $r \in \mathcal{R}$ una ejecución y supongamos también que:

1. Para toda $r' \in \mathcal{R}$ si $t_{\text{start}}(s, r') = t_{\text{start}}(v, r') = t'_0$ entonces $\tau(s, (r', t)) = \tau(v, (r', t))$ para toda $t > t'_0$.

2. $t_{\text{start}}(s, r) = t_{\text{start}}(v, r) = t_0$.

Supongamos ademas que la fórmula φ es "los procesadores s y v *siempre* han marcado la misma lectura". Por la condición 1, la fórmula φ es estable en $\mathcal{R}$, es decir, $\mathcal{R} \models \varphi \Rightarrow \Box\varphi$. Es fácil ver que se cumple también $(r, t_0) \models D_{\{s,v\}}\varphi$, ya que *en todas la ejecuciones en que* los procesadores s y v despiertan en el mismo tiempo real se cumple φ.

Como supusimos que el reloj de v no es libre de tendencia y como no se hace ninguna suposición adicional, para toda $t > t_0$ podemos construir un punto $(r', t') \in \mathcal{R}$ que satisface:

1. $r'_s(t') = r_s(t)$

2. $r'_v(t') = r_v(t)$

3. $t_{\text{start}}(s, r') \neq t_{\text{start}}(v, r')$.

Es decir, en r' en el tiempo real t' los relojes de s y v estan sinconizados, sin embargo no lo han estado *siempre*; de donde $(r', t') \not\models \varphi$. Por la *definición de conocimiento* distribuido obtenemos que $(r', t') \not\models D_{\{s,v\}}\varphi$, de donde podemos concluir que:

$$\not\models D_G\psi \Rightarrow \Box D_G\psi$$

Nótese que a pesar de que en r los procesadores mantienen sus relojes sincronizados *siempre*, no se enteran *nunca*. ∎

Veamos por último como se puede caracterizar el conocimiento inherente por medio del concepto de vistas, para el caso de los sistemas parcialmente síncronos. Sea i un procesador;

diremos que un punto (r', t') i-extiende a un punto (r, t) si $\mathcal{V}_i(r, t) \subseteq \mathcal{V}_i(r', t')$. Similarmente, para un conjunto G de procesadores, diremos que (r', t') G-extiende a (r, t) si $\cup_{i \in G} \mathcal{V}_i(r, t) \subseteq \cup_{i \in G} \mathcal{V}_i(r', t')$. Bajo esta misma idea, es fácil ver que la definición de conocimiento inherente se reduce a:

$$(\mathcal{I}, r, t) \models I_G(\varphi) \quad \text{syss} \quad (\mathcal{I}, r', t') \models \varphi \text{ para todo punto } (r', t') \in \mathcal{I} \text{ tal que } (r', t') \text{ } G\text{-extiende a } (r, t)$$

Nótese que en el caso de que G es el conjunto de todos los procesadores del sistema, si un punto (r', t') G-extiende a (r, t) entonces $\mathcal{V}(r, t) \subseteq \mathcal{V}(r', t')$.

5.2 Conocimiento y Tiempo

En esta sección relacionaremos los conceptos de tiempo y conocimiento en nuestro sistema. Para este propósito necesitamos ampliar el lenguaje de modo que existan fórmulas que nos permitan expresar el paso de tiempo en términos cuantitativos así como la posible estrechez en términos del tiempo real que puede separar a dos eventos relacionados por medio de los enunciados de precedencia sincronizada.

Definición 5.2.1. *Sea $\mathcal{I} = (\mathcal{R}, \Sigma, \pi)$ un sistema interpretado y garantizado. Diremos que la interpretación de $e \overset{\alpha}{\leq} e'$ con $\alpha \in \mathbb{R}$, es estándar en $\mathcal{I}$ si $\pi((r, t), (e \overset{\alpha}{\leq} e')) = true$ si y sólo si $e, e' \in (r, t)$ y $real\_t_r(e) + \alpha \leq real\_t_r(e')$.*

Por el resto del presente trabajo, asumimos que las interpretaciones usadas en el resto de este capítulo interpretan de manera estándar a las fórmulas $e \overset{\alpha}{\leq} e'$. Las fórmulas $e \overset{\alpha}{\leq} e'$ nos permiten acotar por la diferencia en términos de tiempo real que existe entre dos posibles eventos. Nótese que similarmente a la definición en los enunciados de precedencia sincronizada, podemos tomar una $\alpha \leq 0$ en la definición de la fórmula $e \overset{\alpha}{\leq} e'$, para acotar superiormente la diferencia en tiempo real de dos eventos.

Una consecuencia inmediata en esta definición y de la definición de los enunciados de precedencia sincronizada, es que las fórmulas generadas con la información de los enunciados de precedencia sincronizadas son verdaderas. Formalmente:

Proposición 5.2.1. *Sea $\mathcal{I} = (\mathcal{R}, \Sigma, \pi)$ un sistema interpretado y garantizado. Para todo punto (r,t) de $\mathcal{I}$ y para todo par de eventos $e, e' \in (r,t)$ si $\mathcal{V}(r,t) \vdash_\Sigma e \overset{\alpha}{\backsim} e'$ entonces $(\mathcal{I}, r, t) \models e \overset{\alpha}{\leq} e'$*

Demostración. Directa del lema 3.2.1. ∎

Este tipo de fórmulas, son los candidatos idóneos para el análisis de problemas relacionados con sincronización de relojes físicos. Nótese que las fórmulas $e \overset{\alpha}{\leq} e'$ son estables, como la siguiente proposición afirma:

Proposición 5.2.2. *Sea $\mathcal{I} = (\mathcal{R}, \Sigma, \pi)$ un sistema interpretado y garantizado. Para todo punto (r,t) de $\mathcal{I}$ y para todo par de eventos $e, e' \in (r,t)$ si $(\mathcal{I}, r, t) \models e \overset{\alpha}{\leq} e'$ entonces $(\mathcal{I}, r, t') \models e \overset{\alpha}{\leq} e'$ para toda $t' \geq t$.*

Demostración. Directa de las definiciones. ∎

El siguiente teorema muestra en un sentido preciso que los procesadores de un sistema parcialmente síncrono pueden, a lo más, recuperar la información derivada de los enunciados de precedencia sincronizada. Formalmente:

Teorema 5.2.3. *Sea $\mathcal{I} = (\mathcal{R}, \Sigma, \pi)$ un sistema interpretado y garantizado. Sea G el conjunto de todos los procesadores del sistema. Para todo punto (r,t) de $\mathcal{R}$ y para todo par de eventos $e, e' \in (r,t)$ se cumple:*

(a) Si $\mathcal{V}(r,t) \vdash_\Sigma e \overset{\alpha}{\rightarrow} e'$ entonces $(\mathcal{I}, r, t) \models I_G(e \overset{\alpha}{\leq} e')$.

(b) Si $(\mathcal{I}, r, t) \models I_G(e \overset{\alpha'}{\leq} e')$ entonces $\mathcal{V}(r,t) \vdash_\Sigma e \overset{\alpha}{\rightarrow} e'$ con $\alpha' \leq \alpha$.

Demostración Probaremos primero la parte (a). Supongamos que $\mathcal{V}(r,t) \vdash_\Sigma e \overset{\alpha}{\rightarrow} e'$ y sea $(r', t') \in \mathcal{I}$ tal que $(\forall j \in G)(\exists t_j \leq t')$ con $r'_j(t_j) = r_j(t)$. Claramente (r', t') satisface que $\mathcal{V}(r,t) \subseteq \mathcal{V}(r', t')$, de donde obtenemos que $\mathcal{V}(r', t') \vdash_\Sigma e \overset{\alpha}{\backsim} e'$ y por la proposición 5.2.1, obtenemos que $(\mathcal{I}, r', t') \models e \overset{\alpha}{\leq} e'$, como se quería.

Para la parte (b), supongamos que $(\mathcal{I}, r, t) \models I_G(e \overset{\alpha'}{\leq} e')$. Si $\alpha' = -\infty$ entonces cualquier α sirve. Supongamos entonces que $\alpha' \neq -\infty$; veamos primero que la distancia entre entre e y e' en la gráfica de sincronización tiene que ser finita. Supongamos por contradicción que no y sea $N \in \mathbb{R}$, tal que $N < \alpha'$. Por el teorema 3.2.4, existe una ejecución r' tal que r' es una ejecución N extrema hacia e'. El mismo teorema nos asegura que existe un t' tal

que $\mathcal{V}(r,t) \subseteq \mathcal{V}(r',t')$, de donde como supusimos $(\mathcal{I},r,t) \models I_G(e \overset{\alpha'}{\leq} e')$ y por la definción de conocimiento inherente, se sigue que:

$$(\mathcal{I},r',t') \models e \overset{\alpha'}{\leq} e' \tag{5.1}$$

Por suposición $d_{\Gamma(r,t)}(e,e') = -\infty$ y por definición de N ejecución extrema hacia e' se sigue que $real\_t_{r'}(e) + N > real\_t_{r'}(e')$ y como $N < \alpha'$ obtenemos que:

$$real\_t_{r'}(e) + \alpha > real\_t_{r'}(e') \tag{5.2}$$

lo cual es una contradicción con (5.1); por lo tanto la $\mathcal{V}(r,t) \vdash_\Sigma e \xrightarrow{\alpha} e'$, para alguna $\alpha \neq -\infty$. Usando otra vez el teorema 3.2.4, existe un punto $(r',t') \in \mathcal{R}$ con $\mathcal{V}(r',t') = \mathcal{V}(r,t)$ y $real\_t_{r'}(e) + \alpha = real\_t_{r'}(e')$. Por la definición del operador de conocimiento inherente también se cumple que $(\mathcal{I},r',t') \models e \overset{\alpha'}{\leq} e'$ lo cual implica que $\alpha \geq \alpha'$, como se quería. ∎

El teorema 5.2.3 nos muestra que la mejor información acerca de la sincronía relativa de los eventos del sistema que los procesadores podrían recuperar es la provista por los enunciados de precedencia sincronizada. Este mismo teorema muestra que dicha infromación es óptima con respecto al conocimiento inherente de los procesadores. Intuitivamente, esto implica que aún si los procesadores intercambiaran y juntaran la historia local de cada uno, no podría deducir ninguna información adicional a la derivada de los enunciados de precedencia sincronizada. Nótese que la misma demostración se aplica a las vistas locales. Un resultado equivalente se utiliza en [MB94] para estudiar el problema de sincronización interna fuera de línea; en particular se usa para analizar desde el punto de vista del conocimiento el trabajo presentado en [HMM85], donde la sincronización se presenta como un problema de programación lineal.

El siguiente teorema define cuándo es posible para cada procesador recuperar información acerca de la sinronía relativa de los eventos.

Teorema 5.2.4. *Sea $\mathcal{I} = (\mathcal{R},\Sigma,\pi)$ un sistema interpretado y garantizado. Para todo punto (r,t) de $\mathcal{R}$, para todo par de eventos $e,e' \in (r,t)$ y para todo procesador i con $e' \in r_i(t)$ se cumple:*

(a) Si $\mathcal{V}_i(r,t) \vdash_\Sigma e \xrightarrow{\alpha} e'$ entonces $(\mathcal{I},r,t) \models K_i(e \overset{\alpha}{\leq} e')$.

(b) Si $(\mathcal{I},r,t) \models K_i(e_j \overset{\alpha'}{\leq} e_i)$, entonces $\mathcal{V}_i(r,t) \vdash_\Sigma e \xrightarrow{\alpha} e'$ con $\alpha \geq \alpha'$.

Demostración. La prueba es muy similar a la del teorema 5.2.3. Procedemos primero por la parte (a). Supongamos que $(r,t) \vdash_\Sigma e \xrightarrow{\alpha} e'$. Sea $(r',t') \in \mathcal{R}$ con $r'_i(t') = r_i(t)$. Por el teorema 2.3.1, se cumple que $\mathcal{V}_i(r,t) = \mathcal{V}_i(r',t')$ de donde directamente se cumple que $\mathcal{V}_i(r,t) \vdash_\Sigma e \xrightarrow{\alpha} e'$ y por la proposición 5.2.1 obtenemos que $(\mathcal{I},r',t') \models e \overset{\alpha}{\leq} e'$, como se quería

Para la parte (b), supongamos que $(\mathcal{I},r,t) \models K_i(e \overset{\alpha'}{\leq} e')$. Si $\alpha' = -\infty$ el teorema se sigue trivialmente; en caso contrario, el teorema 3.2.4 nos garantiza que la distancia entre e y e' en la gráfica de sincronización es finita (el argumento es igual al dado en el teorema 5.2.3), de donde $\mathcal{V}_i(r,t) \vdash_\Sigma e \xrightarrow{\alpha} e'$, con $\alpha \neq -\infty$. Por los teoremas 2.3.1 y 3.2.4, existe un punto $(r',t') \in \mathcal{R}$ con $\mathcal{V}_i(r',t') = \mathcal{V}_i(r,t)$ y $real\_t_{r'}(e) + \alpha = real\_t_{r'}(e')$. Por la definición de conocimiento, tenemos que $(\mathcal{I},r',t') \models e \overset{\alpha'}{\leq} e'$ lo cual implica que $\alpha \geq \alpha'$, como se quería. ∎

El teorema 5.2.4 caracteriza en un sentido preciso la información que los procesadores pueden recuperar acerca de la sincronía relativa de los eventos de un sistema distribuido. Nótese que por el teorema 5.2.3 dicho conocimiento es también óptimo con respecto al conjunto de todos los procesadores del sistema y sobre la vista local de i en el punto (r,t). Es fácil ver como el teorema 5.2.4 tiene importantes e inmediatas consecuencias en la caracterización de los problemas relacionados con la sincronización de relojes, en particular con la sincronización externa en línea. En el siguiente capítulo se ahonda en esta dirección.

Capítulo 6

Sincronización Externa

6.1 Definición del Problema

En un sistema de sincronización externa existe un procesador distinguido s llamado el *procesador fuente* que tiene un reloj libre de tendencia. El trabajo de los demás procesadores del sistema consiste en acotar en cada instante la lectura del reloj local del procesador s. Denotamos a los sistemas de sincronización externa por $\mathcal{R}^{es}$ donde $\mathcal{R}$ es un sistema parcialmente síncrono.

En el desarrollo de este capítulo, supondremos que el sistema $\mathcal{R}^{es}$ se encuentra generado por una una garantía de sistema junto con un protocolo específico de sincronización. De manera general, dicho protocolo debe satisfacer que cada procesador $i \neq s$ mantenga dos variables de salida en cada punto, denotadas por ext_L_i, ext_U_i. El requerimiento de *correctud* para cada procesador i es que para todo punto (r,t), las variables de salida $ext\_L_i(r,t)$, $ext\_U_i(r,t)$ satisfagan:

$$\tau\,(s,(r',t')) \in [ext\_L_i(r',t'),\, ext\_U_i(r',t')]$$

Nótese que un protocolo trivial que en cada punto mantenga $ext\_L_i(r,t) = -\infty$ y $ext\_U_i(r,t) = \infty$ se considera correcto. En este sentido, se define la *estrechez* de la sincronización del procesador i en un punto (r,t), denotada por Θ_i como la diferencia entre las variables de salida en dicho punto:

$$\Theta_i(r,t) = ext\_U_i(r,t) - ext\_L_i(r,t)$$

Si $\Theta_i(r,t) \neq \infty$ diremos que la salida del procesador i en el punto (r,t) es *no trivial*. Diremos que un protocolo es correcto, si su salida en algún punto es no trivial y correcta.

La siguiente definición captura la idea de protocolo óptimo para el caso de los sistemas de sincronización externa:

Definición 6.1.1. *Diremos que la salida del protocolo de sincronización de un procesador i en un punto (r,t) es* óptima, *si existen puntos $(r^L,t_L),(r^U,t_U) \in \mathcal{R}^{es}$ tales que:*

1. $r^L(t_L) = r_i(t)$ *y* $\tau\left(s,(r^L,t_L)\right) = ext\_L_i(r,t)$.

2. $r^U(t_U) = r_i(t)$ *y* $\tau\left(s,(r^U,t_U)\right) = ext\_U_i(r,t)$

Antes de continuar con el análisis del problema de sincronización externa, probaremos una propiedad de los procesadores equipados con un reloj libre de tendencia, como es el caso del procesador fuente. Formalmente:

Lema 6.1.1. *Sea $(\mathcal{R}^{es},\Sigma)$ un sistema garantizado de sincronización externa. Para todo punto $(r,t) \in \mathcal{R}^{es}$ y para todo par de eventos $e^T, e^{T'} \in r_s(t)$ se cumple:*

$$real\_t_r\left(e^T\right) - real\_t_r\left(e^{T'}\right) = T - T'$$

Demostración. Sea $(\mathcal{R}^{es},\Sigma)$ un sistema garantizado de sincronización externa. Sea $(r,t) \in \mathcal{R}^{es}$ y sea $e^T, e^{T'} \in r_s(t)$ una pareja de eventos. Por construcción de los enunciados de precedencia sincronizada se sigue:

$$\begin{aligned} \mathcal{V}(r,t) &\vdash_\Sigma e^T \xrightarrow{T-T'} e^{T'} \\ \mathcal{V}(r,t) &\vdash_\Sigma e^{T'} \xrightarrow{T'-T} e^T \end{aligned}$$

y por el teorema 3.2.3, obtenemos que:

$$\begin{aligned} real\_t_r\left(e^T\right) + (T - T') &\leq real\_t_r\left(e^{T'}\right) \\ real\_t_r\left(e^{T'}\right) + (T' - T) &\leq real\_t_r\left(e^T\right) \end{aligned}$$

de donde directamente se sigue que $real\_t_r\left(e^T\right) - real\_t_r\left(e^{T'}\right) = T - T'$, como se quería. ∎

Intuitivamente, el lema 6.1.1 muestra que los procesadores con relojes libre de tendencia no generan incertidumbre interna acerca del tiempo conforme éste pasa. Una consecuencia del lema 6.1.1 es que la lectura de un reloj local libre de tendencia se puede caracterizar como una sencilla función lineal, como la siguiente proposición muestra:

Proposición 6.1.2. *Sea* $(\mathcal{R}^{es}, \Sigma)$ *un sistema garantizado de sincronización externa. Sea* $(r,t) \in \mathcal{R}^{es}$ *un punto con* $\mathsf{start}_s \in r_s(t)$. *Si el procesador* s *tiene un reloj libre de tendencia entonces* $\tau(s,(r,t)) = t - t_{\mathsf{start}}(r,s)$.

Demostración. Directa del lema 6.1.1 y de la definición de ejecución. ∎

6.2 El Caso General

A continuación probamos una caracterización precisa de la posible estrechez en la sincronización en cada punto de la ejecución. Formalmente:

Teorema 6.2.1. *Sea* $\mathcal{I} = (\mathcal{R}^{es}, \Sigma)$ *un sistema garantizado de sincronización externa. Sea* $(r,t) \in \mathcal{R}^{es}$ *un punto y* $e \in r_i(t)$ *un evento, con* $real\_t_r(e) = t$. *Si* $\mathcal{V}(r,t) \vdash_\Sigma \mathsf{start}_s \xrightarrow{\alpha} e$ *y* $\mathcal{V}(r,t) \vdash_\Sigma e \xrightarrow{\beta} \mathsf{start}_s$ *entonces* $\tau(s,(r,t)) \in [\alpha, -\beta]$.

Demostración. Sea $\mathcal{I} = (\mathcal{R}^{es}, \Sigma)$ un sistema garantizado de sincronización externa. Sea $(r,t) \in \mathcal{R}^{es}$ un punto y $e \in r_i(t)$ un evento, con $real\_t_r(e) = t$ y supongamos que $\mathcal{V}(r,t) \vdash_\Sigma \mathsf{start}_s \xrightarrow{\alpha} e$ y $\mathcal{V}(r,t) \vdash_\Sigma e \xrightarrow{\beta} \mathsf{start}_s$. Como r es consistente con Σ, por el teorema 3.2.3 se sigue que:

$$real\_t_r(\mathsf{start}_s) + \alpha \leq real\_t_r(e) = t \tag{6.1}$$

$$t = real\_t_r(e) + \beta \leq real\_t_r(\mathsf{start}_s) \tag{6.2}$$

Como por suposición el procesador s es libre de tendencia, por la proposición 6.1.2 tenemos que $\tau(s,(r,t)) = t - t_{\mathsf{start}}(r,s)$ y como además por definición se cumple que $real\_t_r(\mathsf{start}_s) = t_{\mathsf{start}}(r,s)$, las eq. (6.1) y (6.2), directamente implican que $\alpha \leq \tau(s,(r,t)) \leq -\beta$; como se quería. ∎

El teorema 6.2.1 nos permite inmediatamente derivar, de manera equivalente a [PSR94], la cota inferior en la posible estrechez de cualquier algoritmo distribuido de sincronización externa. Formalmente,

Teorema 6.2.2. *Sea* $\mathcal{I} = (\mathcal{R}^{es}, \Sigma)$ *un sistema garantizado de sincronización externa. Para todo procesador* i *y para todo evento* $e \in r_i(t)$, *si* $real\_t_r(e) = t$ *con* $\mathcal{V}(r,t) \vdash_\Sigma \mathsf{start}_s \overset{\alpha}{\rightsquigarrow} e$ *y* $\mathcal{V}(r,t) \vdash_\Sigma e \overset{\beta}{\rightsquigarrow} \mathsf{start}_s$ *entonces* $\Theta_i(r,t) \geq (-\beta - \alpha)$.

Demostración. Sea $\mathcal{I} = (\mathcal{R}^{es}, \Sigma)$ un sistema garantizado de sincronización externa. Sea i un procesador y supongamos que $e \in r_i(t)$, con $real\_t_r(e) = t$, con $\mathcal{V}(r,t) \vdash_\Sigma \mathsf{start}_s \overset{\alpha}{\rightsquigarrow} e$ y $\mathcal{V}(r,t) \vdash_\Sigma e \overset{\beta}{\rightsquigarrow} \mathsf{start}_s$. La condición de correctud del problema de sincronización externa, implica que $\alpha \leq ext\_L_i(r,t)$ y que $ext\_U_i(r,t) \geq -\beta$, de donde directamente obtenemos que $\Theta_i(r,t) = (ext\_U_i(r,t) - ext\_L_i(r,t)) \geq (-\beta - \alpha)$, como se quería ∎

El teorema 6.2.1 junto con el teorema 5.2.4 nos permite derivar inmediatamente un sencillo protocolo de sincronización externa óptimo: un protocolo de información completa que deduzca los enunciados de precedencia sincronizada de la gráfica de sincronización en dicho punto. El protocolo anteriormente descrito es óptimo por el teorema 3.2.4; desgraciadamente, un protocolo información completa no se puede considerar, desde cualquier punto de vista práctico, una solución adecuada, debido al gasto requerido en espacio para conservar la historia completa de cada uno de los procesadores del sistema. Nótese sin embargo que en [PSR94], se muestra que el alto costo en espacio es inherente a los protocolos de sincronización óptimos y generales.

Bajo la suposición adicional de que todos los procesadores del sistema se encuentran equipados con relojes libres de tendencia, en [PSR94] se deriva un sencillo protocolo de sincronización, óptimo y eficiente desde el punto de vista práctico. Los detalles formales de dicho algoritmo se encuentran fuera de el alcance de éste trabajo, sin embargo, el algoritmo se puede derivar del análisis de un sencillo modelo de sincronización externa, como el que se presenta en la siguiente sección.

6.3 Ejemplo: La Técnica de Viaje Redondo

Para ejemplificar el problema de sincronización externa, usaremos un sencillo modelo de dos procesadores, denotados por s y v, los cuales están equipados con relojes libres de tendencia, y conectados por una línea bidireccional de comunicación perfectamente asíncrona, es decir, cada mensaje puede ser entregado en *al menos* 0 unidades de tiempo y a *lo más* ∞ unidades de tiempo.

El trabajo particular de sincronización que se considera es donde v tiene que acotar, en cada instante y lo mejor posible, la lectura del tiempo local de s. La técnica de *viaje redondo* es la base de todos los algoritmos de sincronización externa. A continuación damos una versión simplificada de dicha técnica: periódicamente, el procesador v manda un mensaje a

s, el cual a su vez le responde enviando un mensaje de regreso a v (de ahí el nombre de viaje redondo). Considérese el viaje redondo de la figura 6.1, donde v envía un mensaje m a s, y s contesta enviando m' a v.

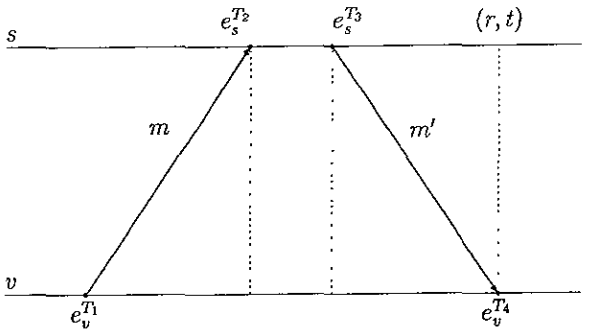


Figura 6.1: Un viaje redondo.

Como el mensaje m permanece en tránsito al menos 0 unidades de tiempo (figura 6.2(a)), se sigue que $\mathcal{V}(r,t) \vdash_\Sigma e_s^{T_3} \overset{0}{\backsim} e_v^{T_4}$. Por otro lado, como m' estuvo en tránsito a lo más $TT = (T_4 - T_1) - (T_3 - T_2)$ unidades de tiempo (figura 6.2(b)), también se cumple que $\mathcal{V}(r,t) \vdash_\Sigma e_v^{T_4} \overset{-TT}{\backsim} e_s^{T_3}$. De donde, por el teorema 5.2.4 se sigue que

$$\begin{aligned}(I,r,t) &\models K_v(e_s^{T_3} \overset{0}{\backsim} e_v^{T_4}) \\ (I,r,t) &\models K_v(e_v^{T_4} \overset{-TT}{\backsim} e_s^{T_3})\end{aligned}$$

Es fácil ver que en este caso y dado su conocimiento, el procesador v puede deducir que $\tau_s(r,t) \in [T_3, T_3 + TT]$. El procesador i puede entonces asignar las variables de salida en $ext\_L_i(r,t) = T_3$, $ext\_U_i(r,t) = T_3 + TT$, cumpliendo los requisitos de correctud y optimalidad arriba descritos. Todos los algoritmos de sincronización externa tienen como base esta técnica. Nótese que en este caso la estrechez de la sincronización es, exactamente, el tiempo total de tránsito T de los mensajes m y m'.

6.3.1 La Técnica de Viajes Múltiples Usada por NTP

En esta sección analizaremos de manera informal la técnica de viajes múltiples usada por NTP [Mil92], el protocolo de sincronización externa usado en INTERNET.

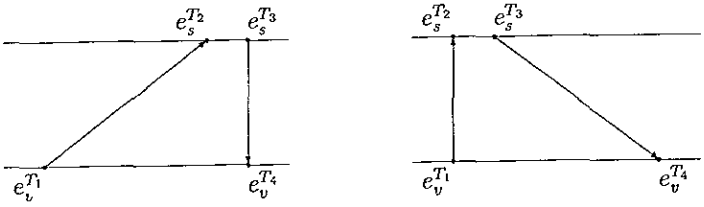
Nótese que en ejemplo de la sección anterior y en el caso de un sólo viaje, la estrechez de la sincronización es exactamente igual al tiempo total de tránsito de los mensajes del viaje redondo; en otras palabras entre más rápido se completa el viaje redondo, mejor es la sincronización adquirida. Esta observación llevó al siguiente diseño de NTP: cuando existen múltiples viajes redondos, el que tiene mínimo tiempo total de tránsito se selecciona como el "mejor", y sus correspondientes cotas son seleccionadas como salida del algoritmo, desechando las cotas anteriores.

Como hemos visto, dado que los problemas de sincronización dependen crucialmente de un uso óptimo del conocimiento de cada procesador, intuitivamente NTP puede estar perdiendo valiosa información cada vez que descarta un viaje, aun cuando este tenga un tiempo total de tránsito menos favorable que otros. A continuación mostraremos por medio de un ejemplo concreto que esta intuición es correcta.

Consideremos la ejecución de la figura 6.3, donde existen dos viajes redondos. Sea $TT_1 = (T_4 - T_1) - (T_3 - T_2)$ y $TT_2 = (T_8 - T_5) - (T_7 - T_6)$ y supongamos que $TT_1 < TT_2$. La suposición que $TT_1 < TT_2$ implica que el tiempo total de tránsito del primer viaje redondo es menor que el tiempo de tránsito del segundo viaje, como la figura 6.3 sugiere.

En este caso, en el punto (r, t), NTP asigna $ext\_L_v(r, t) = T_7$ y $ext\_U_v(r, t) = T_7 + TT_1$, usando las cotas del primer viaje redondo para la sincronización (véase la figura 6.3). Claramente, la elección de cotas que hace NTP es no trivial y correcta; sin embargo, la salida de NTP no es óptima y aún más, la salida de NTP contempla implícitamente escenarios que contradicen la realidad física. En este caso, por ejemplo, NTP asume implícitamente que el escenario de la figura 6.4, donde el mensaje enviado desde s en el evento $e_s^{T_3}$ toma 0 unidades de tiempo real en arribar a v, es posible.

Nótese sin embargo que, como la figura sugiere, la elección del tiempo total de tránsito de los dos viajes redondos es $(T_8 - T_1) - (T_7 - T_2)$, con lo cual se obtiene el resultado óptimo. Dicho de otra forma, la figura sugiere que la elección de $TT^* = (T_8 - T_1) - (T_7 - T_2)$, es mejor que TT_1.



(a) (b)

Figura 6.2: El procesador v considera posibles los siguiente escenarios extremos. En (a), el mensaje m esta en *tránsito* TT unidades de tiempo. En (b), el mensaje m' esta en tránsito TT unidades de tiempo.

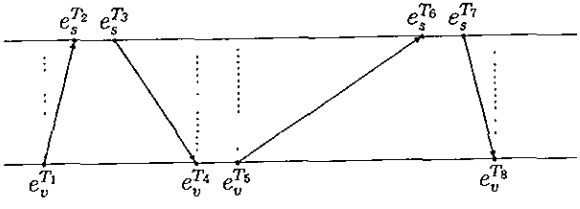


Figura 6.3: Ejecución con dos viajes redondos. Nótese que el tiempo total de tránsito del primer viaje se sugiere menor que el tiempo total de tránsito del segundo viaje.

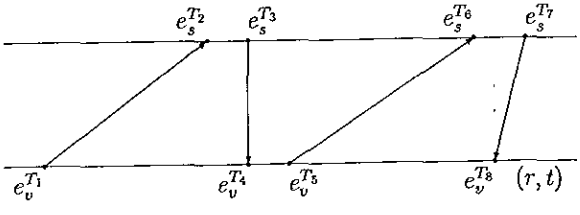


Figura 6.4: Al *desechar* el *segundo* viaje redondo de la figura 6.3, NTP considera posible, de forma implícita, el escenario mostrado en la figura. Nótese que como la figura muestra, este escenario es *imposible*.

Formalmente, en el punto (r,t), el sistema satisface:

$$\mathcal{V}(r,t) \vdash_\Sigma e_s^{T_7} \xrightarrow{0} e_v^{T_8} \tag{6.3}$$

$$\mathcal{V}(r,t) \vdash_\Sigma e_s^{T_8} \xrightarrow{-TT^*} e_v^{T_7} \tag{6.4}$$

La primera cota se toma, como en el caso de un viaje redondo, considerando el caso extremo en el cual el mensaje enviado desde s a v en el evento $e_s^{T_7}$ se entrega en 0 unidades de tiempo real. La segunda garantía se toma cuando se considera el otro caso extremo, donde la entrega del mensaje enviado desde v a s en su evento local $e_v^{T_1}$ se entrega de forma instantánea. Por el teorema 5.2.4, en el punto (r,t) se cumple:

$$(I,r,t) \models K_v(e_s^{T_7} \overset{0}{\leq} e_v^{T_8}) \tag{6.5}$$

$$(I,r,t) \models K_v(e_v^{T_8} \overset{-TT^*}{\leq} e_s^{T_7}) \tag{6.6}$$

Dado su conocimiento en el punto (r,t), el procesador puede deducir que que $\tau(s,(r,t)) \in [T_7, T_7 + TT^*]$. Nótese además que es posible construir ejecuciones, semejantes a la de la figura 6.3, donde TT^* sea arbitrariamente más pequeño que TT_1.

Intuitivamente, y como se hace notar en [PS94], el problema en la técnica de viajes múltiples usada por NTP, es que potencialmente puede aparejar un mensaje "bueno" en una dirección, con uno "malo" en la otra, quedando así imposibilitado a hacer un uso óptimo de su conocimiento. Si el sistema cuenta con más de una línea de comunicación entre los dos procesadores, el aparejamiento que hace NTP de mensajes buenos con malos puede ser aún más severo, debido a múltiples causas, como por ejemplo, la elección de rutas no determinísticas por parte del sistema de entrega de mensajes.

Capítulo 7

Conclusiones

El *presente trabajo presenta* un marco teórico para el análisis de los problema de sincronización desde el punto de vista formal del conocimiento en sistemas distribuidos. La teoría aquí desarrollada muestra una visión alternativa y complementaria al trabajo de Patt–Shamir y Rajsbaum presentado en [PSR94] *y al de Moses y Bloom* presentado en [MB94].

Es interesante notar que los resultados que se desarrollan en el presente trabajo para los operadores de conocimiento descansan plenamente en las propiedades de los sistemas parcialmente síncronos, estudiadas y desarrolladas con las herramientas presentadas en [PSR94] y de manera prácticamente paralela. En este sentido, el análisis aquí presentadado se desarrolla más en términos de combinatoria que en términos de operadores modales y proposiciones lógicas; sin embargo, una profunda comprensión de las propiedades de los sistemas parcialmente síncronos es un requisito inherente a los problemas de sincronización.

En el presente trabajo se opta por adaptar la teoría de [PSR94] al modelo de conocimiento en vez de usar la teoría desarrollada desde este mismo punto de vista en [MB94]; por demás, bastante similar. Considero que el análisis de los sistemas parcialmente síncronos haciendo uso de las gráficas de sincronización se simplifica y clarifica, en mucho por el uso de la Teoría de Gráficas. El uso conjunto de las gráficas de sincronización con el conocimiento apenas comienza y creo que la unión de ambas abstracciones puede ayudar a mejorar la comprensión y el análisis de los problemas de sincronización.

Apéndice A

Diagramas de Tiempo Espacio

Los diagramas de Tiempo Espacio [Lam86], permiten representar de forma gráfica, ejecuciones de sistemas distribuidos. Bajo esta representación, el eje vertical es usado para representar posiciones en el espacio, mientras que el eje horizontal sirve para representar el paso del *tiempo*. Dado que la posición física de los procesadores es, bajo este modelo, inmaterial, los procesadores son representados por medio de lineas horizontales etiquetadas con su nombre. En este trabajo asumimos que el tiempo crece, sobre el eje horizontal, de izquierda a derecha. Véase la figura A.1.

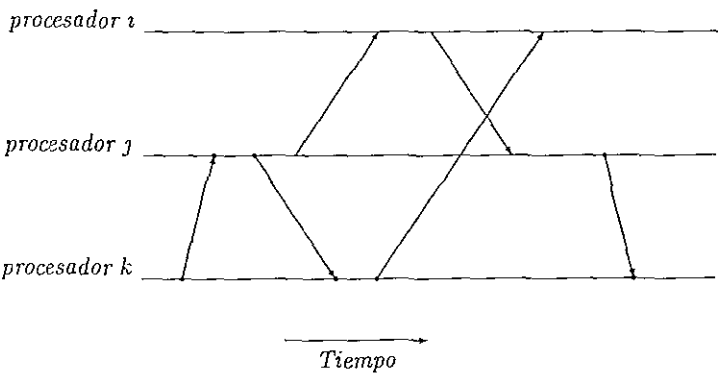


Figura A.1: Ejemplo de un diagrama de Tiempo Espacio.

Los eventos de la ejecución de un sistema distribuido, como son el envío y recepción de mensajes, son representados por puntos, de tal forma que su posición en el diagrama,

queda determinada por el procesador donde sucede el evento, junto con el tiempo de su ocurrencia. Un mensaje entre dos procesadores distintos se representa por medio de una flecha, cuyo origen corresponde al punto que representa envío del mensaje y su extremo la correspondiente recepción.

Bibliografía

[AHR96] Hagit Attiya, Amir Hezberg, and Sergio Rajsbaum. Optimal clock synchronization under different delay assumptions. *SIAM Journal of Computing*, 25(2):369–389, 1996. A preliminary version appered in *Proceedings of the 12th Annual ACM Symposium on Principles of Distributed Computing* (1994).

[AW98] Hagit Attiya and Jeniffer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics.* Mc Graw Hill Publishing Company, 1998.

[Che93] Brian F. Chellas. *Modal logic: an introduction.* Cambridge University Press, 5 edition, 1993.

[FHMV95] Ronal Fagin, Joseph Y. Harplen, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge.* Massachusetts Institute of Technology press, 1995.

[HM90] Joshep Y. Harplen and Yoram Moses. Knowledge and common knowledge in a distributed enviroment. *Journal of the ACM*, 37(3):549–587, 1990. A preliminary version appered in *Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing.*

[HM92] Joshep Y. Harplen and Yoram Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, pages 319–379, 1992.

[HMM85] Joseph Y. Halpern, Nimrod Megiddo, and Ashfaq A. Munshi. Optimal precision on the precence of uncertainty. *Journal of Complexity*, 1(1):170–196, 1985. A preliminary version appered in *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing.*

[Lam78] Leslie Lamport. Time, clocks and the ordering of events in a distributed system. *Communications of the ACM*, 7(21):558–565, 1978.

[Lam86] Leslie Lamport. The mutual exclusion problem. part i: A theory of interprocess communication. *Journal of the ACM*, 2(33):313–326, 1986.

[Lis93] Barbara Liskov. Practical uses of syncronized clocks in distributed systems. *Distributed Computing*, 6:211–219, 1993. Invited talk at the *9th Annual ACM Symposium on Principles of Distributed Computing.*

[Lyn97] Nancy A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, Inc, 1997.

[MB94] Yoram Moses and Ben Bloom. Knowledge, timed precedence and clocks (preliminary report). In *Proceedings of the 13th ACM Symposioum on Principles of Distributed Compiuting.*, pages 294–303, August 1994.

[Mil92] David L. Mills. Network time protocol (version 3): Specification, implementation and analysis. RFC 1305, Network Working Group, University of Delaware, March 1992.

[PS94] Boaz Patt-Shamir. *A Theory of Clock Synchronization.* PhD thesis, Massachusetts Institute of Technology, October 1994.

[PSR94] Boaz Patt-Shamir and Sergio Rajsbaum. A theory of clock sinchronization. In *Proceedings of the 26th Symposium on Theory of Computation*, pages 810–819, May 1994.

[SWL88] Barbara Simons, Jeniffer Lundelius Welch, and Nancy Lynch. An overview of clock sinchronization. Research Report RJ 6505 (63306), IMB, October 1988.

[WL84] Jeniffer Lundelius Welch and Nancy Lynch. An upper and lower bound for clock sincronization. *Information and Control*, 62(2-3):190–204, August/September 1984.