Estudio comparado de lenguajes gráficos de especificación de sistemas

Sandra N. Martinez
Licenciada en Sistemas de Información

Director: Dr. Alfredo Olivero
Co-director: Dr. Gustavo Rossi

Tesis presentada para obtener el grado de Magíster en Ingeniería de Software.
Facultad de Informática - Universidad de La Plata
Diciembre 2011

Dedicatoria

A Nicolás, Rodrigo y Mariano,
son mi inspiración para seguir
siempre adelante.

Agradecimientos

A Axel y mis hijos, por su apoyo incondicional.

A mis padres y hermana por su confianza y aliento constante para llevar adelante este trabajo.

A Alfredo, mi tutor, por su tiempo y paciencia infinita.

CONTENIDO:

FIGURAS:

Capítulo 1: Introducción y Objetivos

1.1 Introducción

El proceso de desarrollo de software o nuevas tecnologías (Internet, telecomunicaciones, telefonía celular, satélites, etc.) comienza con el arduo trabajo de especificar los requerimientos. Seguidamente se diseña un sistema que satisfaga estos requerimientos pero esto no es suficiente para asegurar que se ha llegado al cumplimiento de los objetivos, se debe validar el software desarrollado contra las propiedades que el mismo debe cumplir.

Una de las formas de asegurar que el software o sistema obtenido funcione según esas propiedades especificadas antes de su codificación es utilizar modelos del sistema que se propone como solución para satisfacer estos requerimientos. Esto permite chequear si el software en cuestión responde o no a dichas propiedades o requerimientos y tomar decisiones a bajo costo.

Por otro lado, surgen técnicas basadas en métodos formales y herramientas que permiten expresar estas propiedades y verificar luego el cumplimiento o no de las mismas por parte del software desarrollado. Pero estos métodos formales suelen ser difíciles de transferir a la industria de desarrollo de software. Una de las razones es la gran brecha entre estos formalismos y los utilizados en estas industrias para documentar y describir los sistemas. Un ejemplo es la notación Z [S98] que ha tenido un éxito considerable entre los usuarios de métodos formales pero no así entre los desarrolladores. O sea, que la especificación de requerimientos del software sigue siendo hecha, en su mayoría, de manera informal y por eso la forma más común para validar la confiabilidad de un software es mediante el testeo.

Una parte de la solución a esta situación es el uso de formalismos visuales para expresar los requerimientos. Son mucho más intuitivos que los formalismos textuales ya que con una imagen se puede describir estas propiedades con mayor facilidad que con palabras. Un ejemplo de esto es el lenguaje gráfico TE (TimeEdit) [SHE01] y UML (Unified Modeling Language) [RSC97].

Hay otros formalismos que tienen ambos aspectos, el formal y el gráfico, como por ejemplo MSC (Message Sequence Charts) [ITU04] y VTS (Visual Timed event Scenarios) [BKO05]. Estos no sólo ofrecen un entorno sencillo para especificar los requerimientos de manera gráfica sino que también cuentan con una base formal que permite la verificación automática del cumplimiento o no de las propiedades expresadas.

Las herramientas utilizadas en la industria para definir requisitos suelen mostrar en forma gráfica lo que se espera que suceda con el sistema. Su finalidad principal es describir el sistema permitiendo una mejor comunicación entre los diseñadores y quienes harán la implementación del mismo, como así también una forma de documentar el sistema desarrollado. En la mayoría de los casos carecen de base formal que permita aplicar técnicas automáticas de verificación.

Teniendo en cuenta que los lenguajes utilizados por la industria tienen gran difusión y que, por otro lado, los empleados a nivel académico y científico suelen poseer la propiedad de ser verificables automáticamente, surge la necesidad entonces de vincular estos lenguajes y obtener beneficios de estos dos aspectos.

Además, si bien se pueden encontrar algunos trabajos que relacionan herramientas que se utilizan en el contexto industrial, los mismos son parciales y no involucran, a nuestro conocimiento, una comparación con los lenguajes formales desarrollados en las últimas décadas.

Al relacionar los métodos utilizados en la industria de desarrollo de software y las utilizadas en el ámbito académico se podrá continuar con el uso de herramientas conocidas para definir requerimientos de una manera intuitiva pero obteniendo modelos verificables en forma automática.

1.2 Objetivo

El objetivo de esta tesis es estudiar, analizar, comparar y vincular distintos formalismos o herramientas de modelado utilizados tanto en la industria como en el ámbito científico para expresar requerimientos y/o propiedades de los modelos de sistemas informáticos. De esta comparación surgen las fortalezas y las debilidades de cada uno de ellos.
En los casos que sean factibles se define la traducción total o parcial de un formalismo a otro. Esto posibilitará la concepción de traductores automáticos sacando ventaja del uso generalizado de los lenguajes empleados por la industria con la posibilidad que cuentan los lenguajes formales de validar las propiedades automáticamente.

1.3 Alcance

Podemos encontrar distintos tipos de lenguajes o herramientas para verificar requerimientos. Están los netamente gráficos, los que usan sólo texto, los que tienen su definición formal o los que combinan algunos de estos aspectos, como por ejemplo UML.
Para realizar este trabajo se eligieron tres herramientas que cubren estos aspectos de diferentes formas y que hacen que su uso sea muy dispar. Los formalismos seleccionados son: TimeEdit (TE), Message Sequence Charts (MSC) y Visual Timed event Scenarios (VTS).
Se escogió TE por su sencillez y por ser puramente gráfico. Debido a que fue desarrollado para un problema específico (como se verá en el siguiente capítulo), es una herramienta que no ha tenido mayor difusión.
A diferencia de TE, se incorporó MSC a este trabajo por ser una de las herramientas más utilizadas en el ámbito industrial del software relacionado a telecomunicaciones. Cuenta con varios graficadores disponibles y también con una definición textual en SDL (System Design Languages), aunque carece de una formalización total.
VTS se estudió por ser un lenguaje que tiene tanto un aspecto gráfico como una definición formal completa. Permite expresar situaciones no triviales como ordenes parciales entre eventos, el primero o el último evento de un conjunto, etc. También cuenta con un editor de gráficos para VISIO que permite generar un archivo XML que representa al escenario.

1.4 Contribuciones

Para cumplir con los objetivos de este trabajo se realizaron las siguientes contribuciones:

- ✓ Se propuso una formalización de la herramienta TE, ya que la misma carecía de ella. Esto permitió una comparación más completa y no restringida sólo a la parte gráfica.
- ✓ Se desarrolló un generador de corridas para el formalismo VTS, que toma un escenario descripto en un archivo de texto como entrada y genera un archivo de salida de corridas que cumplen con las propiedades del escenario sin considerar los aspectos temporales. Las corridas obtenidas se muestran como expresiones regulares donde las E (o ∑) representan secuencias de eventos potencialmente infinitas.
- ✓ Se establecieron reglas para traducir tanto las representaciones formales como gráficas entre TE y VTS.
- ✓ Se establecieron reglas para traducir las representaciones formales, gráficas, y textuales de MSC a VTS.
- ✓ Se compararon los poderes expresivos de TE, MSC y VTS.

1.5 Organización del texto

El texto de este documento está organizado de la siguiente manera:
El Capítulo 1 se presenta este trabajo a través de una introducción y las contribuciones obtenidas con el mismo. Luego en el Capítulo 2 se presentan los lenguajes de especificación estudiados, mientras que en el Capítulo 3 se profundiza sobre los beneficios de la verificación de requerimientos y se presenta el generador de corridas desarrollado.
A continuación en el Capítulo 4 se muestran las traducciones entre TE y VTS y en el Capítulo 5 las traducciones de MSC a VTS.
La comparación entre TE, MSC y VTS se desarrolla en el Capítulo 6.
Finalmente, en el Capítulo 7 se presentan las conclusiones alcanzadas y los trabajos a futuro.
También se adjunta un listado de siglas utilizadas en este trabajo y un listado de referencias bibliográficas.

Capítulo 2: Lenguajes de especificación de requerimientos

2.1 La necesidad de especificar requerimientos

Cuando se inicia el proceso de desarrollo de software es imprescindible poder contar con definiciones correctas de los requerimientos o propiedades que debería cumplir el software en cuestión. Para esto se cuenta con distintas técnicas que permiten listar estas propiedades sin ambigüedad evitando distintas interpretaciones de las mismas.
Actualmente las llamadas “nuevas tecnologías” involucran gran volumen de desarrollo de software. Hoy la telefonía celular, Internet y las telecomunicaciones ofrecen más servicios y nuevas funcionalidades haciendo cada vez más desafiantes los requerimientos o propiedades que debe cumplir el software que utilizan. A medida que estos requerimientos aumentan se hace más compleja la forma de corroborar que el software desarrollado cumpla estos requerimientos o propiedades especificadas [IEEE98].

Utilizando distintas herramientas en la etapa de análisis, como el DFD (Diagrama de Flujo de datos) que modela qué funciones debe cumplir el sistema para satisfacer al usuario o como el DER (Diagrama de Entidad-Relación) que modela los datos almacenados y sus relaciones, se pueden generar modelos del sistema antes de su codificación. Estas herramientas son muy útiles para la comunicación entre personas del proyecto (desarrolladores y programadores) y entre los desarrolladores y el usuario para asegurarse que lo que se está haciendo cumple con sus necesidades y expectativas. Sin embargo, estas metodologías carecen de la capacidad de verificar si los sistemas que representan aseguran el cumplimiento de los requisitos o propiedades exigidas.

Por otro lado tenemos los lenguajes de especificación de requerimientos (siendo de interés para este trabajo aquellos requerimientos asociados a eventos temporales) que utilizando notación gráfica o simbólica simplifican la formalización de estas propiedades y permiten modelar distintas trazas (secuencias de eventos que pueden observarse en una ejecución) o escenarios (conjuntos de trazas) que cumplen con las mismas.

Es decir, que por un lado se obtienen los modelos del sistema y por el otro los requerimientos para ser chequeados [P04].

En nuestros días contamos con herramientas que permiten analizar los errores en distintos tipos de software (incluyendo distribuidos o concurrentes) pero hay ciertos problemas de lógica de control o comunicaciones que son difíciles de testear aún valiéndose de ellas.

El esquema que se muestra en la Figura 1 permite observar un modelo del proceso de verificación. Los modelos nombrados anteriormente (DFD y DER) no son adecuados para modelar sistemas con requerimientos vinculados a eventos. Para modelar estos sistema se podría utilizar autómatas con tiempos y la definición de las propiedades podría realizarse en VTS que cuenta con un traductor a dichos autómatas. Como verificador se podría utilizar Kronos [BDMOTY98] que permite especificar sistemas con autómatas con tiempos, grafos

temporizados y requerimientos con TCTL (Timed Computation Tree Logic) obteniendo como resultado un diagnóstico de por qué una propiedad se cumple o no.

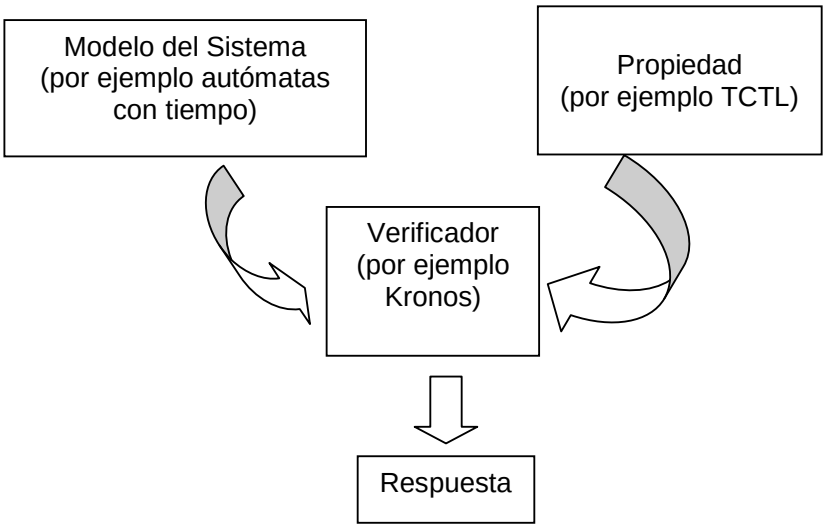
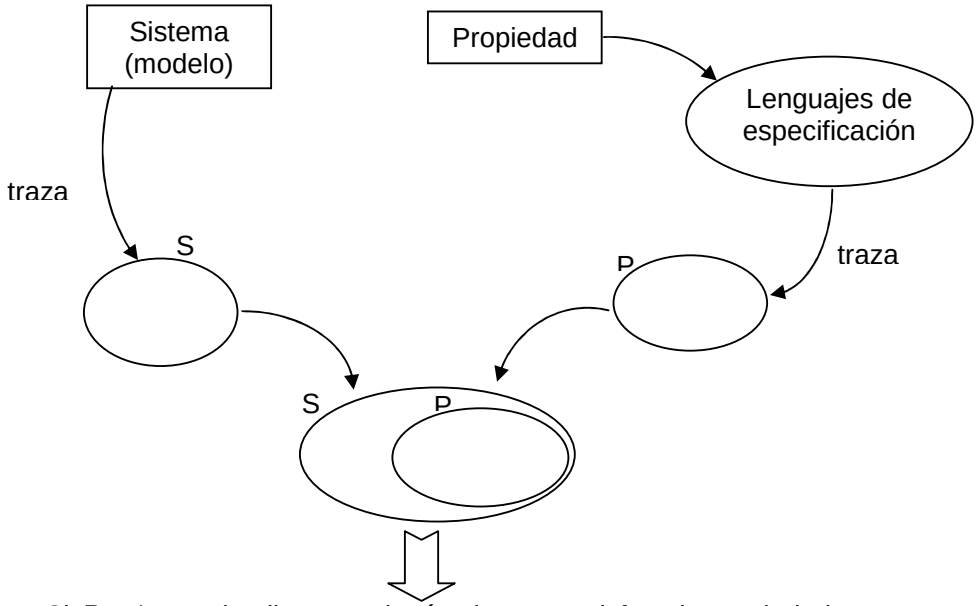


Figura 1: Proceso de verificación

Si tenemos en cuenta las trazas que se obtienen de las corridas del modelo y las que surgen como resultado de las propiedades especificadas por los lenguajes observaríamos el esquema que se presenta en la Figura 2.



Si $P = \phi$ esto implica que ningún sistema satisface la propiedad.

Si $P \subset S$ pero $P \neq S$ entonces existen trazas que satisfacen la propiedad y otras no.

Si $P = S$ implica que el sistema satisface la propiedad de manera general.

Figura 2: Perspectiva desde las trazas

Lo que obtenemos son un conjunto de trazas del sistema (S) y un conjunto de trazas de propiedades (P). Si existe una traza del modelo del sistema que satisface las propiedades puede darse una de las siguientes situaciones:

- Si las propiedades son “positivas” la satisfacción implica “OK” (liveness).
- Si las propiedades son “negativas” la satisfacción implica “ERROR”.

Es importante aclarar que tanto S como P pueden ser conjuntos infinitos, es decir, que la cantidad de trazas que se pueden obtener tanto del modelo como de los requerimientos especificados pueden resultar infinitas. No es parte de este trabajo analizar las maneras posibles de validación de los conjuntos de trazas aún siendo expresadas de manera finita (con expresiones regulares) para determinar si se satisfacen o no las propiedades especificadas.

2.2 Lenguajes de especificación de requerimientos basados en eventos

Si bien existen distintos tipos de requerimientos y distintos lenguajes de especificación, para este trabajo se tuvieron en cuenta aquellos lenguajes basados en eventos o los que trabajan con envío y recepción de mensajes tomando estas acciones como eventos distintos. O sea, se trabajó con aquellos lenguajes que nos permiten especificar los eventos que ocurren en ciertos momentos, al mismo tiempo o en momentos inesperados. Se denomina *evento* a la ocurrencia de algo que causa un estímulo al sistema. Como respuesta a este estímulo se puede disparar una transición a otro estado o la obligación de permanencia en el estado en que se está.

Al trabajar con mensajes se consideraron eventos distintos al envío de un mensaje y a su recepción puesto que ocurre un cambio de estado con relación al mismo mensaje.

Las trazas obtenidas tanto por el modelo del sistema como por el lenguaje de especificación serán una secuencia de eventos teniendo en cuenta el paso del tiempo.

2.3 TimeEdit (TE)

2.3.1 ¿Cómo surge?

Esta herramienta surge en los laboratorios Bell como solución a un problema de especificación y verificación de más de 110 requerimientos distintos de un complejo sistema de telecomunicaciones. Teniendo bien en claro los beneficios de utilizar un lenguaje formal para especificar los requerimientos eligieron primeramente LTL (Linear Temporal Logic) que permite describir cómo los eventos y estados del sistema evolucionan en el tiempo [SHE01]. Pero surge el inconveniente de lo difícil y arduo que es expresar requisitos usando las fórmulas de LTL, aún para los expertos. Tenían entonces dos alternativas. Una era capturar todos los requerimientos mediante fórmulas de LTL, con el consecuente tiempo que demandaría expresarlos. La segunda opción era encontrar un modo natural para generar los requerimientos formales sin escribir las fórmulas de LTL. Se optó por la segunda opción surgiendo así este editor gráfico que describe requerimientos a partir de una representación visual.
Con esta herramienta pudieron especificar los requerimientos de una manera más intuitiva con líneas de tiempo que mostraban 4 o 5 eventos y 2 o 3 restricciones en cada gráfico.
Para la verificación de estos requerimientos se tradujeron los escenarios de TE a Büchi Automata (funcionalidad del TimeLine Editor) para luego utilizar Spin, una herramienta de

verificación también desarrollada en los laboratorios de Bell. (Sitio oficial: http://spinroot.com/spin/whatispin.html#A)

2.3.2 Aspecto gráfico

La notación gráfica del TE utiliza los siguientes elementos:

✓ **Línea temporal**

La línea temporal está representada por una línea horizontal donde el tiempo avanza de izquierda a derecha.

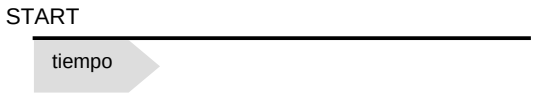


Figura 3: Componente de un gráfico TE: Línea temporal

✓ **Marcas**

Descendiendo desde esta línea temporal se encuentran barras verticales denominadas *marcas* que representan instantes en el tiempo donde ocurrió un evento de interés. No se asumen intervalos fijos de tiempo entre las marcas. Los eventos son asociados a estas marcas y pueden ser de tipo **e**, **r**, ó **f**. Los eventos de tipo **e** son eventos esperados, opcionales y sirven para corroborar la precisa ejecución del sistema que estamos modelando. Los eventos de tipo **r** o requeridos son aquellos que deben suceder si ocurrieron todos los eventos anteriores definidos en la línea de tiempo. Si estos eventos están ausentes se produce un error. Por último están los eventos de tipo **f** o de falla que se indican en el editor con una cruz roja en la marca y son aquellos que no deben ocurrir si se han observado todos los eventos previos de la línea de tiempo. La ocurrencia de un evento f indica un error.

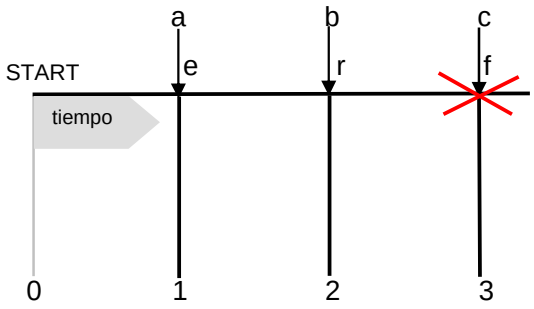


Figura 4: Componente de un gráfico TE: Marcas

En la Figura 4 se observa que en la marca 1 se espera encontrar un evento ***a***, en la marca 2 *debe* ocurrir un evento ***b*** (si esto no sucede se produce un error) y si luego ocurre un evento ***c*** se produce un error.

✓ **Restricciones**

Por debajo de la línea de tiempo se observan las *restricciones*, indicadas con líneas horizontales, que van de una marca a otra y que especifican los eventos que **no** se desea que ocurran en ese lapso. Estas restricciones pueden incluir o no las marcas de inicio y final.

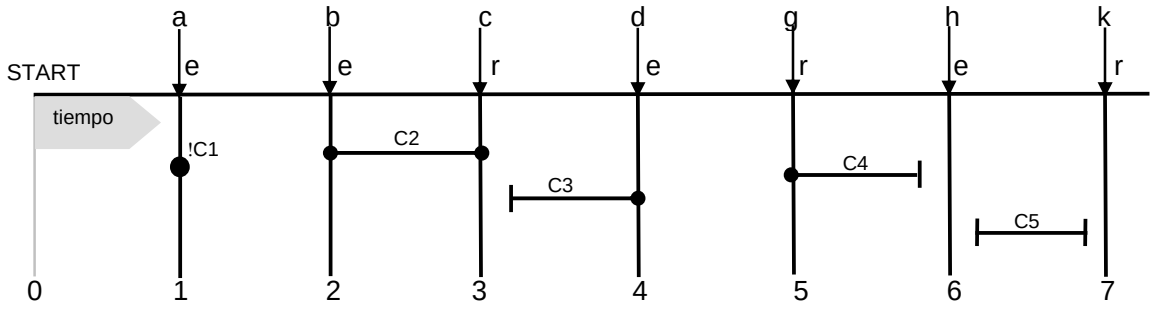


Figura 5: Componente de un gráfico TE: Restricciones

En la Figura 5, en la marca 1 se produce un evento ***a*** pero no se desea que ocurra un evento ***c1***. Luego se produce un evento ***b*** en la marca 2 y más tarde debe ocurrir un evento ***c*** en la marca 3 pero no se desea la ocurrencia del evento ***c2*** cuando se producen los eventos de las marcas 2 y 3 ni entre estas dos marcas.
Después de la ocurrencia obligatoria del evento ***c*** en la marca 3 y hasta la ocurrencia del evento ***d*** en la marca 4 inclusive no se desea que ocurra el evento ***c3***.
Después debe ocurrir un evento ***g*** en la marca 5 y hasta antes que se produzca el evento ***h*** en la marca 6 no se desea que ocurra el evento ***c4***.
Luego ocurre el evento ***h*** en la marca 6 y hasta antes que se produzca el evento ***k*** no se desea que ocurra el evento ***c5***.
Después debe ocurrir un evento ***k*** en la marca 7.

2.3.2.1 Herramienta gráfica

Si bien en el trabajo de Margaret Smith et al *Event and Constraints: A graphic Editor for Capturing Logic Requirements of Programs* [SHE01] se indica una dirección de Internet (actualmente inexistente) para obtener el software de TimeLine Editor, fue necesario pedir autorización a los autores para utilizarlo dentro del ámbito académico puesto que actualmente esta herramienta no es de acceso gratuito.
La Figura 8 muestra un gráfico obtenido con esta herramienta.

2.3.3 Aspecto textual

Esta herramienta de especificación es totalmente gráfica y no posee asociada ninguna descripción textual, como LTL, ya que eso era lo que se pretendía evitar al desarrollar el TimeLine Editor.

2.3.4 Aspecto formal

Hasta donde sabemos, no hay publicaciones que demuestren una especificación formal de este lenguaje, por lo que se ha desarrollado esta notación formal en este trabajo.

En TimeEdit un escenario es una tupla $\langle E; M; a; t; S; \alpha \rangle$ donde:

E es un conjunto de eventos,
$M = \{M_i / i=0, n\}$ es un conjunto finito totalmente ordenado de “marcas” o líneas,
$a : M \rightarrow E$ es una función que asigna a cada marca un evento,
$t : M \rightarrow \{e, f, r\}$ es una función que asigna a cada marca un tipo de evento. Los eventos pueden ser e (esperados), f (de falla) o r (requeridos),

$S = TT \cup TF \cup FT \cup FF$ es el conjunto de segmentos o "constrains". Este conjunto está formado por las siguientes particiones:

$TT \subseteq M \times M$ son las restricciones que incluyen las marcas de inicio y final

$TF \subseteq M \times M$ son las restricciones que incluyen las marcas de inicio pero no las del final

$FT \subseteq M \times M$ son las restricciones que incluyen las marcas de final pero no las de inicio

$FF \subseteq M \times M$ son las restricciones que no incluyen las marcas de inicio ni las de final

Todas estas particiones están formadas por pares $\langle M_i, M_j \rangle$ con i ≤ j. Y

$\alpha : S \rightarrow 2^E$ es la función que asocia a cada segmento un conjunto de eventos prohibidos.

Semántica

Para interpretar la esencia de TE se debe tener en cuenta las siguientes definiciones que surgen a partir de la traducción del gráfico TE a un autómata.

En el trabajo en que se presenta esta herramienta [SHE01] se plantea una traducción de los distintos elementos de un gráfico TE a un autómata.

- Eventos: corresponden a los distintos sucesos asociados a las marcas que permiten pasar de un estado a otro que puede asumir el sistema en cuestión.
- Estados: situación en la que se encuentra el sistema mientras no suceden los eventos de la siguiente marca.
- Transiciones: Cuando sucede el evento de la marca y se cumplen las restricciones asociadas a ésta, se pasará al próximo estado.
- Secuencia temporal: es una secuencia de eventos creciente de tiempo comenzando desde la marca 0 o start.

En esta traducción, se pueden observar distintos estados dependiendo del tipo de evento asociado a cada marca: los eventos esperados permiten asumir estados aceptados (circulo simple) (ver Figura 6), los eventos requeridos generan estados no aceptados (doble círculo) mientras no suceden y los eventos de falla presentan estados finales por error si suceden (ver Figura 7).

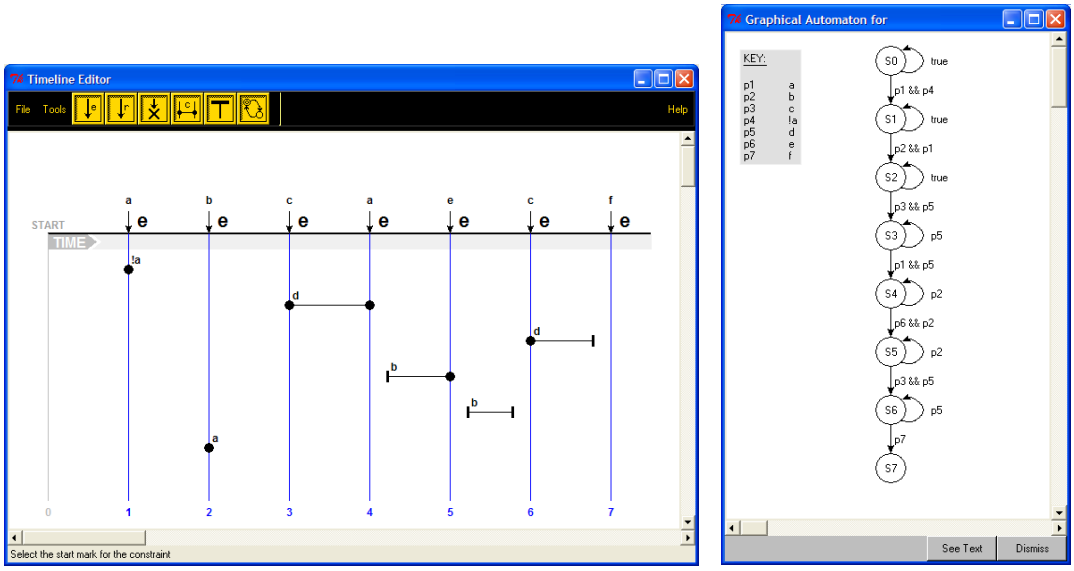


Figura 6: Ejemplo de escenario TE con eventos esperados y su autómata correspondiente

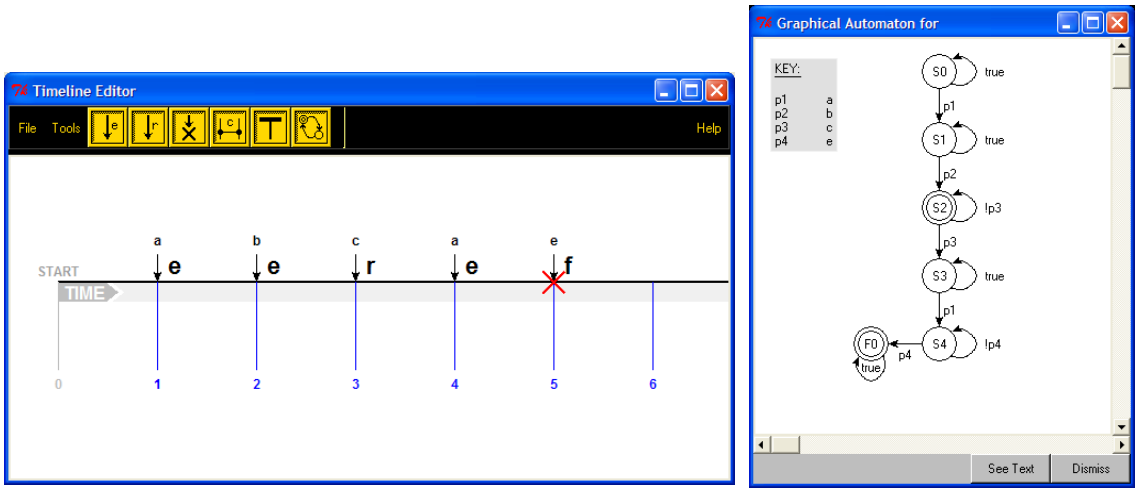


Figura 7: Ejemplo de escenario TE con eventos requerido y de falla y su autómata correspondiente

Una traza (secuencia de eventos) sin eventos de falla o error corresponde a un escenario deseado (propiedad de liveness).

2.3.5 Ejemplo:

En la Figura 8 se observa un gráfico obtenido con el editor de TE. En este gráfico se han dibujado 7 momentos (líneas verticales) con sus respectivos eventos de los cuales los primeros 5 son esperados, el siguiente requerido y el último un error. También se determinaron los eventos prohibidos entre los distintos momentos (líneas horizontales).

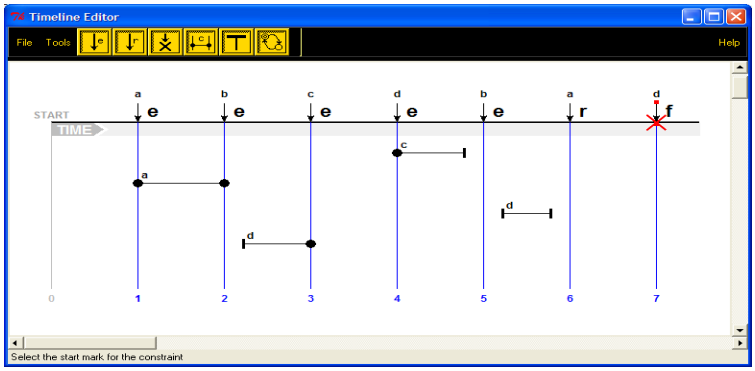


Figura 8: Ejemplo del editor TE

La descripción formal correspondiente al escenario de la Figura 8 es:

$S = \langle E; M; a; t; S; \alpha \rangle$ donde:

E = {a; b; c; d}
M = {0; 1; 2; 3; 4; 6; 7}
a = {0 →start; 1 →a; 2 →b; 3 →c; 4 →d; 5 →b; 6 →a; 7 →d}
t = {1 →e; 2 →e; 3 →e; 4 →e; 5 →e; 6 →r; 7 →f}
TT = {⟨1, 2⟩}
TF = {⟨4, 5⟩}
FT = {⟨2, 3⟩}

FF = {⟨5, 6⟩}
S = {⟨1, 2⟩; ⟨4, 5⟩; ⟨2, 3⟩; ⟨5, 6⟩}
α = {⟨1, 2⟩→{a}; ⟨4, 5⟩→{c}; ⟨2, 3⟩→{d}; ⟨5, 6⟩→{d}}

2.4 Message Sequence Charts (MSC)

2.4.1 ¿Cómo surge?

MSC es un lenguaje gráfico para la especificación de trazas de sistemas, en particular sistemas distribuidos, que se enfocan en el intercambio de mensajes entre entidades y su entorno.

A fines de los '80 ya se había comenzado a utilizar diagramas auxiliares SDL en SCs (Sequence Charts) utilizando esta visualización gráfica (trazas) en sistemas de comunicación [BKO05] [GGR93].
Debido al gran interés sobre Sequence Charts encontrado en los foros de SDL, el CCITT (*Comité Consultivo Internacional Telegráfico y Telefónico)* sugiere la estandarización tanto en su parte gráfica como textual del uso del SDL para estos fines. Surge así, en junio de 1990 un nuevo lenguaje llamado Message Sequence Charts (MSC).
Primeramente el MSC recomendado por Z.120 fue aprobado por ITU (International Telecomunication Union) en 1992. La popularidad de este lenguaje creció más allá de lo esperado. Antes de 1992: se define la dialéctica de SC, y desde 1992 hasta el 2000 se hicieron las siguientes mejoras al MSC:

- En 1992: se definen instancias, mensajes, eventos y condiciones.
- En 1996: se amplía la estructura de MSC a HMSC (High_level MSC) y se definen el orden general y las *inline expressions.*
- En 2000: se define el manejo del tiempo, se adapta a OO (Object Oriented) y se incorporan las llamadas a métodos remotos.

2.4.2 Aspecto gráfico

En un marco con encabezado que delimita el **entorno**, se indican los distintos actores involucrados o **instancias** y una línea vertical que hace de **línea de tiempo** correspondiente a cada una de estas instancias. Con flechas que parten de una instancia a otra se indican los flujos de los mensajes, indicando el origen de la flecha un **evento** de tipo enviar (send) y la punta de la flecha un **evento** de tipo recibir (receive). Sobre estas flechas se indica el **mensaje** enviado.

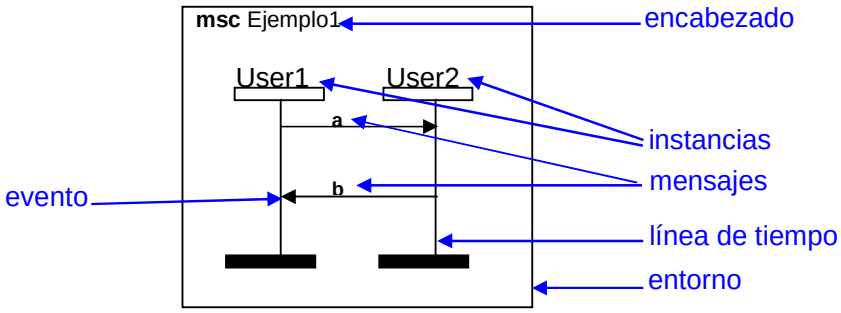


Figura 9: Elementos de un MSC

El ejemplo que se muestra en la Figura 9 representa un MSC básico. Además de estos elementos podemos encontrar las siguientes estructuras:

- **Creación y finalización de instancias**

En el gráfico de la Figura 10 se muestra cómo la instancia Autorizar es creada (línea punteada) y es finalizada (cruz) en el mismo MSC

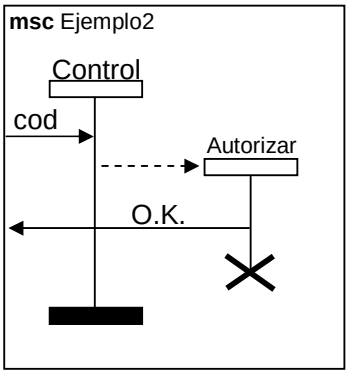


Figura 10: Creación y finalización de instancias en MSC

- **Mensajes perdidos y mensajes encontrados**

En la Figura 11 se observa que:

- ✓ la instancia User A envía un mensaje ***b*** que es recibido por la instancia User B.
- ✓ la instancia User A envía un mensaje ***a*** que no es recibido por la instancia User B (mensaje perdido).
- ✓ la instancia User B recibe un mensaje ***c*** que no es enviado por la instancia User A (mensaje encontrado).

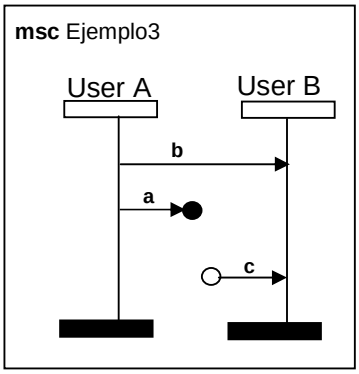


Figura 11: Mensajes perdidos y encontrados en MSC

- **Co-región**

La co-región se representa por una zona de línea punteada en una instancia y significa que todos los eventos que suceden en esta zona pueden darse en cualquier orden. En la Figura 12, la instancia User puede recibir el mensaje ***b*** y luego el mensaje ***c*** o recibir primero el mensaje ***c*** y luego el mensaje ***b***.

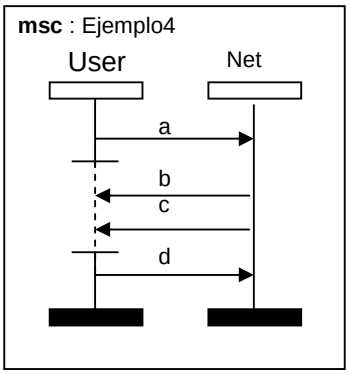


Figura 12: Co-región en MSC

- **Timers**

Al incorporar el objeto timer se pueden observar las siguientes opciones:

✓ **Iniciar el timer y detenerlo**

En el gráfico de la Figura 13, luego de enviar el mensaje ***b***, la instancia Net inicializa el timer ***T1*** que tendrá una duración menor a 20 unidades de tiempo. Cumplido el plazo, el timer se detiene y esta instancia recibe el mensaje ***c***.

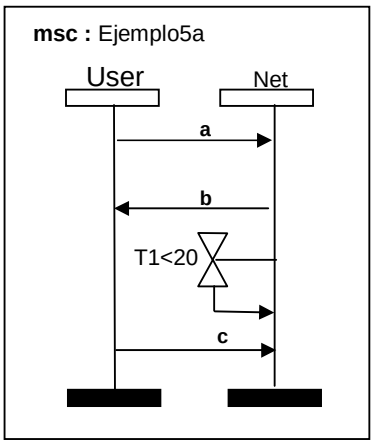


Figura 13: Iniciar un timer y detenerlo en MSC

✓ **Iniciar el timer y resetearlo**

En el gráfico de la Figura 14, luego de recibir el mensaje ***a***, la instancia Net inicializa el timer ***T1*** (sin indicar su duración en unidades de tiempo), envía el mensaje ***b*** y resetea el timer antes de recibir el mensaje ***c***.

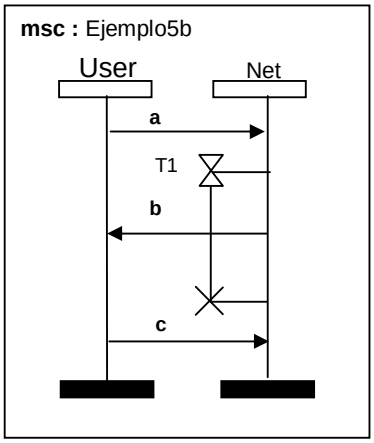


Figura 14: Iniciar un timer y resetearlo en MSC

- **Expresiones condicionales**

Las expresiones condicionales o in-line expressions son:

✓ **Alt (alternativas)**: En una sección se presentan distintas alternativas de ejecución separadas por una línea punteada. En caso de no darse una de las alternativas expresadas se estará en una situación de error. Luego sigue la secuencia. En la Figura 15 puede suceder:

- se envía el mensaje **a** y luego que se envíe los mensajes ***b*** y ***a,*** o
- se envía el mensaje **a** y luego se envíen los mensajes ***b***, ***c*** y ***a***.

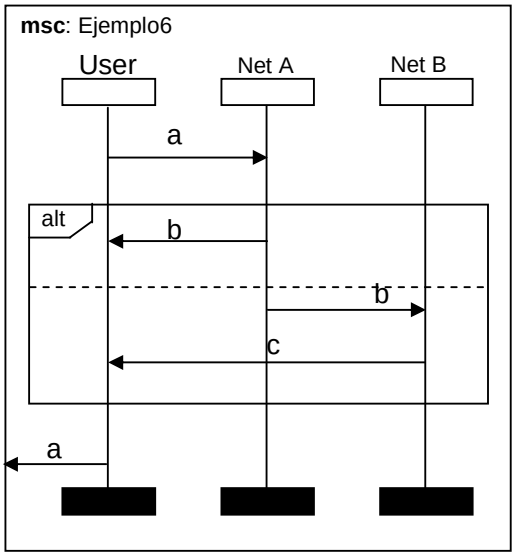


Figura 15: Alternativas en MSC

✓ **Loop <n, m> (iteraciones)**: Se ejecuta la sección del loop n veces como mínimo y m veces como máximo. En la Figura 16, se enviarán los mensajes ***b*** y ***c,*** repitiéndose estos envíos 2, 3, 4 o 5 veces, y por último se enviará el mensaje ***d***.

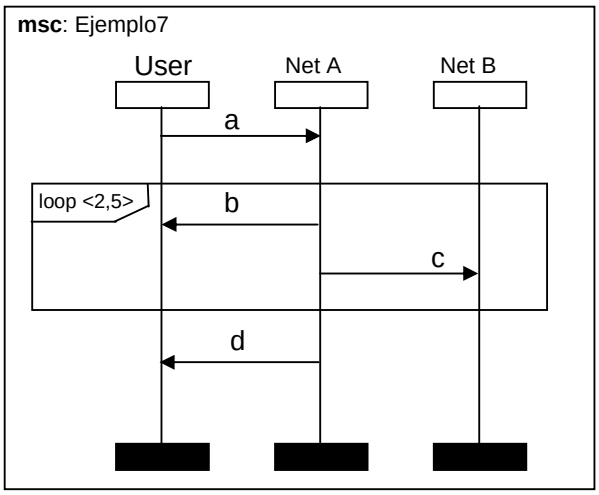


Figura 16: Loop en MSC

- ✓ **Exc (excepción)**: Se ejecutan los elementos de la sección exc y después el MSC termina o se ejecuta los eventos que siguen a esta sección. En la Figura 17 se enviará el mensaje ***a*** y luego puede suceder que:
  - se envíe el mensaje ***b*** y finalice el MSC, o
  - se envíe el mensaje ***c***.

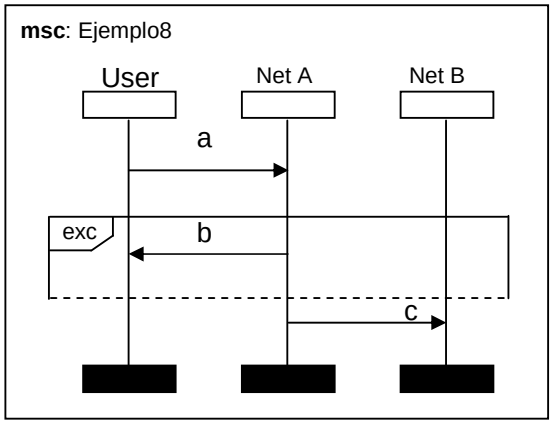


Figura 17: Excepción en MSC

- ✓ **Opt (opción)**: Es igual a un alt donde la segunda opción está vacía. En la Figura 18 se enviará el mensaje ***a***, puede que se envíe el mensaje ***b*** (es optativo) y luego se envía el mensaje ***c***.

- ✓ **Seq (secuencia)**: Es una simple operación secuencial, los eventos suceden en el orden en que aparecen en esta sección.

- ✓ **Par (paralelo)**: Todas las secciones se ejecutan. La única restricción es que el orden está preservado internamente en cada sección.

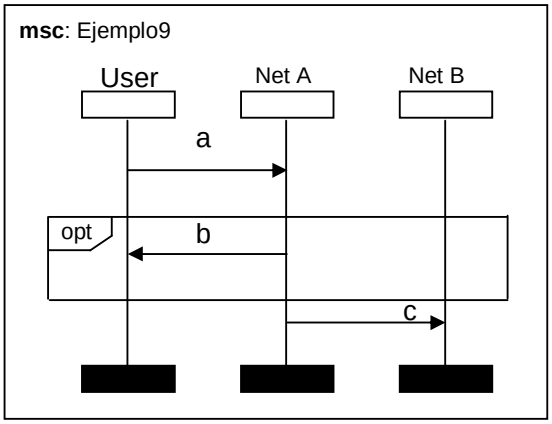


Figura 18: Opción en MSC

- **HMSC**

Un gráfico HMSC (High_level MSC) muestra cómo se combinan distintos gráficos MSC [RGG96] y cómo se relacionan entre sí. En un HMSC se expresan a través de flujos de control distintas composiciones y alternativas [GHM03] [MR97].
Un HMSC consta de distintos tipos de nodos unidos por flechas. En la recomendación Z.120 se indica una restricción: todos los nodos deben ser alcanzados desde el nodo de inicio.
Un ejemplo de cómo se grafica un HMSC sería lo que se muestra en la Figura 19.

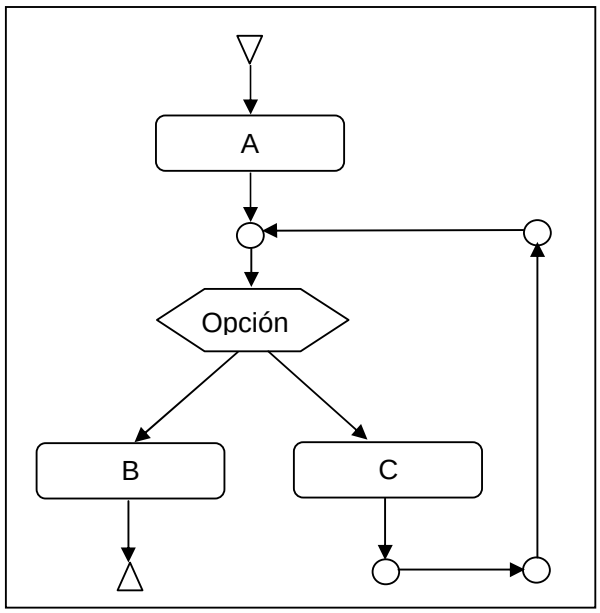


Figura 19: Ejemplo de gráfico HMSC

En este trabajo se tendrá en cuenta a los gráficos MSC, no los HMSC, para realizar el estudio comparativo entre los lenguajes de especificación seleccionados.

2.4.2.1 Herramientas gráficas

Debido al uso difundido del MSC, existen diferentes herramientas que han incorporado plantillas que permiten realizar estos gráficos. Entre estos software encontramos:

- ✓ Microsoft Visio presenta la extensión Sequence Chart Studio (http://scstudio.sourceforge.net/).
- ✓ Se puede obtener de forma gratuita un paquete para generar gráficos MSC con LaTeX2e (http://www.tex.ac.uk)/.
- ✓ EventStudio también tiene una herramienta que genera MSC (http://www.eventhelix.com/eventstudio/).
- ✓ Trace2UML (http://trace2uml.tigris.org/) y SmartDraw (http://www.smartdraw.com/) permiten entre sus diagramas realizar gráficos MSC.

También existen herramientas específicas para hacer gráficos MSC. Por ejemplo:

- ✓ MscGen (http://www.mcternan.me.uk/mscgen/).
- ✓ MscTracer (http://www.pragmadev.com/product/tracing.html).
- ✓ Sequence Diagram Editor (http://www.sequencediagrameditor.com/).

2.4.3 Aspecto textual

Si bien MSC es más conocido por su aspecto gráfico también cuenta con una parte textual. En la publicación Z.120, ITU Recomendation [ITU04], se indica cómo se dibuja cada elemento de un gráfico MSC y cómo se define textualmente. En este documento se hace mención a un meta lenguaje que fue evolucionando con las distintas series Z100 de las recomendaciones del ITU. Este lenguaje se denomina SDL (http://www.sdl-forum.org/MSC/index.htm) y cuenta con una estructura bien definida donde se describe cada componente del gráfico MSC con un identificador y calificadores [R98].

Utilizando como ejemplo la Figura 9, se describe textualmente al MSC según las recomendaciones del Z.120

```
msc ejemplo1;
        instance User1;
                out a to User2;
                in b from User2;
        endinstance;
        instance User2;
                in a from User1;
                out b to User1;
        endinstance;
endmsc;
```

También se permite escribir el gráfico de la Figura 9 de la siguiente forma:

```
msc ejemplo1;
        User1: instance;
        User2: instance;
        User1: out a to User2;
        User1: in b from User2;
        User2: in a from User1;
        User2: out b to User1;
        User: endinstance;
        Net  : endinstance;
```

endmsc;

En este ejemplo los eventos están descriptos en el orden en que aparecen en cada instancia (FIFO). Esto no es requerido por la recomendación del ITU. Con el MSC96 apareció otra forma de describir un MSC denominada *event-oriented description*. En este caso se enumeran los eventos en el orden en que se espera que ocurran recorriendo el MSC de arriba hacia abajo. En esta forma se nota la aplicación del orden vertical (en cada instancia) y horizontal.

Entonces el ejemplo anterior también se lo podría describir como:

msc ejemplo1;
User1: ***instance***;
User2: ***instance;***
User1: ***out*** a ***to*** User2;
User2: ***in*** a ***from*** User1;
User2: ***out*** b ***to*** User1;
User1: ***in*** b ***from*** User2;
User: ***endinstance;***
Net : ***endinstance***;
endmsc;

En el Anexo **A** se describen textualmente los ejemplos de MSC presentados en la sección 2.4.2 sobre aspecto gráfico.

2.4.4 Aspecto formal

Debido a la complejidad con que nos encontramos al pasar de un gráfico MSC a una definición formal completa que abarque todos los aspectos que se observan en dicho gráfico, en este trabajo se considera la definición formal de un MSC básico.
Existen distintos trabajos presentando una definición formal de MSC:

- ✓ *Message Sequence Charts* de D. Harel y P.S. Thiagarajan [HT03]
- ✓ *The formalization of Message Sequence Charts* de S Mauw [M96]
- ✓ *High-Level Message Sequence Charts and Projections* de B. Genest, L. Hélouët y A. Muscholl [GHM03]
- ✓ *Quantifying the Discord: Order Discrepancies in Message Sequence Charts* de E. Elkind, B. Genest, D. Peled, y P. Spoletini [GPS07]
- ✓ *Model checking of Message Sequence Charts* de R. Alur y N. Yannakakis [AY99]
- ✓ *Analyzing Message Sequence Charts* de A. Muscholl y D. Peled [MP00]

En este trabajo se adoptó la notación propuesta por A. Muscholl y D. Peled.

Se define un MSC como una tupla $\langle V, <, P, \mathcal{N}, L, T, N, m \rangle$ donde:

V es un conjunto (finito) de eventos,

$< \subseteq V \times V$ es una relación de orden parcial sobre V, y tiene su clausura transitiva y reflexiva $<^*$,

P es un conjunto de procesos,

$\mathcal{N}$ es un conjunto de nombres de mensajes,

$L: V \to P$ asocia a cada evento un proceso,

$T: V \to \{s, r, l\}$ describe cada evento como s (send), r (receive) y l (local),

$N: V \to \mathcal{N}$ asocial cada evento con un nombre de mensaje, y

m: $V \rightarrow V$ función parcial que asocia un evento send y un receive. Cada evento $s \in V$ se relaciona exactamente con un evento $r \in V$ y viceversa que denota que $m(s) = r$ y $m(r) = s$. Cada evento e y f puede ser asociado con otro solamente si $N(e) = N(f)$.

Esta definición formal, al igual que las mencionadas anteriormente, se refiere a un MSC básico (bMSC). Se puede definir un bMSC como un escenario donde se asume un conjunto finito de instancias (agentes, objetos), un conjunto finito de procesos (enviar y recibir mensajes y también acciones internas) y un conjunto finito de mensajes [HT03].

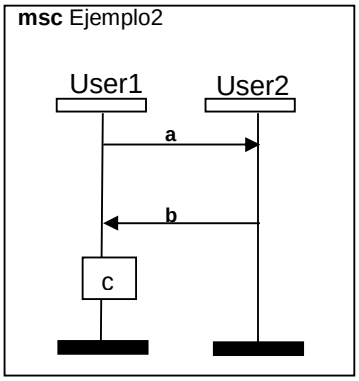
Semántica
Así como se encuentran distintos trabajos que tratan la descripción formal del lenguaje MSC, también podemos encontrar diferentes trabajos que abordan la semántica del MSC.

- En el trabajo que presenta S. Mauw [M96] sobre la formalización de MSC hace mención a dos tipos de definiciones semánticas para los lenguajes de programación: una observa el aspecto de la denotación y la otra el aspecto operacional.
  La que define el aspecto de la denotación consiste en la traducción de una expresión del lenguaje en una expresión matemática mientras que la definición semántica según el aspecto operacional consiste en un procedimiento para transformar una expresión en la realización de un paso de ejecución resultando una nueva expresión después de realizado ese paso.
  La primera de estas definiciones permite, por ejemplo, analizar formalmente casos de MSC donde se descomponen en sub MSC, identificar si dos MSC son idénticos o cuántas trazas genera un MSC.
  Con la definición semántica operacional se puede realizar prototipos o testear una expresión.
  El autor de este trabajo ubica la definición semántica de MSC que se describe en el documento Z.120 [ITU04] como una descripción informal y de alguna forma operacional.

- En el trabajo de Reniers [R98] también se presenta una definición semántica de MSC. Para eso se define a un documento MSC como un conjunto de MSCs. A cada MSC del documento se asocia una ecuación. Por ejemplo la ecuación $\overline{A}$ = S donde $\overline{A}$ es la variable asociada al MSC de nombre A y S es la definición semántica del cuerpo de este MSC.
  Entonces un documento MSC se presenta como un conjunto de ecuaciones.
  En este trabajo se encuentran detalladas las definiciones semánticas de cada elemento de un MSC.

- En el trabajo *Scenario-based Validation and Verification for Real-Time Software: On Run Conformance and Coverage for MSC-Graphs* [BOB03] se presenta otra definición semántica diferente. En este caso se asume un MSC como un conjunto de bMSC y se define semánticamente un gráfico MSC como un conjunto de eventos temporales sin repetición, contenidos en estos bMSC. La secuencia de eventos debe cumplir con dos criterios: causalidad y restricciones temporales. La semántica de un gráfico MSC estará dada por la concatenación asincrónica de gráficos bMSC (los eventos de un segundo bMSC se agregan a continuación de los eventos del primer bMSC según el orden visual de cada instancia).

2.4.5 Ejemplo
Se asume el siguiente escenario MSC:



Se puede identificar los siguientes eventos:

s1: envío del mensaje a
r2: recepción del mensaje a
s3: envío del mensaje b
r4: recepción del mensaje b
l5: actividad c

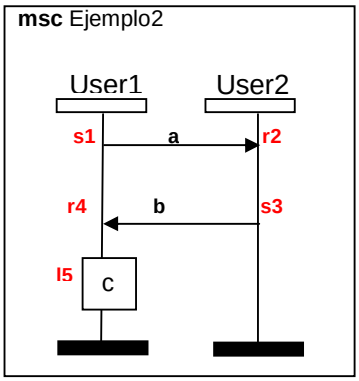


Figura 20: Asignación de eventos en un escenario MSC

Su descripción formal es:

msc *Ejemplo2* = $\langle V, <, P, \mathcal{N}, L, T, N, m \rangle$

V = {s1, r2, s3, r4, l5}
$<$ = {(s1, r2), (r2, s3), (s3, r4), (s1, r4), (r4, l5)}
P = {User1, User2}
$\mathcal{N}$ = {a, b}
L = {s1 → User1, r2 → User2, s3 → User2, r4 → User1, l5 → User1}
T = { s1 → s , r2 → r , s3 → s , r4 → r , l5 → l }
N = {s1 → a, r2 → a, s3 → b, r4 → b}
m = {s1 → r2, r2 → s1, s3 → r4, r4 → s3}

2.4.6 Ejemplos de aplicaciones de gráficos MSC

Se pueden encontrar varios trabajos donde se ha utilizado MSC en diferentes casos. Algunos de estos trabajos son:

- Genest y Muscoll [GM05] presentan un análisis formal de especificaciones basadas en MSC relacionadas con máquinas de estados finitos.

- D. Harel y P. S. Thiagarajan [HT03] en su presentación enumeran una serie de trabajos relacionados con MSC.
- D. Peled [P02] presenta un trabajo de especificación y verificación de protocolo de comunicación.
- Darondeau, Genest, Hélouët [DGH08] presentan un trabajo sobre aplicar MSC a sistemas distribuidos.
- En el sitio oficial del foro de SDL (http://www.sdl-forum.org/Publications/Papers.htm) se puede acceder a varios trabajos relacionados con aplicaciones de MSC.

2.5 Visual Timed event Scenarios (VTS)

2.5.1 ¿Cómo surge?

La herramienta VTS [BKO05] [ABKO04] surge como una necesidad de contar con un lenguaje gráfico basado en eventos donde se puede expresar de manera natural causalidad, restricciones temporales entre ocurrencias de eventos (ordenados o no), el primero (first) o el último (last) elemento en acontecer en un grupo de eventos entre otras cosas. Se presenta entonces a VTS como un lenguaje visual para definir requerimientos basados en eventos de una manera simple y amigable, destinado a usuarios no expertos en formalismos (como lógicas lineales (LTL), arborescentes, modales (CTL) y temporales (TCTL), etc.).
VTS cuenta con un editor de escenarios y un traductor de estos escenarios a modelos de autómatas con tiempo para chequear el cumplimiento de los requerimientos por parte del modelo del sistema en análisis. En la actualidad también se han desarrollado traductores a Redes de Petri con tiempo y lenguaje SAL.

2.5.2 Aspecto gráfico

La notación gráfica de VTS se basa en puntos y flechas.

Puntos

- Los puntos representan eventos que ocurren durante la ejecución. Son etiquetados por un conjunto de eventos (puede ser vacío) y significa que ocurre un evento del conjunto. El nombre es sólo de referencia y es optativo.

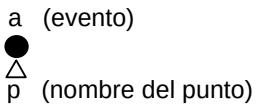


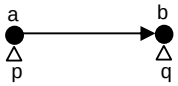
Figura 21: Componentes de un escenario VTS: Puntos

- Se representa con diferentes símbolos el comienzo y el final de una corrida del sistema. Permite hablar de eventos que ocurren (o no) desde el comienzo o hasta el final.

Figura 22: Componentes de un escenario VTS: Símbolos de inicio y final

Flechas

- Las flechas representan relaciones de precedencia entre los puntos.

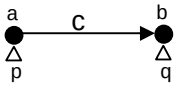


p precede a ***q***

Figura 23: Componentes de un escenario VTS: Flechas

Restricciones

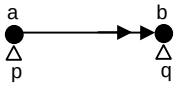
- Sobre las flechas se indican los eventos prohibidos entre los puntos relacionados por dichas flechas.



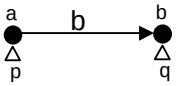
entre ***a*** y ***b*** no debe observarse el evento ***c***

Figura 24: Componentes de un escenario VTS: Evento prohibido

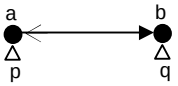
- Las dobles flechas son otra forma de indicar ciertos eventos prohibidos.



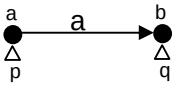
es equivalente a



en ***q*** se observa el próximo evento ***b*** después de ***p***

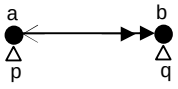


es equivalente a

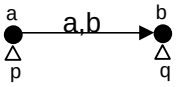


en ***p*** se observa el evento ***a*** anterior a ***q***

Uniendo los dos casos:



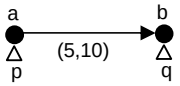
es equivalente a



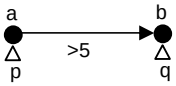
en ***p*** se observa el evento ***a*** anterior a ***q***
y en ***q*** se observa el próximo evento ***b*** después de ***p***

Figura 25: Componentes de un escenario VTS: Restricciones de eventos anteriores o siguientes

- Por debajo de las flechas se puede indicar las restricciones sobre la distancia temporal (ejemplo: intervalos de tiempo) que debe cumplirse entre los eventos de los puntos unidos por estas flechas. Estos intervalos son abiertos o cerrados con límites naturales.



se debe observar un evento ***b*** después de un evento ***a*** entre 5 y 10 unidades de tiempo

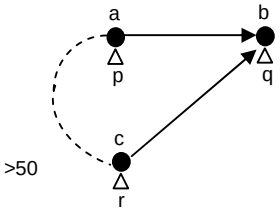


se debe observar un evento ***a*** y luego un evento ***b*** después de transcurridas 5 ó más unidades de tiempo

Figura 26: Componentes de un escenario VTS: Restricción temporal entre puntos unidos por flechas

- Con una línea entrecortada se denotan restricciones temporales entre puntos donde no interesa el orden.

se deben observar distintos eventos **a** en ***p*** y ***q***

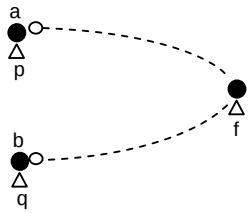


se deben observar los eventos **a** y **c** **(ambos antes de b)** sin importar el orden pero separados por más de 50 unidades de tiempo

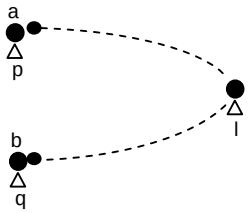
Figura 27: Componentes de un escenario VTS: Restricción temporal entre puntos sin unir por flechas

Primero y último

- Se puede representar al primer y último evento que ocurre de un conjunto de eventos. Esto permite hablar del primero o último evento de un conjunto sin necesidad de saber cuál es.



en ***f*** se observará el primer evento que ocurra entre ***p*** y ***q***



en ***l*** se observará el último evento que ocurra entre ***p*** y ***q***

Figura 28: Componentes de un escenario VTS: Primero o último de un conjunto de puntos

Escenarios condicionales

Otra alternativa que brinda VTS es la posibilidad de definir escenarios condicionales [BKO05]. Estos son utilizados en los casos en que se encuentra coincidencia en una traza dada para un sub-escenario (antecedente) y se debe extender la misma coincidencia sobre un conjunto se escenarios posibles (consecuentes). En la Figura 29 se muestra un ejemplo de escenario condicional pudiéndose observar que los escenarios consecuentes 1 (en azul) y 2 (en naranja) comparten el antecedente. La interpretación de este escenario es la siguiente: si se observa la secuencia de eventos ***a b a*** entonces se deberá observar a continuación una de las 2 secuencias de los consecuentes 1 o 2.

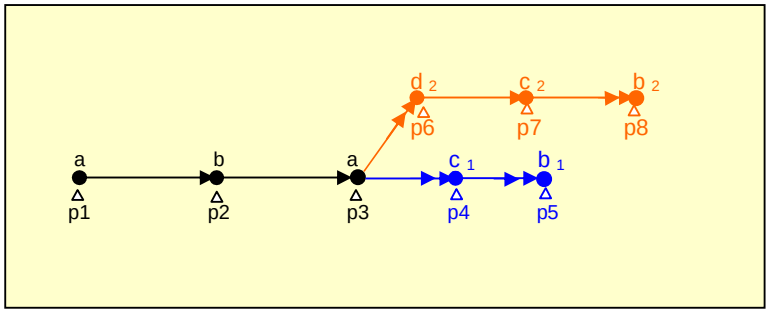


Figura 29: Ejemplo 1 de escenario condicional de VTS

Cabe recordar que los consecuentes no necesariamente vienen a continuación del escenario antecedente, sino que pueden ser previos a éste o estar vinculados a eventos que ocurren en el medio de dicho antecedente.

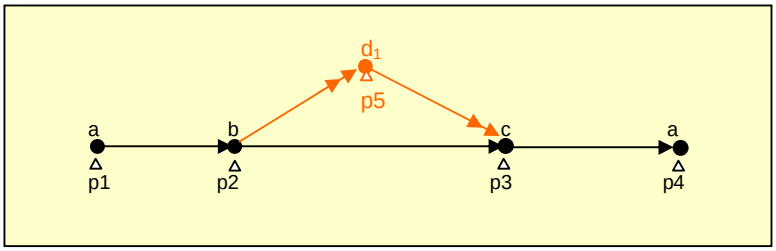


Figura 30: Ejemplo 2 de escenario condicional en VTS

En la Figura 30 se muestra un ejemplo de escenario condicional donde se observa el consecuente (en naranja) en medio del antecedente. La interpretación de este escenario es que si se observa la secuencia de eventos ***a b c a*** entonces se deberá observar un evento ***d*** entre ***b*** y ***c***.

2.5.2.1 Herramienta gráfica

Se puede bajar desde el sitio http://lafhis.dc.uba.ar/vts/ el template para VISIO que permite graficar escenarios en VTS. En la Figura 31 se observa un escenario VTS generado con dicha plantilla.

2.5.3 Aspecto textual

Utilizando la plantilla de VISIO mencionada en el punto anterior se pueden generar escenarios de VTS y exportar el gráfico obtenido como un archivo XML (eXtensible Markup Language). Este meta lenguaje tiene un formato bien definido y utilizando etiquetas permite definir distintas estructuras o elementos. De esta manera se obtiene una definición textual del escenario graficado que puede ser leído con cualquier editor de texto.

El texto XML que describe un escenario VTS define los siguientes elementos:

- ✓ **Escenario**: Al inicio del texto encontramos la declaración de inicio del escenario y al final de la descripción de todos los elementos del escenario se indica el cierre del mismo.

```
<scenario id='nombre_escenario'>
:
</scenario>
```

- ✓ **Eventos**: Se definen todos los eventos dentro de una sección que comienza y finaliza con la declaración *events*.

```
<events>
        <event>e1</event>
        <event>e2</event>
        <event>en</event>
</events>
```

- ✓ **Puntos**: Se definen todos los puntos y los eventos asociados a los mismos dentro de una sección que comienza y finaliza con la declaración *points*.

```
<points>
        <point id='p1'>
                <event>e1</event>
        </point>
        <point id='p2'>
                <event>e2</event>
        </point>
</points>
```

- ✓ **Precedencia**: Se define la precedencia entre dos puntos dentro de una sección que comienza y finaliza con la declaración *precedence*.

```
<precedence>
        <precedes>
                <point-ref id='p1'/>
                <point-ref id='p2'/>
        </precedes>
</precedence>
```

- ✓ **Eventos prohibidos**: Al indicar la precedencia entre dos puntos se indican los eventos prohibidos entre ellos. En caso de una flecha en el gráfico con doble punta se indicará con *marks='next'* y caso de una flecha con punta en ambos extremos con *marks='prev'.* Los eventos prohibidos entre los puntos se indican con *event*.

```
<precedes>
        <point-ref id='p1'/>
        <point-ref id='p2'/>
                <event>e1</event>
</precedes>
<precedes marks='next'>
```

```
        <point-ref id='p2'/>
        <point-ref id='p3'/>
</precedes>
<precedes marks='prev'>
        <point-ref id='p3'/>
        <point-ref id='p4'/>
</precedes>
```

- ✓ **Restricciones temporales**: Al indicar la precedencia entre dos puntos se indica el rango de la restricción temporal entre ellos.

```
<precedes>
        <point-ref id='p1'/>
        <point-ref id='p2'/>
                <interval>
                        <lower-bound value='0' included='false'/>
                        <upper-bound value='1' included='false'/>
                </interval>
</precedes>
```

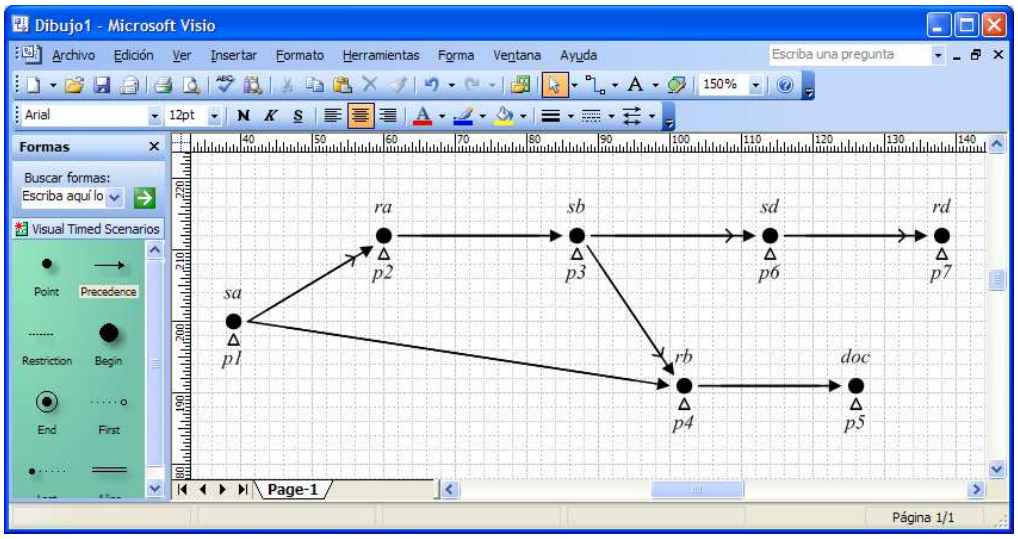


Figura 31: Ejemplo del editor de VISIO para gráficos VTS

Teniendo en cuenta el escenario VTS realizado en VISIO que se muestra en la Figura 31, se transcribe a continuación el texto XML obtenido a partir del mismo.

```
<scenario id=' ejemplo1'>
        <events>
                <event>sa</event>
                <event>ra</event>
                <event>sb</event>
                <event>rb</event>
                <event>doc</event>
                <event>sd</event>
                <event>rd</event>
        </events>
        <points>
                <point id='p1'>
                        <event>sa</event>
                </point>
```

```
                <point id='p2'>
                        <event>ra</event>
                </point>
                <point id='p3'>
                        <event>sb</event>
                </point>
                <point id='p4'>
                        <event>rb</event>
                </point>
                <point id='p5'>
                        <event>doc</event>
                </point>
                <point id='p6'>
                        <event>sd</event>
                </point>
                <point id='p7'>
                        <event>rd</event>
                </point>
        </points>
        <precedence>
                <precedes marks='next'>
                        <point-ref id='p1'/>
                        <point-ref id='p2'/>
                </precedes>
                <precedes>
                        <point-ref id='p2'/>
                        <point-ref id='p3'/>
                </precedes>
                <precedes>
                        <point-ref id='p1'/>
                        <point-ref id='p4'/>
                </precedes>
                <precedes marks='next'>
                        <point-ref id='p3'/>
                        <point-ref id='p4'/>
                </precedes>
                <precedes>
                        <point-ref id='p4'/>
                        <point-ref id='p5'/>
                </precedes>
                <precedes>
                        <point-ref id='p3'/>
                        <point-ref id='p6'/>
                </precedes>
                <precedes marks='next'>
                        <point-ref id='p6'/>
                        <point-ref id='p7'/>
                </precedes>
        </precedence>
</scenario>
```

2.5.4 Aspecto formal

El desarrollo formal de esta herramienta es el siguiente [BKO05]:

En VTS un escenario es una tupla $\langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$ donde:

Σ es un conjunto finito de eventos,

P es un conjunto finito de puntos,

$\ell : P \to 2^{\Sigma}$ es la función donde a cada punto de P le corresponde un subconjunto de eventos de Σ, puede ser $\varnothing$,

$\not\equiv \subseteq P \times P$ es una relación asimétrica entre los puntos de desigualdad,

$<\subseteq \left(P \overset{+}{\bigcup} \{0\} \times P \overset{+}{\bigcup} \{\infty\} \right) \setminus \{(0,\infty)\}$ es una relación de precedencia entre los puntos, 0 y ∞ representan el comienzo y el final de la ejecución respectivamente,

$<_F \subseteq P \times P$ es una relación que representa el primer punto en un conjunto,

$<_L \subseteq P \times P$ es una relación que representa el último punto en ocurrir,

$\gamma : (\not\equiv \bigcup <) \to 2^{\Sigma}$ es la función que asigna a cada par de puntos un conjunto de eventos prohibidos entre ellos, y

$\delta : (\not\equiv \bigcup <) \to \Phi$ es la función que asigna a cada par de puntos una restricción de tiempo entre dos puntos, donde Φ es el conjunto de todas las restricciones de tiempo.

Semántica

Para comprender la filosofía de VTS se deben tener en cuenta las siguientes definiciones:

- *Secuencia*: En un conjunto dado los elementos del mismo tienen un orden o posición.
- *Secuencia temporal*: Es una secuencia creciente de tiempo representada por números reales no negativos.
- *Corrida o trazas*: Es un conjunto de pares de secuencias de instantes con o sin eventos asociados y una secuencia temporal de la misma longitud.

La formalización de estas definiciones se encuentra en [BKO05] y [ABKO04].

En la semántica de VTS se asigna a cada escenario un conjunto de corridas que lo satisfacen.

Los puntos con etiquetas representan eventos en la corrida. Estos eventos pueden coincidir en una posición particular de la corrida si el evento en esa posición está entre los eventos permitidos asociados al punto por la función ℓ (ver definición de escenario).

Los puntos sin etiquetas son llamados instantes y representan momentos en la ejecución donde no ocurren eventos.

Una coincidencia o *match* es cuando vemos que los eventos entre los puntos en un escenario concuerdan con las posiciones en una corrida, mostrando cómo la corrida satisface el escenario.

Semántica existencial de VTS: Un sistema define un conjunto de corridas. Se dice entonces que un sistema satisface un escenario si existe una corrida que lo satisface. Si existe una corrida que satisface el escenario puede darse una de las siguientes situaciones:

- Si el escenario es “bueno” la satisfacción implica “OK” (liveness).
- Si el escenario es “malo” la satisfacción implica “ERROR” (anti-escenario).

Existe una extensión de VTS que se refiere a los *escenarios condicionales*: Para todo escenario, si la corrida satisface el antecedente $\Rightarrow$ satisface el consecuente (cuando hay un solo consecuente) o satisface alguno de los consecuentes (cuando son más de uno). Esto es, si hay una coincidencia entre una dada corrida y un sub-escenario (antecedentes), la misma coincidencia debe extenderse a uno de los escenarios consecuentes. En los escenarios condicionales, el escenario está dado por el sub-escenario antecedente S_0 y el conjunto de sub-escenarios consecuentes S_1, S_2 y S_3. O sea $S = \langle S_0, \{S_i\} i = 1; n \rangle$. En el ejemplo de la Figura 29 $S = \langle S_0, \{S_i\} i = 1;2 \rangle$ ya que tiene dos consecuentes como se describe en la siguiente sección de ejemplos.

2.5.5 Ejemplos:

- ✓ Considerando el siguiente escenario S en VTS

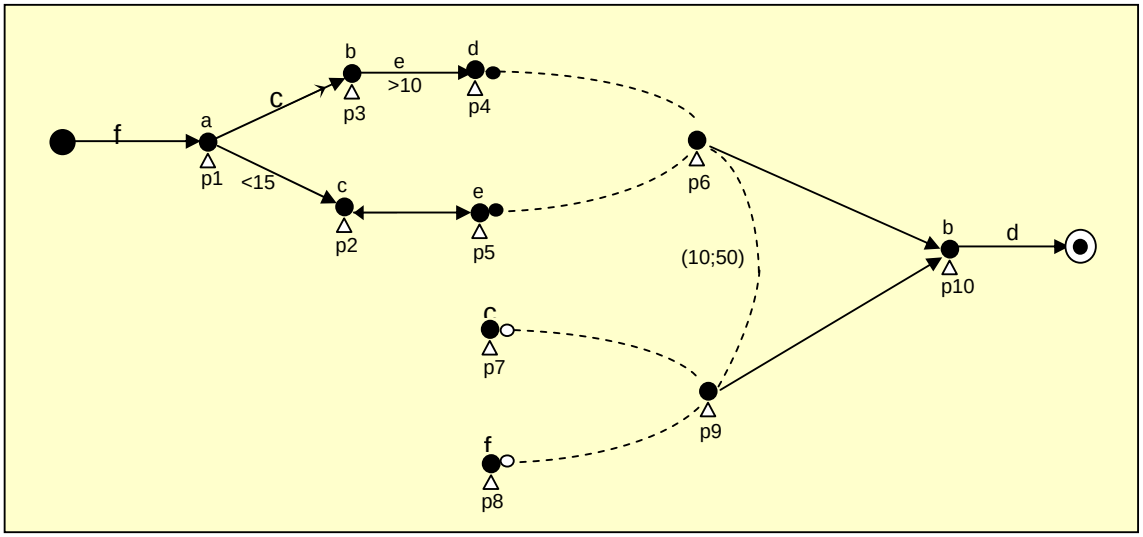


Figura 32: Ejemplo de escenario VTS

su descripción formal es :

$S = \langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$ donde:

Σ = {a; b; c; d; e; f}

P = {p1; p2; p3; p4; p5; p6; p7; p8; p10}

ℓ = {p1→{a}; p2→{c}; p3→{b}; p4→{d}; p5→{e}; p7→{c}; p8→{f}; p10→{b}}

$\not\equiv$ = {(p6, p9)}

$<$ = {(0, p1); (p1, p3); (p3, p4); (p1, p2); (p2, p5); (p6, p10); (p9, p10); (p10;∞)}

$<_F$ = {(p4, p6); (p5, p6)}

$<_L$ = {(p7, p9); (p8, p9)}

γ = {(0, p1) →{f}; (p1, p3) →{c, b}; (p2, p5) →{c}; (p3, p4) →{e}; (p10;∞) →{d}}

δ = {(p1, p2) → [0, 15); (p3, p4) →> (10,∞); (p6, p9) →(10, 50)}

✓ Tomando el gráfico de la Figura 29 se tendría la siguiente definición formal del escenario condicional:

La definición del antecedente sería:

$S_0 = \langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$ donde:

Σ = {a; b}

P = {p1; p2; p3}

ℓ = {p1 →{a}; p2 →{b}; p3 →{a}}

$\not\equiv$ = ∅

$<$ = {(p1, p2); (p2, p3)}

$<_F$ = ∅

$<_L = \varnothing$
$\gamma = \varnothing$
$\delta = \varnothing$

Las definiciones de los consecuentes son:

$S_1 = \langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$ donde: Σ = { a; b; c } P = {p1; p2; p3; p4; p5 } ℓ = { p1→{a}; p2→{b}; p3→{a}; p4→{c}; p5→{b}} $\not\equiv = \varnothing$ $<$ = { (p1,p2); (p2,p3); (p3,p4); (p4,P5)} $<_F = \varnothing$ $<_L = \varnothing$ γ = {(p3,p4) →{c}; (p4,P5) →{b}} $\delta = \varnothing$	$S_2 = \langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$ donde: Σ = { a; b; c; d } P = {p1; p2; p3; p6; p7; p8} ℓ = { p1→{a}; p2→{b}; p3→{a}; p6→{d}; p7→{c}; p8→{b} } $\not\equiv = \varnothing$ $<$ = { (p1,p2); (p2,p3); (p3,p6); (p6,p7); (p7,p8) } $<_F = \varnothing$ $<_L = \varnothing$ γ = {(p3,p6) →{d}; (p7,P8) →{b}} $\delta = \varnothing$

Entonces el escenario será $S = \langle S_0, \{S_i\} i = 1;2 \rangle$

2.5.6 Ejemplos de corridas

Considerando el siguiente escenario VTS

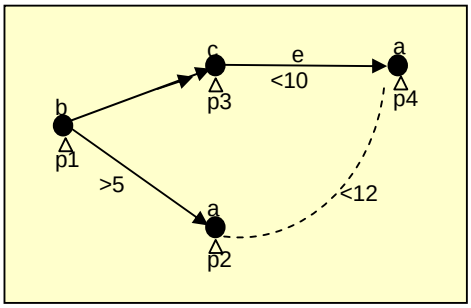
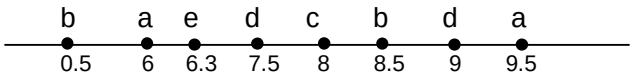


Figura 33: Ejemplo 1 de escenario VTS

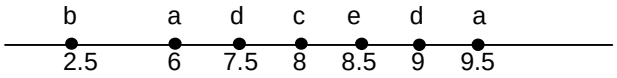
y las siguientes corridas del sistema graficadas en una línea de tiempo donde se muestran la secuencia de los eventos (sobre la línea) y en la unidad de tiempo (contando desde el comienzo de la corrida) en que se detectó ese evento (debajo de la línea), se puede sacar las siguientes conclusiones:

σ1



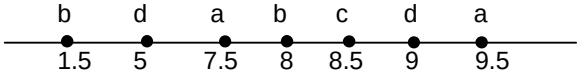
- ✓ esa porción de traza o corrida satisface el escenario. Notar que los eventos ***a*** no son el mismo y se producen con menos de 12 unidades de tiempo uno de otro.

σ2



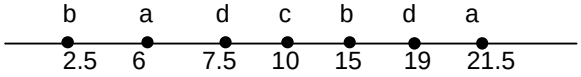
- ✓ esa porción de traza no satisface el escenario ya que entre ***c*** y ***a*** hay un evento ***e.***

σ3



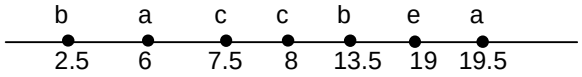
- ✓ satisface el escenario.

σ4



- ✓ esta porción de corrida no satisface el escenario puesto que entre los eventos ***c*** y ***a*** hay más de 10 unidades de tiempo.

σ5



- ✓ esta porción de traza no satisface el escenario. Después del evento ***c*** aparece un evento ***a*** pero antes que pasaran 5 unidades de tiempo, entre ***c*** y ***a*** aparece el evento ***e*** y además el evento ***a*** ocurre más de 10 unidades de tiempo desde ***c***.

Considerando ahora el siguiente escenario VTS

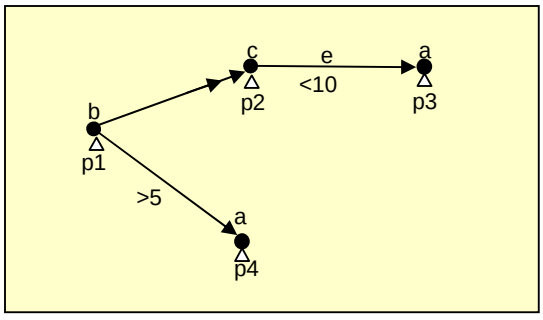
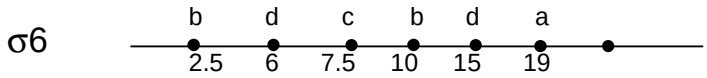


Figura 34: Ejemplo 2 de escenario VTS

Y se obtuviera la siguiente corrida:



- ✓ esta porción de corrida satisface el escenario puesto que el último evento ***a*** no está sujeto a la relación de asimetría pudiendo ser el mismo para los puntos P3 y P4.

2.5.7 Ejemplos de aplicaciones de VTS

Un ejemplo de uso de VTS puede encontrarse en el trabajo “Visual Timed Event Scenarios” [ABKO04]. Se trata de un proyecto para la Bolsa de Comercio de Buenos Aires. El grupo de trabajo en cuestión debía sugerir formas rigurosas de especificar y resolver requerimientos de tiempo-real de los agentes de bolsa encontrando oportunidades en las ofertas y recibiendo datos de compras actualizados.
Además uno de las formas propuestas recolecta las ofertas acumuladas en las máquinas de los agentes que están situadas en una LAN.

En el trabajo "A Scenario-Matching Approach to the Description and Model Checking of Real-Time Properties" [BKO05] se presenta otro caso de uso de VTS. En este caso se trata de un problema de un sistema de control que consiste en un componente central y dos sensores remotos. Estos sensores toman valores de ciertas variables del entorno y son almacenados en una memoria compartida. Cuando la central necesita estos valores emite una señal a los sensores. Al recibir la señal, los sensores toman el último valor desde la memoria compartida y la envían a la central. Tras recibir las dos porciones de datos, otro proceso puede utilizar esta información para tomar decisiones.

Capítulo 3: ¿SRS verificables?

En una especificación de requerimientos de software o SRS (Software Requirements Specification) cada requerimiento representa una funcionalidad que deberá cumplir el software a construir. Por lo tanto se necesita especificar estos requerimientos de manera que no puedan ser interpretados de distintas maneras, es decir sin ambigüedad.
Según el documento *IEEE Recommended Practice for Software Requirements Specifications* se define una SRS verificable o testeable sí y solo sí cada requisito especificado es verificable. Un requisito es verificable sí y solo sí existe un método finito de relación costo-beneficio positivo con el cuál una persona o una máquina puede verificar que el producto de software cumple dicho requisito.

En general cualquier requisito ambiguo no es verificable. Decir “El sistema deberá tener una buena interface” no es un requisito verificable porque no se puede saber exactamente lo que significa “buena interface” ya que es un concepto.

El lenguaje natural resulta ambiguo en algunos casos, como así también ciertos lenguajes gráficos, para especificar requerimientos. Esto se soluciona con la utilización de los lenguajes formales. Pero ¿esto nos asegura que estos requerimientos definidos formalmente sean verificables o testeables?.
Tenemos entonces un problema de chequeo del modelo o lo que se conoce como “model checking”. Se entiende como “model ckeching” a un conjunto de técnicas para el análisis automático del correcto funcionamiento de un sistema. Un chequeador del modelo (model checker) recibe como entrada una descripción formal del sistema a analizar y un conjunto de propiedades, generalmente expresadas como fórmulas de lógica temporal, que se espera sean cumplidas por el sistema. El chequeador del modelo confirma el cumplimiento de estas propiedades o la violación de las mismas (contraejemplo o error). [M01].

¿Cómo se verifican entonces, los modelos de los sistemas cuyas propiedades fueron especificadas con TE, MSC y VTS?.

3.1 Verificación

3.1.1 TimeEdit:

Como se mencionó anteriormente en el capítulo 2, esta herramienta gráfica se desarrolló para expresar requerimientos sin recurrir a formalismos. Para la verificación de los requerimientos definidos con esta herramienta se tradujeron los escenarios de TE a Büchi Automata (funcionalidad del TimeLine Editor) para luego utilizar Spin, un software verificador también desarrollado en los laboratorios de Bell (http://spinroot.com/spin/whatispin.html#A). Esta herramienta es de uso gratuito a partir de 1991 y utiliza como entrada para realizar la verificación un archivo escrito en PROMELA [M01]. PROMELA es un lenguaje con una

sintaxis parecida al C que permite expresar el modelo para luego ser simulado y verificado por SPIN.

3.1.2 MSC:

Existen trabajos que presentan traducciones de escenarios MSC a distintas herramientas para su verificación:

- ✓ En *Message Sequence Charts* de Genest, Muscholl y Peled [GMP03] se asocia un CFM (Communicating Finite-state Machine) a un MSC.
- ✓ En *Model Checking of Message Sequence Charts* de Rajeev Alur y Mihalis Yannakakis [AY99] se propone el uso de autómatas para verificar un MSC.
- ✓ En *Implementing and Verifying MSC Specifications Using PROMELA/xSPIN* de Stefan Leue y Peter B. Ladkin [LL97] se propone una implementación en PROMELA.
- ✓ En *Automatic Translation Of MSC Diagrams Into Petri Nets* de S. Kryvyy, L. Matvyeyeva y M. Lopatina [KML03] se sugiere una traducción de un MSC estándar a redes de Petri.
- ✓ En *An analysis of MSC* de Ladkin y Leue [LL92] se determina la expresividad de MSC con un autómata de Büchi.
- ✓ Oystein Haugen *presenta trabajos donde* compara gráficos de UML y MSC [H04] [H01].

Una de las formas posibles para verificar un MSC es a partir de un autómata. Teniendo un autómata con tiempos se puede utilizar Kronos o UPPAAL para verificar los modelos [BY03]. Tanto en Kronos (http://www-verimag.imag.fr/DIST-TOOLS/TEMPO/kronos/) como en UPPAAL (http://www.uppaal.org/) el sistema está modelado por un autómata con tiempos mientras que las propiedades están expresadas en TCTL (Timed CTL).

3.1.3 VTS:

En el sitio http://lafhis.dc.uba.ar/vts/ se puede obtener un traductor de VTS a autómatas con tiempos para luego utilizar Kronos o UPPAAL como se mencionó anteriormente. También se proponen otros verificadores para VTS como ZEUS (http://lafhis.dc.uba.ar/vintime/).
A partir de traducciones de VTS a redes de Petri, se puede optar por otros verificadores como TINA o TPN [MOYB08].
Recientemente se definió una traducción de VTS a SAL (http://sal.csl.sri.com/) que posibilita diversas formas de análisis [OY10].

3.2 Generador de corridas de VTS

Dado que la herramienta gráfica VTS cuenta con su parte formal en forma completa (TE no tenía descripción formal al comienzo de este trabajo y MSC cuenta con definiciones parciales), se puede tener en cuenta este aspecto para analizar las trazas que genera un escenario VTS.
En este trabajo de tesis se implementó un software que, recibiendo como entrada la información de la descripción formal de un escenario VTS (ver Figura 35), genera un listado de las corridas que cumplen con dicha descripción. Estas corridas muestran las secuencias de eventos de las trazas obtenidas con expresiones regulares.
Estas trazas están indicadas como expresiones regulares donde E (o ∑) representa el conjunto de todas las combinaciones posibles de los eventos.
Si bien el conjunto de trazas que cumplen con el escenario es infinito, al expresarlas de esta manera se las puede mostrar como una lista finita ya que al reducir todas las combinaciones de eventos (restando los prohibidos) en E, se puede expresar la secuencia de eventos que

se debe observar según el orden de los puntos por los que se pasan sin detener la atención en todas las posibilidades de eventos que pueden suceder entre estos puntos.

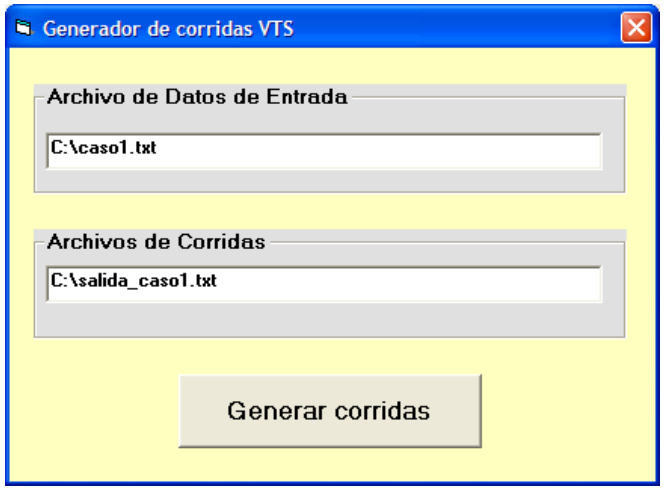


Figura 35: Interface del generador de corridas

O sea, que en vez de tomar un autómata como representación del sistema y las propiedades expresadas ya sea en TCTL o CTL, se propone un listado de trazas que cumplen con el escenario VTS que luego podrían ser comparadas con las trazas obtenidas con el modelo para analizar si se cumplen o no con las propiedades o requerimientos especificados. No es parte de este trabajo esta comparación.

En caso de tener un escenario VTS con antecedentes y consecuentes (gráfico condicional), se traduce a escenarios simples y luego se pueden generar las corridas correspondientes a dichos escenarios.

Este generador está desarrollado en Visual Basic y tanto el archivo de entrada como el de salida son simplemente texto.

Recordemos que en VTS un escenario es una tupla $\langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$ donde:

Σ es un conjunto finito de eventos,
P es un conjunto finito de puntos,
$\ell : P \to 2^{\Sigma}$ es la función donde a cada punto de P le corresponde un subconjunto de eventos de Σ, puede ser $\varnothing$,
$\not\equiv \subseteq P \times P$ es una relación asimétrica entre los puntos de desigualdad,
$< \subseteq \left(P \overset{+}{\bigcup} \{0\} \times P \overset{+}{\bigcup} \{\infty\} \right) \setminus \{(0,\infty)\}$ es una relación de precedencia entre los puntos, 0 y ∞ representan el comienzo y el final de la ejecución respectivamente,
$<_F \subseteq P \times P$ es una relación que representa el primer punto en un conjunto,
$<_L \subseteq P \times P$ es una relación que representa el último punto en ocurrir,
$\gamma : (\not\equiv \bigcup <) \to 2^{\Sigma}$ es la función que asigna a cada par de puntos un conjunto de eventos prohibidos entre ellos, y
$\delta : (\not\equiv \bigcup <) \to \Phi$ es la función que asigna a cada par de puntos una restricción de tiempo entre dos puntos, donde Φ es el conjunto de todas las restricciones de tiempo.

El archivo de entrada tiene la siguiente estructura:

- ✓ Eventos: un renglón por cada evento. Comienzan con la letra **e** seguida de coma y el nombre del evento.
- ✓ Línea de separación con el texto x,x.
- ✓ Relación de precedencia entre los puntos: un renglón por cada relación. Comienza con la letra **d** seguida de una coma y los puntos también separados por coma. (los puntos son nombrados simplemente con números y **Start** indica inicio y **end** indica final)
- ✓ Función de asignación de eventos a puntos: un renglón por cada evento en cada punto. Comienza con la letra **r** seguida por una coma, el número que representa al punto, nuevamente coma y el evento asociado.
- ✓ Relación que representa el primer punto de un conjunto: un renglón por cada punto este conjunto. Comienza con la letra **f**, luego una coma y los puntos de esta de también separados por coma.
- ✓ Relación que representa el último punto de un conjunto: un renglón por cada punto este conjunto. Comienza con la letra **l**, luego una coma y los puntos de esta relación, relación, también separados por coma.
- ✓ Relación asimétrica: un renglón por cada par de puntos de la relación. Comienza con la letra **a**, luego una coma seguida por los puntos de esta relación también separados por coma.
- ✓ Línea de separación con el texto x,x,x.
- ✓ Eventos prohibidos: un renglón por cada evento prohibido asociado a un par de puntos. Comienza con la letra **p** seguida de coma, luego los puntos del par de la restricción y el evento prohibido también separados por coma.

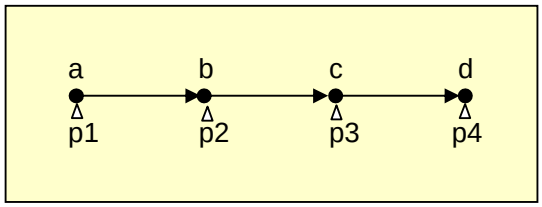
En esta primera etapa del desarrollo del generador se decidió tomar las restricciones de las unidades de tiempo como un conjunto ∅ para todos los casos debido a su complejidad para medir dichos tiempos.

El archivo de salida presenta al comienzo el siguiente texto explicativo:
"Dado el siguiente escenario VTS: S={E, P, l, d, <, <f, <l, ep, rt} donde:"
"E: conjunto finito de eventos"
"P: conjunto finito de puntos"
"l: función donde a cada punto de P le corresponde un subconjunto de eventos de E"
"d: relación asimétrica entre los puntos de desigualdad"
"<: relación de precedencia entre los puntos"
"<f: relación que representa el primer punto de un conjunto"
"<l: relación que representa el último punto de un conjunto"
"ep: función que asigna a cada par de puntos un conjunto de eventos prohibidos entre ellos"
"rt: función que asigna a cada par de puntos una restricción de tiempo entre ellos"

Luego se muestra los valores que asumen los distintos conjuntos, funciones y/o relaciones. A continuación se listan las trazas obtenidas que satisfacen el escenario definido mostrando cómo debe ser la sucesión de los puntos para cada corrida y cómo se traduce esta secuencia con los eventos asociados a cada punto.

Ejemplo del Generador de Corridas de un escenario VTS

Teniendo en cuenta el siguiente escenario VTS



El archivo de entrada es el que se muestra en la Figura 36 y el archivo de salida el que se muestra en la Figura 37.

Figura 36: Archivo de entrada

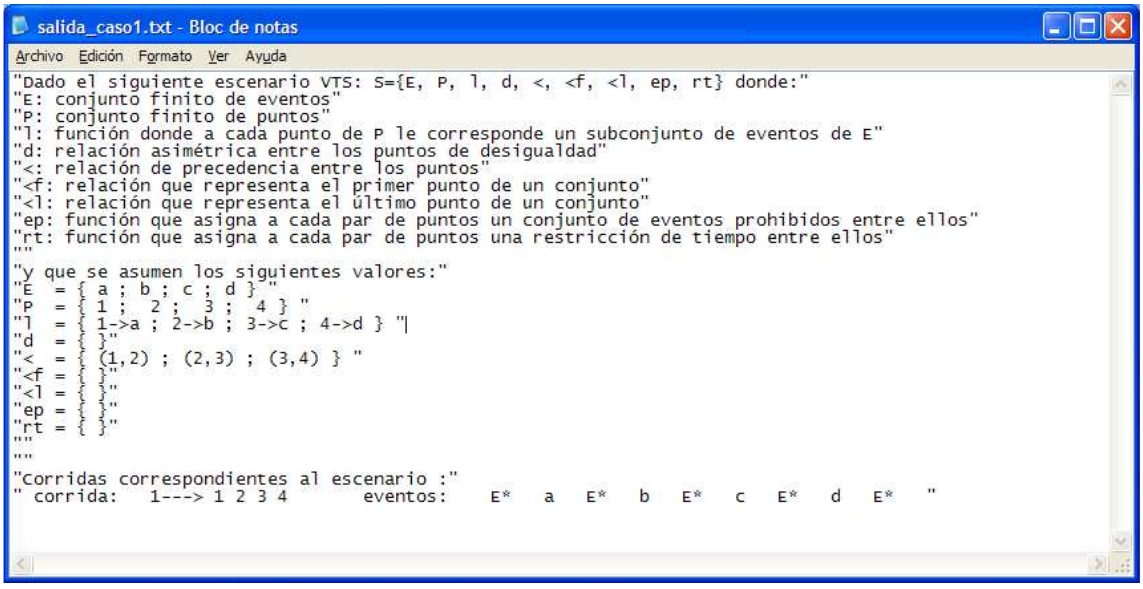


Figura 37: Archivo de salida

En el Anexo A se presentan distintos casos de escenarios VTS con sus respectivos archivos de entrada y salida.

Capítulo 4: Comparación TE y VTS

Se analizaron los aspectos gráfico, textual y formal de estos dos formalismos y se establecieron las siguientes relaciones para pasar de uno a otro.

4.1 Pasando de un escenario TE a uno VTS

4.1.1 Relación entre gráficos

Para pasar de un gráfico TE a uno VTS se tendrá en cuenta las siguientes observaciones:

- ✓ Todos los gráficos de TE comienzan desde START, por lo tanto el escenario VTS equivalente debe comenzar con un punto de inicio.
- ✓ Se asume cada marca de tipo esperado con un evento en el gráfico TE como un punto con un evento en el escenario VTS.
- ✓ La relación de precedencia de los eventos del escenario VTS equivalente está dada por el orden en que se suceden las marcas y las restricciones en la línea de tiempo del gráfico TE.
- ✓ Como el evento asociado a una marca de tipo esperado de un gráfico TE es el primero de esos eventos que sucede desde la marca anterior, se toma dicho evento como prohibido entre los puntos del escenario VTS que representan la marca anterior y la marca en cuestión.

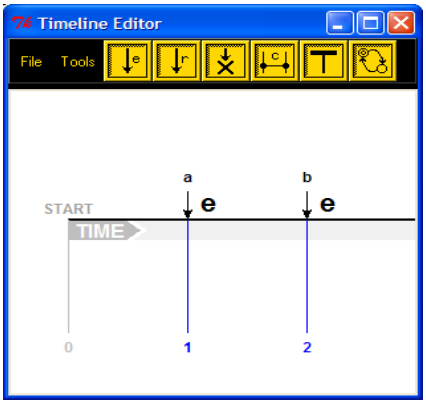
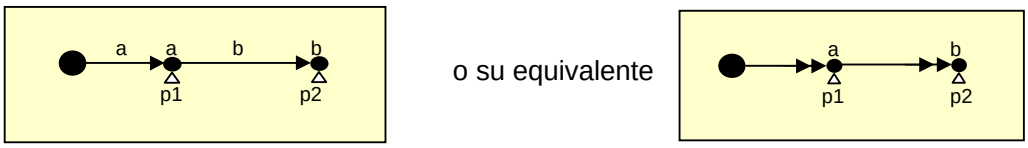


Figura 38: Ejemplo simple de escenario TE

El gráfico de la Figura 38 es el caso más simple que se puede tener en un escenario TE. Se toma cada marca e (esperado) y su evento como un punto en VTS con dicho evento asociado. En la marca 1 se observa el evento ***a***. Este es el primer evento ***a*** que sucede desde el comienzo. En la segunda marca del ejemplo se observa un evento ***b***, este es el primer evento ***b*** después del evento ***a*** por lo que el escenario VTS equivalente sería:



✓ Cuando se muestran más de un evento en una marca en un gráfico TE (eventos simultáneos), como se observa en las marcas 1 y 3 de la Figura 39, serán dos puntos distintos en el escenarios VTS.
Si el evento simultáneo al evento de la marca es la negación de uno de los eventos definidos (ver marca 1 de la Figura 39) entonces éste será un evento prohibido entre dos puntos, uno representando al evento de la marca y el otro sin evento asociado.
Si el evento simultáneo al evento de la marca no es la negación de uno de los eventos definidos (ver marca 3 de la Figura 39) entonces se tendrá un punto representando el evento de la marca con dicho evento asociado y el otro punto estará asociado al evento simultáneo.
En ambos casos, por ser eventos simultáneos, ambos puntos están unidos por una restricción temporal igual a cero.

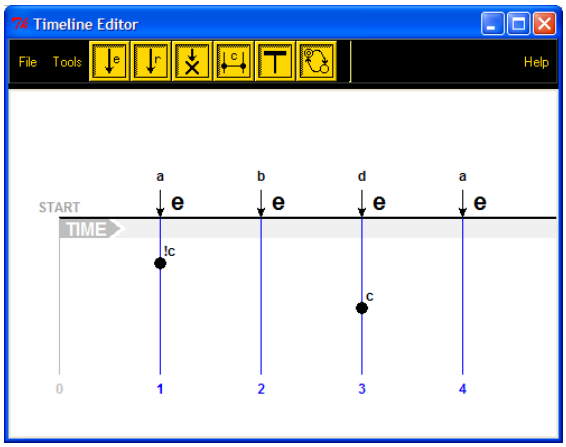
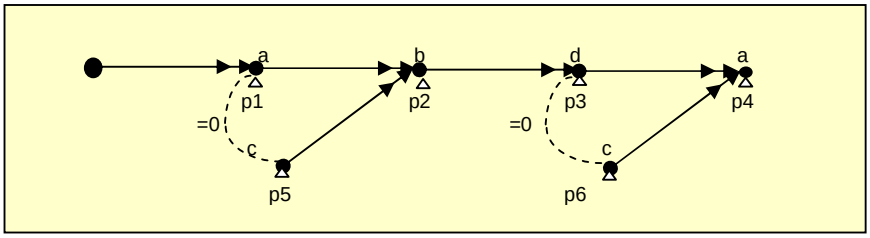


Figura 39: Ejemplo de eventos simultáneos

El escenario VTS que representa a este escenario TE es el siguiente:



✓ Cuando se muestren restricciones en el gráfico TE que incluyen las marcas de inicio y final (como se observa entre las marcas 2 y 3 de la Figura 40) serán en el escenario VTS eventos prohibidos entre los puntos que representan dichas marcas.

- ✓ Cuando se muestren restricciones en el gráfico TE que incluyen la marca de inicio pero no la de final (como se observa entre las marcas 4 y 5 de la Figura 40), se agregará en el escenario VTS un punto entre las marcas de inicio y final sin evento asociado. Se agrega un arco entre el punto que representa la marca de inicio y este punto. El evento asociado a la marca final y el evento asociado a la restricción serán eventos prohibidos en este arco. Se agrega un arco desde este nuevo punto y el que representa la marca final de la restricción con el evento de esta marca como evento prohibido (doble flecha) y se agrega una restricción temporal >0.
- ✓ Cuando se muestren restricciones en el gráfico TE que incluyen la marca final pero no la de inicio (como se observa entre las marcas 5 y 6 de la Figura 40), se agregará en el escenario VTS un punto entre las marcas de inicio y final sin evento asociado. Se agrega un arco entre el punto que representa la marca de inicio y este punto. El evento asociado a la marca final será el evento prohibido en este arco y se agrega una restricción temporal >0.
  Se agrega un arco desde este nuevo punto y el que representa la marca final de la restricción con el evento de esta marca y el de la restricción como eventos prohibidos.
- ✓ Cuando se muestre restricciones en el gráfico TE que no incluyen la marca final ni la de inicio (como se observa entre las marcas 6 y 7 de la Figura 40), serán en el escenario VTS dos puntos intermedios sin eventos asociados. Se agrega un arco entre el punto que representa la marca de inicio y el primero de estos puntos. El evento asociado a la marca final será el evento prohibido en este arco con una restricción temporal >0. Se agrega un arco entre los dos punto agregados y asociando como eventos prohibidos al evento de la marca final y al de la restricción. Se agrega un arco desde el segundo punto y el punto que representa la marca final de la restricción con el evento de esta marca como evento prohibido y una restricción temporal >0.

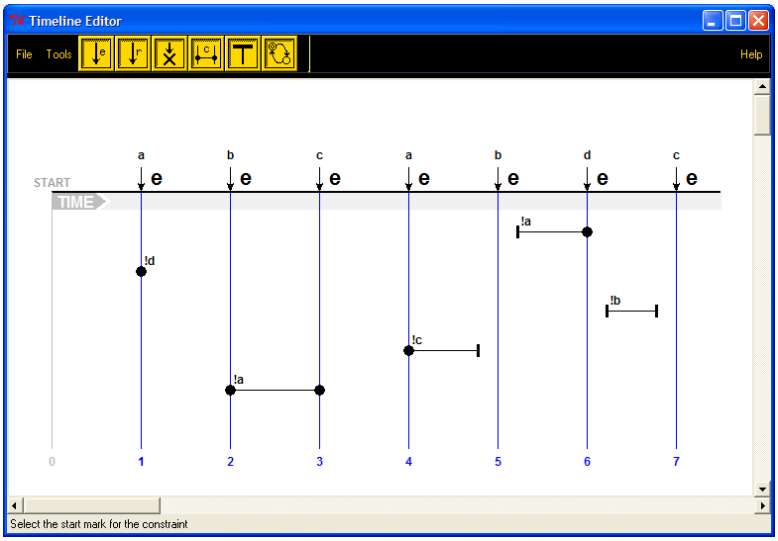
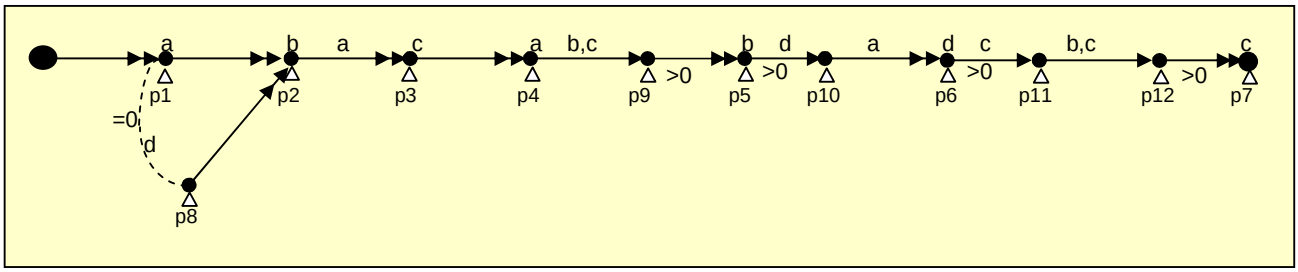


Figura 40: Ejemplo de escenario TE con eventos simultáneos y prohibidos

El siguiente escenario VTS representa el escenario TE de la Figura 40 donde:

- Los puntos p1 al p7 se relacionan con las respectivas marcas.
- El punto p8 representa el evento simultáneo de la marca 1.
- El punto p9 corresponde al punto que se genera con el extremo final de la restricción de tipo TF anterior a la marca 5.

- El punto p10 corresponde al punto que se genera con el extremo inicial de la restricción de tipo FT anterior a la marca 6.
- Los puntos p11 y p12 corresponden a los puntos que se generan con el extremo inicial y final respectivamente de la restricción de tipo FF anterior a la marca 7.



✓ Los eventos de las marcas de tipo requerido de un gráfico TE (como se observa en la marca 3 de la Figura 41) se mostrarán como eventos prohibidos en el escenario VTS desde el punto que representa a la marca anterior hasta el final. Entonces este escenario debería ser FALSO para que cumpliera con la aparición del evento de la marca de tipo requerido.

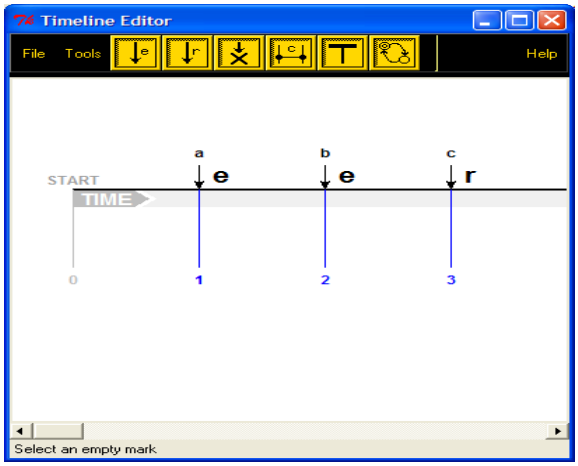
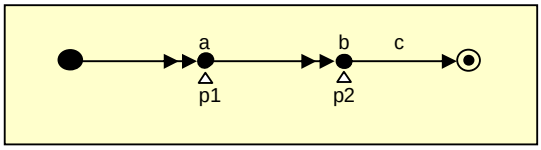


Figura 41: Escenario TE con marca tipo r

El siguiente escenario VTS equivale al escenario TE de la Figura 41.



✓ En el ejemplo de la Figura 42 se muestran dos marcas r con una marca e (esperado) en el medio.

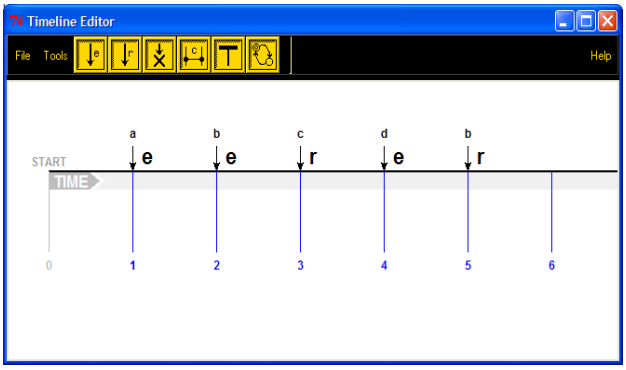
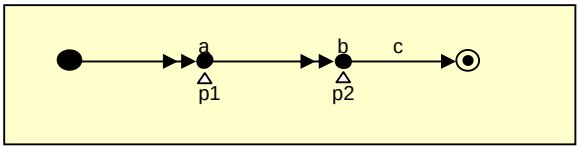


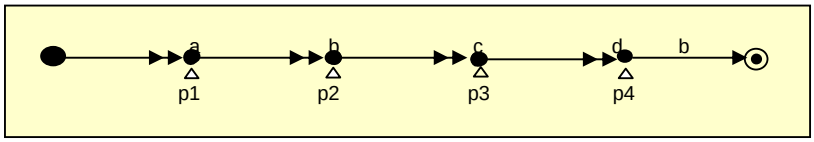
Figura 42: Escenario TE con varias marcas r

Para poder representar este escenario TE en VTS se observa que debe ocurrir un evento ***c*** luego de los eventos ***a*** y ***b***, y después se debe observar el evento requerido en la marca 5. Se obtienen los siguientes escenarios VTS debiéndose observarse el primero para luego darse el segundo. Los escenarios VTS deberían ser los siguientes:

Primer escenario: este debería ser FALSO para cumplir con el primer evento de tipo requerido de la marca 3.



Segundo escenario: este también debería ser FALSO para cumplir con el segundo evento de tipo requerido la marca 5.



✓ En el caso de presentarse dos marcas r seguidos como en la Figura 43, se puede expresar este escenario en VTS de las siguientes formas:

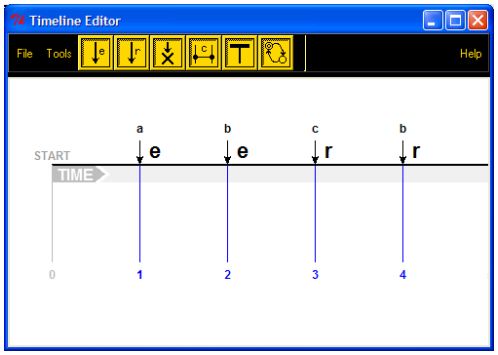
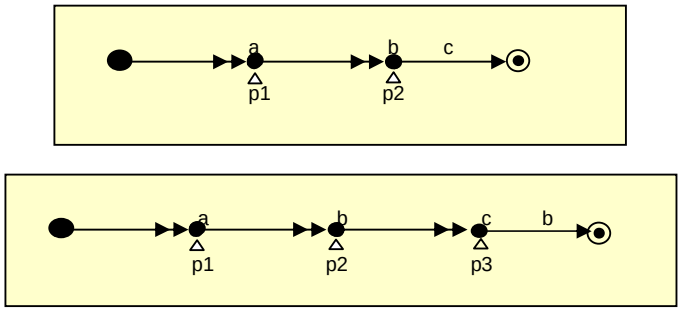


Figura 43: Escenario TE con marcas r consecutivas

Deberá observarse el primer escenario como FALSO para luego evaluar el segundo también como FALSO.



✓ Los eventos asociados a una marca de falla de un gráfico TE se mostrarán en un escenario VTS como eventos prohibidos que hacen finalizar la corrida.

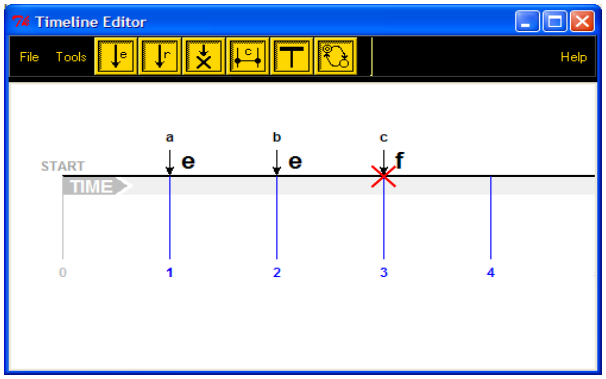
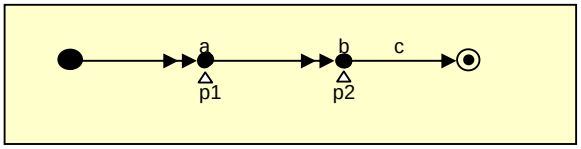


Figura 44: Escenario TE con marca de falla

En la Figura 44 se observa en la marca 3 de falla el evento ***c***. Este evento asociado a la marca de falla o error se representa en un escenario VTS como un evento prohibido desde la marca anterior hasta el final. Entonces este escenario debería ser VERDADERO para que cumpla con el de TE. Si es falso no cumple.



✓ En el caso de tener un escenario TE que tuviera una marca f y eventos prohibidos entre esta marca y la anterior como lo muestra la Figura 45 se incorporará al escenario VTS un punto sin evento asociado. Se unirá con una flecha el punto que representa la marca anterior a la de falla y este punto agregado teniendo como eventos prohibidos el evento indicado en la restricción (entre la marca anterior y la marca de falla) y el evento asociado a la marca de falla. También se le sumará una restricción temporal >0. Este escenario VTS deberá ser VERDADERO para cumplir con el escenario TE.

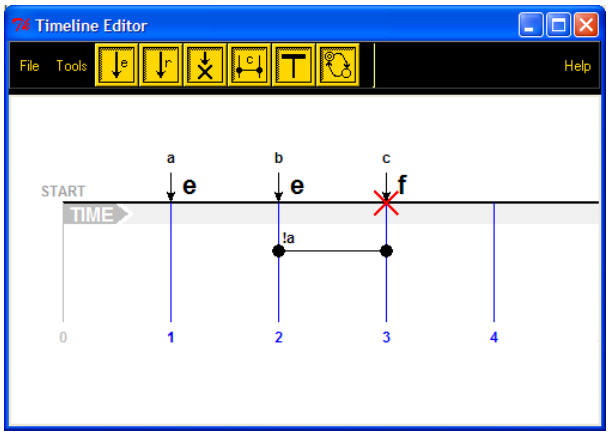
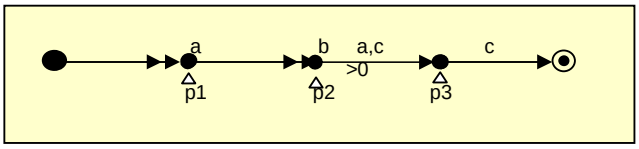


Figura 45: Escenario TE con una marca f y evento prohibido

El escenario VTS que representa el caso de la Figura 45 es el siguiente:



✓ También podría darse un escenario TE donde se presenta una marca f seguida de una marca r como en la Figura 46:

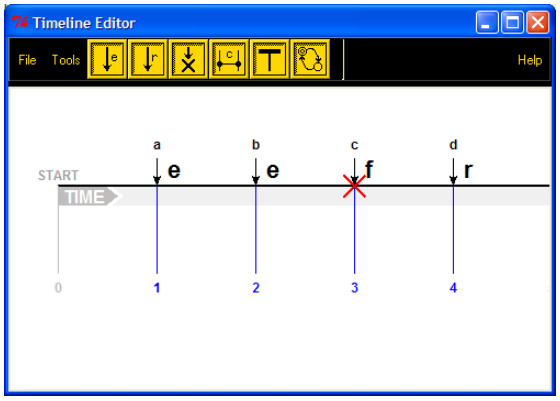
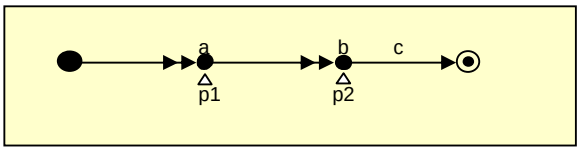
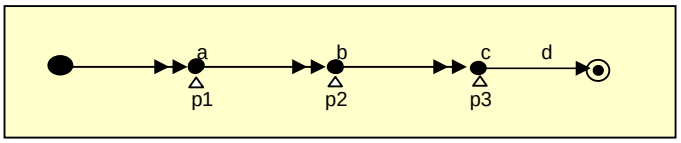


Figura 46: Escenario TE con una marca f seguida de una marca r

En el caso del escenario de la Figura 46 se representa con dos escenarios VTS debiendo observarse el primero como VERDADERO y luego el segundo como FALSO. Los escenarios VTS son como los siguientes:





4.1.2 Relación textual

Debido a que el lenguaje TE no cuenta con una definición textual, no se puede establecer esta relación.

4.1.3 Relación formal

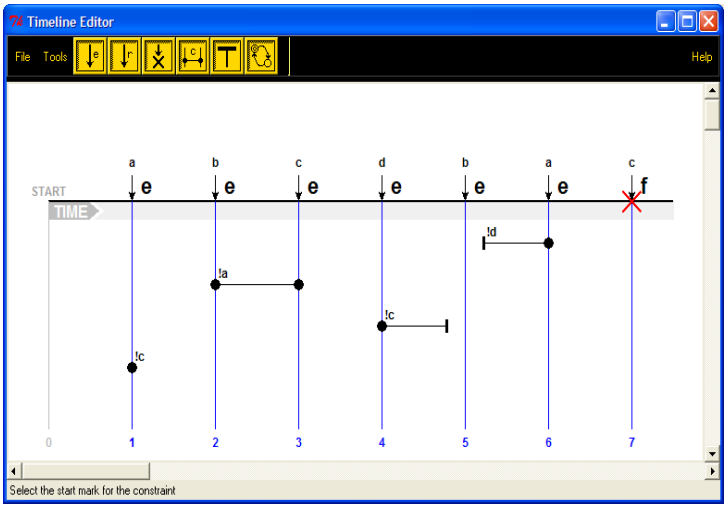
Cuando pasamos formalmente de un escenario TE a uno VTS, el escenario resultante tendrá $<_F$ y $<_L$ iguales a vacío ya que no se observan estas facilidades de VTS en la herramienta TE.

Por otro lado, las marcas de tipo f y las marcas de tipo r se perderán entre los eventos prohibidos, que causan la finalización de la corrida aunque en un caso el escenario deberá ser falso (en el caso de la marca r) y en el otro verdadero (en el caso de la marca f).

De TE a VTS	
El escenario $S_{VTS} = \langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$ que resulta a partir de un escenario $S_{TE} = \langle E; M; a; t; S; \alpha \rangle$ se obtendrá de la siguiente manera:	
Σ	• Se relaciona con E . Son el mismo conjunto.
P	• Se relacionan con el conjunto de las marca, un punto por cada marca de tipo ***e***. • Se incorpora un punto cuando en el escenario TE se observen eventos simultáneos asociado a una marca (tipo TT con misma marca de inicio y final). • Se incorpora un punto cuando en el escenario TE se tiene una restricción que no incluya la marca de inicio (tipo FT) o marca de fin (tipo TF), y dos puntos en caso de restricciones que no incluyen la marca de inicio ni la de final (tipo FF).
ℓ	• Se relaciona con a coincidiendo el tipo de relación: asigna un evento a una marca ***e***. En VTS será la asignación de eventos a puntos. • Los puntos que surgen de las restricciones de tipo TF, FT y FF no tienen eventos asociados. • Los puntos que surgen de eventos simultáneos asociados a una marca (tipo TT con mismo inicio y final) no tendrán eventos asociados si el evento simultáneo es la negación de uno de los eventos definidos. • Los puntos que surgen de eventos simultáneos asociados a una marca (tipo TT con mismo inicio y final) tendrán a dicho evento como asociado si el evento simultáneo es la negación de uno de los eventos definidos.
$\not\equiv$	• Se agregan pares formados por el punto que surge al observar un evento simultáneo asociado a una marca tipo TT con mismo inicio y final y el punto que representa dicha marca.
$<$	• No tiene su similar en TE puesto que, por definición, las marcas en TE con sus eventos forman un conjunto totalmente ordenado. Por lo tanto se obtendrán los pares ordenados de la observación de las marcas y de las restricciones en éstas o entre éstas en la línea de tiempo. • Las restricciones que afectan únicamente a una marca (tipo TT con mismo inicio y final) agregan un arco en el escenario VTS desde ese punto a la marca siguiente. • Se incorporan arcos desde los puntos agregados por las restricciones que no incluyan marcas de inicio y/o marcas de fin (tipo TF, FT y FF) a las marcas anteriores o siguientes según corresponda al extremo FALSO.
$<_F$	No tiene sus similares en TE. Por lo tanto resulta $\varnothing$.
$<_L$	No tiene sus similares en TE. Por lo tanto resulta $\varnothing$.
γ	Se relaciona con a y t de la siguiente manera: • El evento asociado a una marca de tipo esperado del gráfico TE es el primero de esos eventos que sucede desde la marca anterior y se

	toma dicho evento como prohibido entre los puntos del escenario VTS que representan la marca anterior y la marca en cuestión. • Los eventos asociados a las marcas r también se tomarán como eventos prohibidos desde la marca anterior a esta marca r y un punto de final. • Los eventos asociados a las marcas f también se tomarán como eventos prohibidos desde la marca anterior a esta marca r y un punto de final. También se relacionan con α de la siguiente manera: • Los eventos de las restricciones entre marcas de tipo TT con distinto inicio y final, serán los prohibidos en los segmentos del escenario VTS que representan a las marcas en cuestión. • Si el evento simultáneo asociado a una marca tipo TT con mismo inicio y final es la negación de uno de los eventos definidos en P entonces este será un evento prohibido entre el punto que representa la marca y el evento simultáneo. • Los eventos de las restricciones de tipo TF, FT y FF serán los prohibidos en los segmentos creados en el escenario VTS que representan los lapsos entre la marca inicio y el comienzo de la restricción y/o entre el final de la restricción y la marca final.
δ	• Cuando encontramos eventos prohibidos asociados a una marca únicamente (restricciones del tipo TT con misma marca de inicio y final) se observará una restricción temporal = 0 entre el punto que representa esa marca y dicha restricción. • Cuando tenemos restricciones de tipo TF, FT o FF se observará una condición temporal > 0 en los segmentos que representan el lapso entre la marca inicio y el comienzo de la restricción y/o entre el final de la restricción y la marca final.

Supongamos el siguiente escenario TE expresado gráfica y formalmente:



$S = \langle E; M; a; t; S; \alpha \rangle$ donde:

E = {a; b; c; d}

M = {0;1; 2; 3; 4; 5; 6; 7}

a = {0 →start; 1 →a; 2 →b; 3 →c; 4 → d; 5 → b; 6 → a; 7 → c}

t = {1 →e; 2 →e; 3 →e; 4 →e; 5 →e; 6 →e; 7 →f }

TT = {⟨1,1⟩; ⟨2,3⟩}

TF = {⟨4,5⟩}

FT = {⟨5,6⟩}

FF = {}

S = {⟨1,1⟩; ⟨2,3⟩; ⟨4,5⟩; ⟨5,6⟩}

α = {⟨1,1⟩→{!c}; ⟨2,3⟩→{!a}; ⟨4,5⟩ →{!c}; ⟨5,6⟩→{!d}}

Figura 47: Ejemplo de escenario TE

Se determinan los puntos a considerar en el escenario VTS:

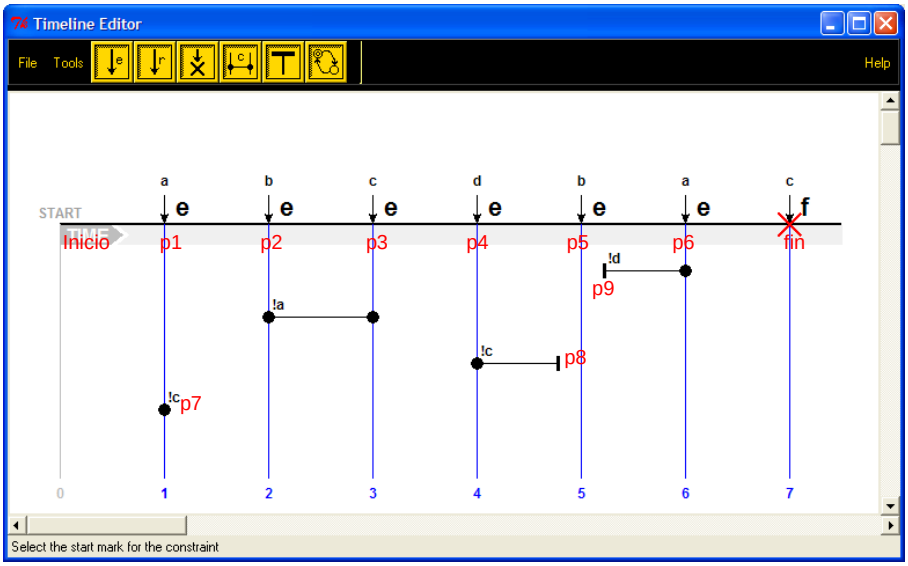


Figura 48: Asignación de puntos

Aplicando lo dicho anteriormente y teniendo en cuenta la asignación de puntos como se observa en la Figura 48, se obtendría el siguiente escenario VTS $S_0 = \langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$ de la siguiente manera:

$S = \langle E; M; a; t; S; \alpha \rangle$ donde:

E = {a; b; c; d}

M = { 0;1; 2; 3; 4; 5; 6; 7}

a = {0 →start; 1 →a; 2 →b; 3 →c; 4 → d; 5 → b; 6 → a; 7 → c}

t = {1 →e; 2 →e; 3 →e; 4 →e; 5 →e; 6 →e; 7 →f}

TT = {⟨1,1⟩; ⟨2,3⟩ }

TF = {⟨4,5⟩}

FT = {⟨5,6⟩}

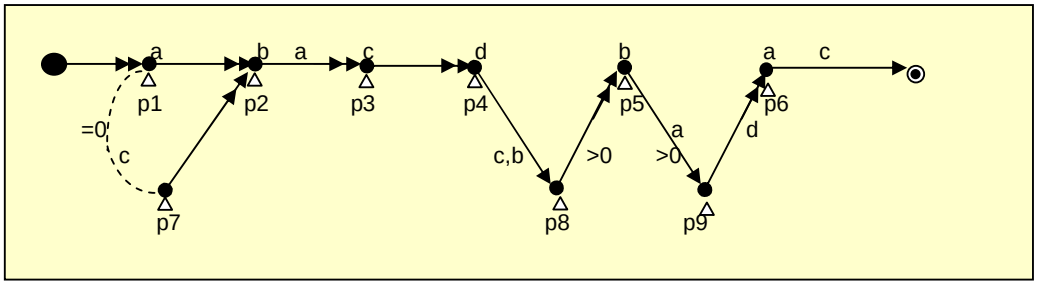
FF = {}

S = {⟨1,1⟩; ⟨2,3⟩; ⟨4,5⟩; ⟨5,6⟩ }

α = {⟨1,1⟩→{!c}; ⟨2,3⟩→{!a}; ⟨4,5⟩ →{!c}; ⟨5,6⟩→{!d}}

$S_0 = \langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$
Σ = { a; b; c; d }
P = { p1; p2; p3; p4; p5; p6; p7; p8, p9 }
ℓ = { p1→{a}; p2→{b}; p3→{c}; p4→{d}; p5→{b}; p6→{a}}
$\not\equiv$ = { (p1,p7) }
$<_L = <_F = \varnothing$
$<$ = { (p1,p2); (p2,p3); (p3,p4); (p7,p2); (p4,p8); (p8,p5), (p5,p9); (p9,p6); (p6,∞) }
γ ={ (p1,p7)→{c}; (p1,p2)→{b}; (p2,p3)→{a, c}; (p3,p4)→{d}; (p4,p8)→{c, b}; (p8,p5)→{b}; (p5,p9) →{a}; (p9,p6)→{d, a}; (p6,∞) →{ c}}
δ ={ (p1,p7)→[0,0]; (p8,p5)→ (0, ∞); (p5,p9)→ (0, ∞) }

El escenario VTS que representa este escenario TE es el siguiente:



4.1.4 Ventajas y desventajas

Después de realizar distintos ejemplos y casos posibles, se observa que siempre se puede pasar, con mayor dificultad en algunos casos, de un gráfico de TE a escenarios de VTS.
Se han observado las siguientes desventajas en los resultados de VTS con respecto a TE:

- ✓ Un gráfico de TE con distintos restricciones de eventos prohibidos (eventos asociados a segmentos que pueden incluir o no las marcas iniciales y finales) es más simple de interpretar observando el gráfico TE que el escenario que surge en VTS.
- ✓ Un gráfico de TE con varias marcas de tipo requerido y/o de falla equivale a varios escenarios de VTS.
- ✓ En ciertos casos el escenario VTS que resulta con una marca de tipo requerido o con una marca de falla es el mismo para los dos casos y es necesario aclarar cuando debe ser FALSO o VERDADERO para cumplir con la propiedad (ver Figura 41 y Figura 44).

Por lo enumerado anteriormente se puede concluir que:

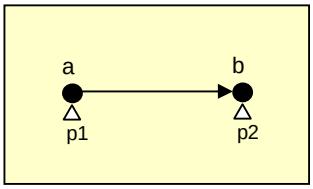
- ✓ Cuando los eventos se suceden ordenadamente en el tiempo es más simple de interpretar el gráfico que surge en TE que el de VTS.
- ✓ En TE es más simple observar los estados que se satisfacen o cuáles no entre marcas.

4.2 Pasando de un escenario VTS a uno TE

4.2.1 Relación entre gráficos

Para pasar de un gráfico VTS a uno TE se tendrá en cuenta las siguientes observaciones:

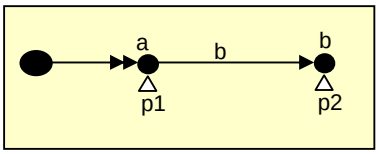
- ✓ En un escenario VTS no siempre se debe comenzar con un punto de inicio pero en un gráfico TE siempre comienza con una marca 0 o Start.
- ✓ En VTS indicamos la ocurrencia de un evento en un punto pero eso no significa que sea el primero de esos eventos que ocurre. En los escenarios de TE se indica la secuencia en que se sucederán los eventos en el tiempo asumiendo que el evento asociado a las marcas suceden por primera vez desde la marca anterior.
  Dado el siguiente escenario VTS:



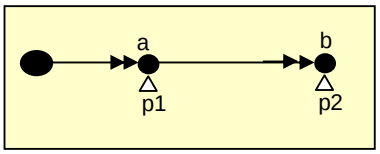
Este escenario VTS es muy simple pero no se puede representar en TE. Este escenario VTS indica que ocurre un evento ***a*** y luego otro ***b*** pudiendo ocurrir cualquier otro evento (incluyendo eventos ***a*** y ***b***) entre estos dos.

- ✓ Se asume que un punto con un evento en un escenario VTS que sucede por primera vez desde el punto anterior, se refleja en un gráfico TE como un evento asociado a una marca de tipo esperado.

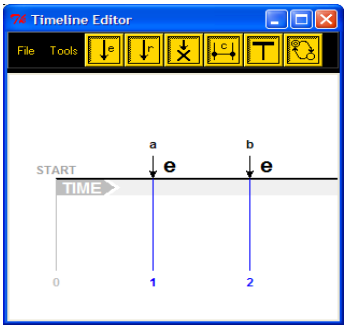
Teniendo este escenario VTS



o su equivalente



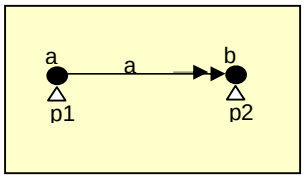
su escenario TE equivalente es el siguiente:



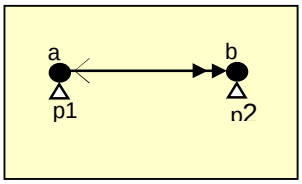
En los siguientes ejemplos de escenarios VTS se toma una porción de traza sin incluir el punto de inicio. Al generar el gráfico TE equivalente se observará siempre la marca o (o start).

- ✓ En un escenario VTS donde se muestran como prohibidos el punto de origen del arco y el de destino, si tiene como equivalente en TE una restricción de tipo TT tomando como marcas de inicio y final de la restricción a las marcas del gráfico TE que representan los puntos del arco del escenario VTS.

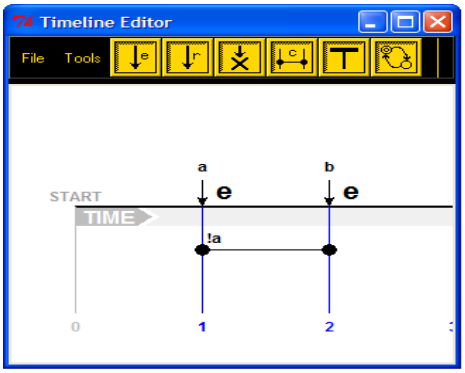
Si el escenario VTS es el siguiente:



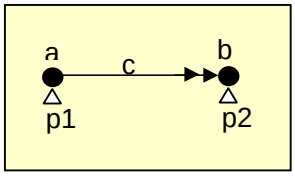
o su equivalente



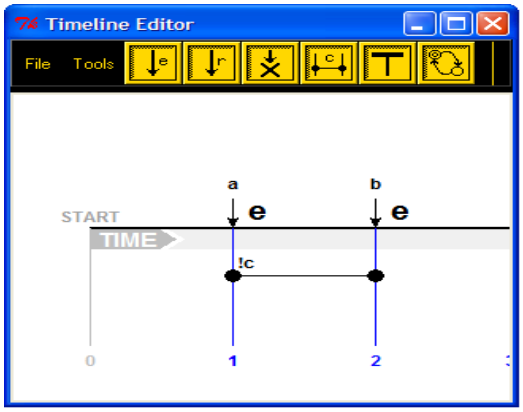
En TE se vería de la siguiente manera:



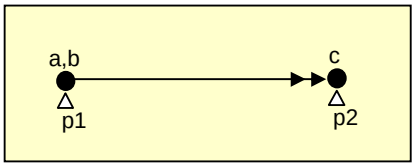
- ✓ Si en un escenario VTS se muestran eventos prohibidos entre dos eventos, estos son posibles de mostrar en un escenario TE como restricciones de tipo TT entre las marcas que se crean a partir de los puntos del arco sobre el que se encuentra el evento prohibido.



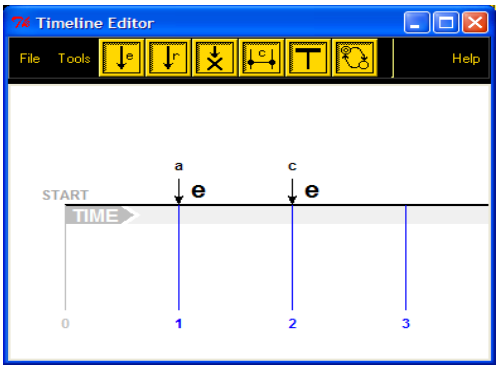
En TE se vería de la siguiente manera:



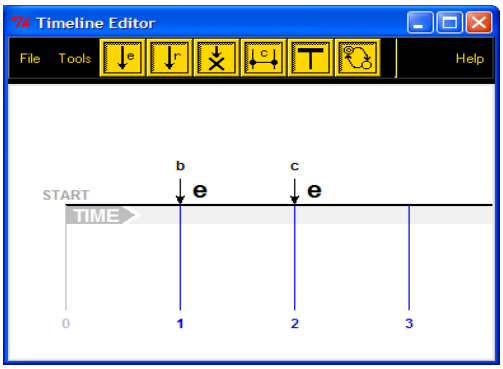
- ✓ En VTS podemos tener más de un evento (conjunto) asociado a un punto. Esto indica que ocurrirá uno de esos eventos asociados a ese punto. En TE se asocia un determinado evento a una marca por lo que se necesitan más de un gráfico TE para representar el siguiente caso.



Este escenario VTS no se puede representar en un único gráfico de TE ya que en el punto ***p1*** puede darse el evento ***a*** o el evento ***b***. Se necesitan dos escenarios TE, uno para cada uno de los eventos posibles en la primera marca:



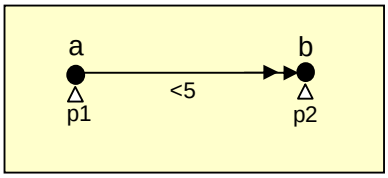
En este gráfico se representó la posibilidad que ocurra el evento ***a*** en el punto ***p1*** antes que el evento ***c*** en el punto ***p2.***



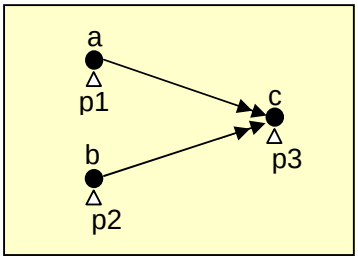
En este otro se muestra la posible ocurrencia del evento ***b*** en el punto ***p1*** antes que el evento ***c*** en el punto ***p2.***

- ✓ En VTS podemos tener restricciones temporarias dadas por la relación ***δ*** las que en TE no pueden ser graficadas puesto que no hay medición de tiempo; simplemente los eventos se suceden en una línea temporal sin ninguna indicación de cuántas unidades de tiempo transcurren entre las marcas.

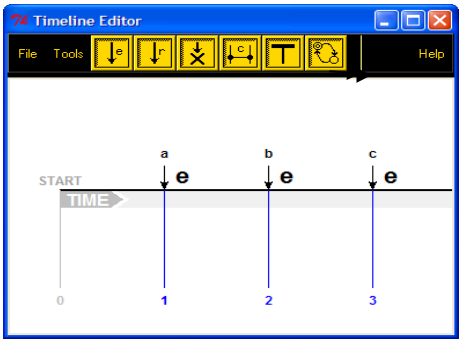
  En el siguiente escenario VTS se debe observar una restricción temporal entre los eventos ***a*** y ***b***. Esto no puede representarse en un gráfico TE.



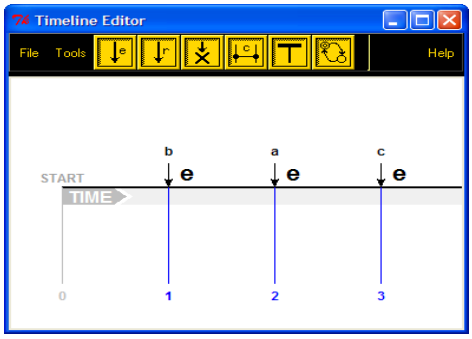
- ✓ Una de las facilidades de VTS es la posibilidad de saber que se dan ciertos eventos antes que otro pero no necesariamente debe saberse en qué orden. En el siguiente escenario VTS se sabe que deben darse los eventos ***a*** y ***b*** antes de ***c*** pero no importa en qué orden.



  Para poder mostrar esto en TE se necesitan dos gráficos, uno en que suceda el evento ***a*** antes que el evento ***b*** y luego el evento ***c***, y otro en que se observe primero el evento ***b*** antes que el evento ***a*** y luego el evento ***c***.



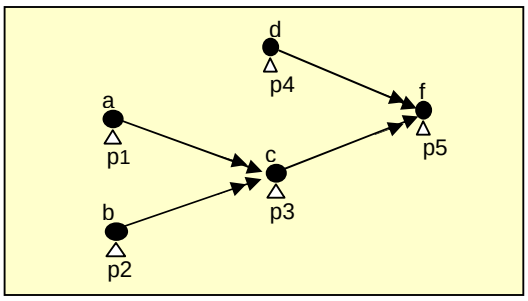
En este gráfico vemos la opción que ocurra el evento ***a*** antes que ***b*** y por último el evento ***c***.



En este caso ocurre el evento ***b*** antes que el ***a***. Luego ocurre el evento ***c.***

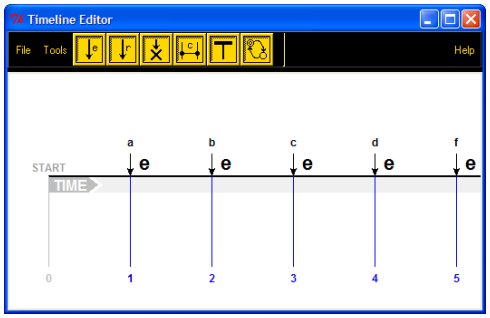
Si se tiene un escenario más complejo en cuanto a la secuencia de eventos sin un orden estricto, el escenario VTS es muy simple e intuitivo mientras que en TE se necesitan varios gráficos para mostrar lo mismo.

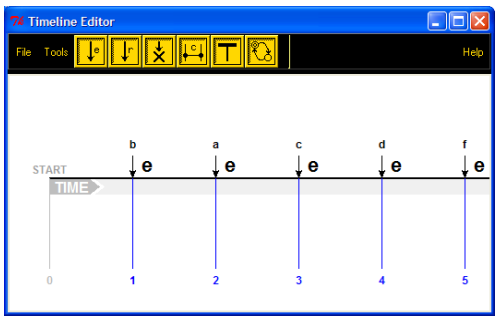
Por ejemplo, si se observa este escenario VTS



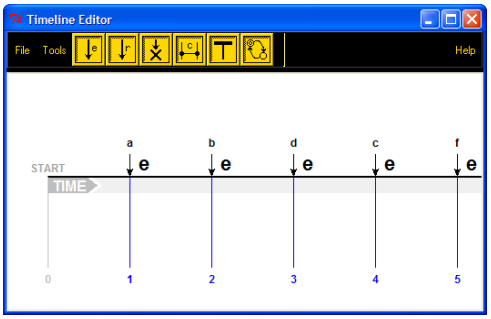
se necesitarán los siguientes ocho gráficos TE para representar lo mismo.

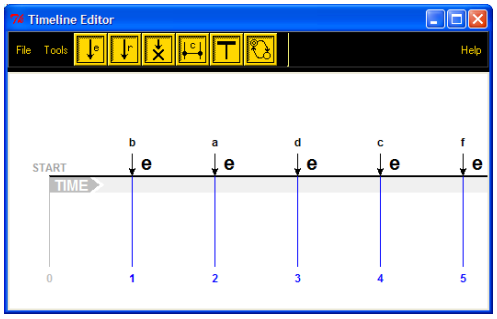
- En estos gráficos ocurre el evento ***a*** o ***b*** antes que el ***c***, ***d*** y ***f***.



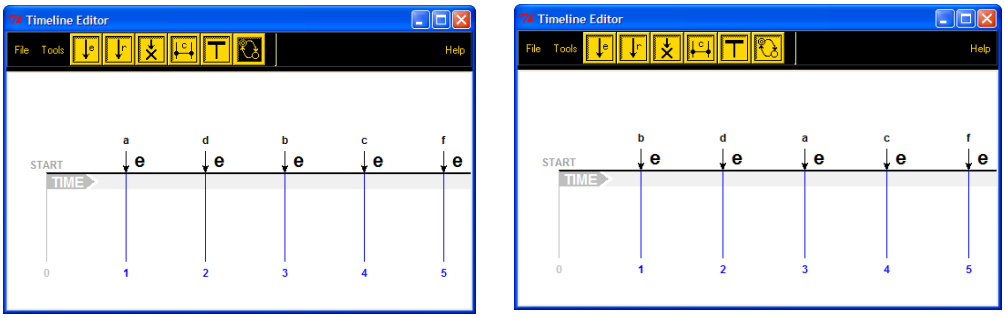


- En estos gráficos ocurre el evento ***a*** o el evento ***b*** antes que el ***d***, ***c*** y ***f***.

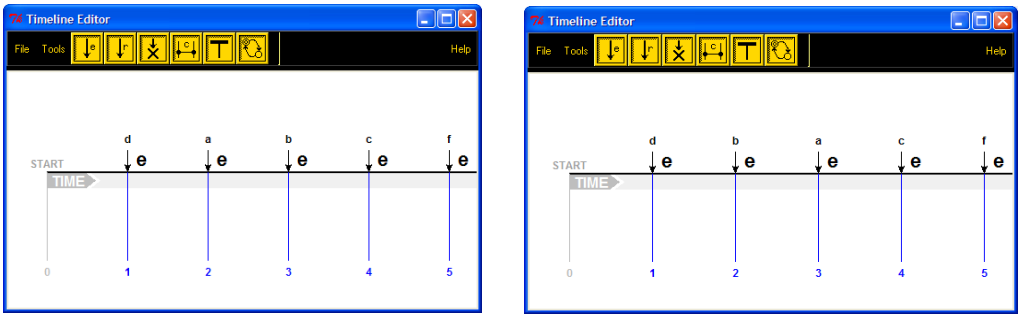




- En estos gráficos ocurre el evento ***d*** entre los eventos ***a*** y ***b***. Luego ocurren los eventos ***c*** y ***f***.



- En estos gráficos ocurre el evento ***d*** y luego los eventos ***a*** o ***b*** antes que los eventos ***c*** y ***f***.



✓ Cuando en un escenario VTS se indica un relación de asimetría entre puntos, esto se representa en TE con distintas marcas con igual evento asociado pero en distintos momentos.

En este caso los puntos ***p1*** y ***p2*** tienen la misma etiqueta pero no es el mismo evento.

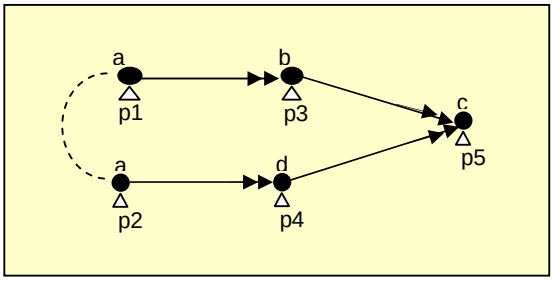
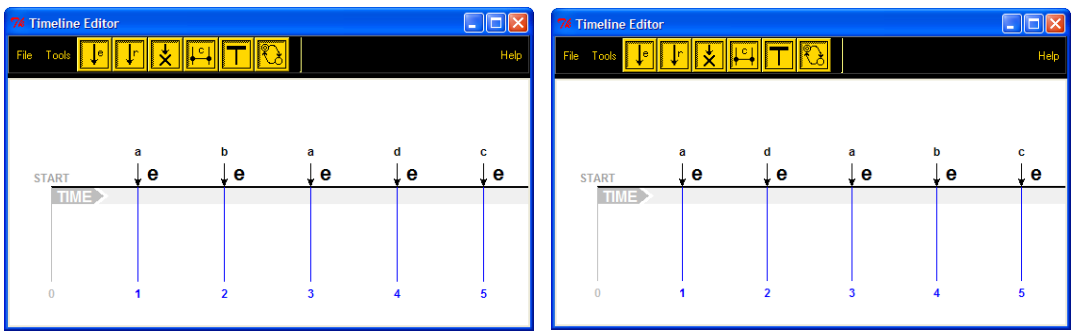
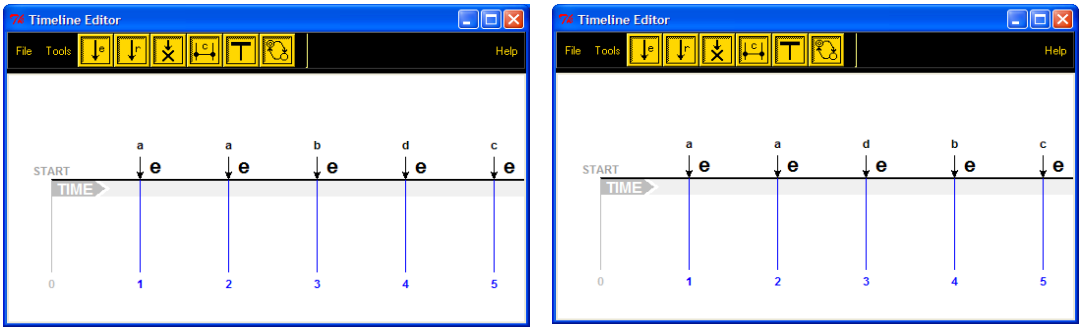


Figura 49: Escenario VTS con relación de asimetría

Se necesitan los siguientes 4 gráficos de TE para representar este escenario de VTS.

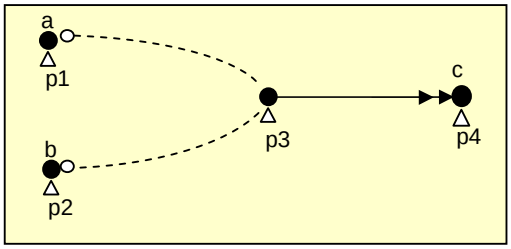




Si en este ejemplo quitamos la restricción de asimetría en el escenario VTS, se necesitaría un gráfico más similar al último pero con una sola marca con el evento **a** asociado (tendría una marca menos, sin la marca 1 o 2).

✓ Cuando se tiene en cuenta los puntos primeros y últimos en un escenario VTS, se necesita, como mínimo, un gráfico TE por cada punto de estos conjuntos. Considerando los siguientes casos de escenarios de VTS, se puede observar que en el ejemplo ***caso A*** se muestra un conjunto de primeros mientras que en el ***caso B*** un conjunto de últimos.

Caso A



Caso B

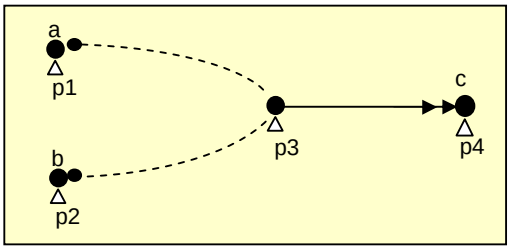
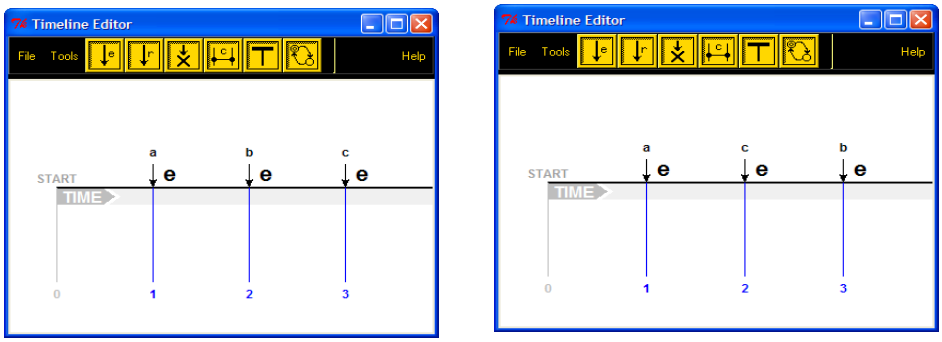
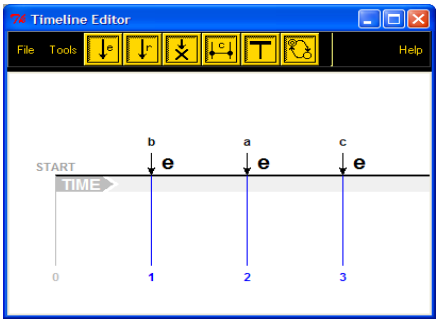


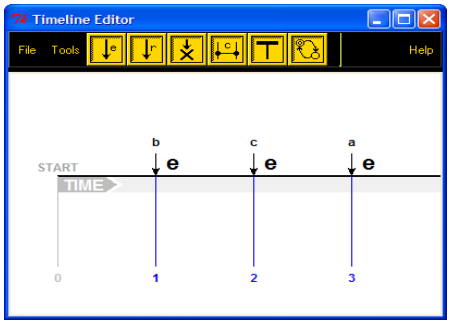
Figura 50: Escenarios VTS con puntos primeros y últimos

En el ***caso A*** en ***p3*** se asocia el evento que ocurra primero entre ***a*** y ***b***. Mientras en el ***caso B*** en ***p3*** se asocia el evento que ocurra en último lugar entre ***a*** y ***b.***

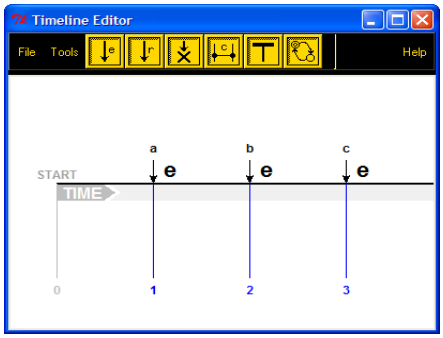
El **caso A** estaría representado por los siguientes 4 gráficos de TE donde ocurre en un caso el evento ***a*** y en el otro el evento ***b*** como primero del grupo:

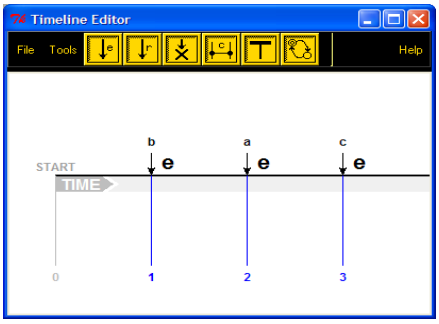






El **caso B** estaría representado por los siguientes 2 gráficos de TE donde ocurre en un caso el evento ***a*** y en el otro el evento ***b*** como último del grupo:





En un ejemplo más complejo podríamos observar un escenario de VTS como el siguiente:

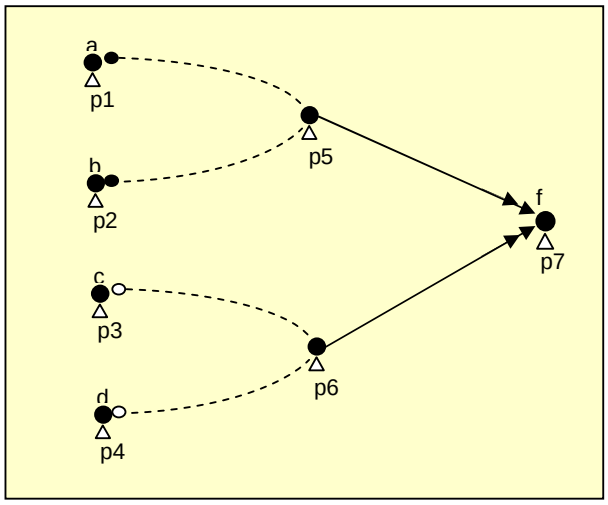
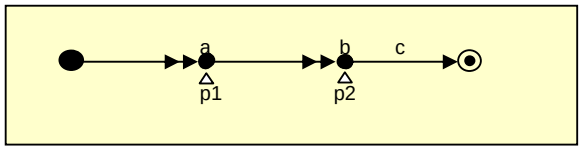


Figura 51: Escenario VTS que combina puntos primeros y últimos

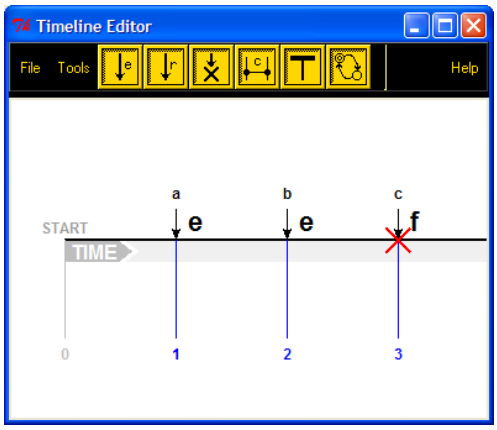
Se necesitarían 28 gráficos TE para representar este escenario VTS mostrando la siguiente secuencias de puntos.

p1 p2 p3 p4 p7	p2 p1 p3 p4 p7	p3 p1 p4 p2 p7	p4 p1 p2 p3 p7
p1 p2 p3 p7 p4	p2 p1 p3 p7 p4	p3 p1 p2 p4 p7	p4 p1 p2 p7 p3
p1 p2 p4 p3 p7	p2 p1 p4 p3 p7	p3 p1 p2 p7 p4	p4 p1 p3 p2 p7
p1 p2 p4 p7 p3	p2 p1 p4 p7 p3	p3 p2 p1 p4 p7	p4 p2 p1 p3 p7
p1 p3 p2 p4 p7	p2 p3 p1 p4 p7	p3 p2 p1 p7 p4	p4 p2 p1 p7 p3
p1 p3 p2 p7 p4	p2 p3 p1 p7 p4	p3 p2 p4 p1 p7	p4 p2 p3 p1 p7
p1 p3 p4 p2 p7	p2 p3 p4 p1 p7	p3 p4 p1 p2 p7	p4 p3 p1 p2 p7
p1 p4 p2 p3 p7	p2 p4 p1 p3 p7	p3 p4 p2 p1 p7	p4 p3 p2 p1 p7
p1 p4 p2 p7 p3	p2 p4 p1 p7 p3		
p1 p4 p3 p2 p7	p2 p4 p3 p1 p7		

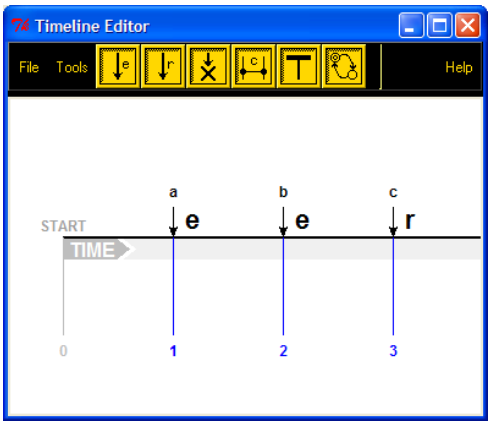
✓ Cuando se observa un punto final con un evento prohibido en VTS, se puede mostrar esto en un gráfico TE con dicho evento prohibido asociado a un marca de tipo de falla o de tipo requerido indicando si el escenario VTS debe ser verdadero o falso.
En el siguiente escenario VTS el evento ***c*** puede ser una marca de falla si el escenario debe ser verdadero (caso A) o una marca de requerimiento si es falso (caso B). En ambos casos el escenario VTS es el siguiente:



Caso A



Caso B



4.2.2 Relación textual

Si bien VTS tiene un aspecto textual en XML, no se puede relacionar con TE porque esta herramienta carece de dicho aspecto.

4.2.3 Relación formal

Dada la sencillez de los escenarios TE y la mayor expresividad de VTS, esta relación sólo puede darse en ciertos escenarios VTS. Cuando se observe un punto en un escenario VTS al cuál llega más de una flecha implicará más de un escenario en TE. Lo mismo sucede con el primer o último punto de un conjunto.

Debido a las diferencias entre estas herramientas mencionadas anteriormente en la traducción formal de TE a VTS se perderá la información de $<_F$, $<_L$ y δ de la definición formal de un escenario VTS.

Tampoco se podrán expresar en TE los escenarios condicionales de VTS dado que en estos escenarios cada vez que se detecta el antecedente se tiene que detectar un consecuente. En TE los eventos suceden una sola vez.

Por las diferencias enumeradas sólo se podrán traducir a TE los escenarios VTS que cumplan las siguientes condiciones:

- los eventos se sucedan en un cierto orden (dado por la relación de precedencia),
- sólo hay un evento por punto,
- cada evento será el primero que suceda desde el evento anterior,
- no hay restricciones temporales,
- no se consideran escenarios condicionales,
- si a un punto del escenario VTS llega más de una flecha, se tendrá más de un escenario TE y
- si $\not\equiv = \varnothing$ y se observan por precedencia 2 puntos contiguos con el mismo evento asignado, se traducirá en dos escenarios TE, uno con 2 marcas con ese evento y otro con una sola marca.

De VTS a TE	
El escenario $S_{TE} = \langle E; M; a; t; S; \alpha \rangle$ que resulta a partir de un escenario $S_{VTS} = \langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$ se obtendrá de la siguiente manera:	
E	• Se relaciona con $\sum$. Son el mismo conjunto.
M	• Se relacionan con las marcas del conjunto P, se asigna una marca a cada punto. • Se debe tener en cuenta el orden en que se dan estos eventos, o sea que se debe enumerar las marcas según se suceden los eventos en el escenario VTS definidos en $<$. • Tener en cuenta que en TE siempre aparece en el gráfico la marca 0 o comienzo y en VTS no siempre.
a	• Se relaciona con ℓ coincidiendo el tipo de relación: asigna un evento a una marca sin tener en cuenta la marca 0 de inicio.
t	• En general, a los puntos de VTS podemos asignarles el tipo e. • El tipo f y r se observarán cuando se tengan eventos prohibidos en la flecha entre un punto y un punto final de un escenario VTS. Dependiendo de si el escenario VTS es verdadero o falso la marca será de tipo r o f.

TT , TF, FT , FF $S = TT \cup TF \cup FT \cup FF$	• Se relaciona con los elementos de $<$ que tienen eventos prohibidos asociados. Estos serán del tipo TT con extremos distintos.
α	• Se relaciona con γ. • En TE siempre el evento asociado a una marca es el primero que sucede desde la marca anterior por lo que se asume como evento prohibido este evento entre la marca anterior y la marca a la que está asociado.

A continuación se presentan ejemplos de traducción de un escenario VTS a uno TE. En estos ejemplos vemos una porción de traza o corrida por lo que no se indican los puntos de inicio y fin.

Ejemplo 1

Teniendo el siguiente escenario VTS:

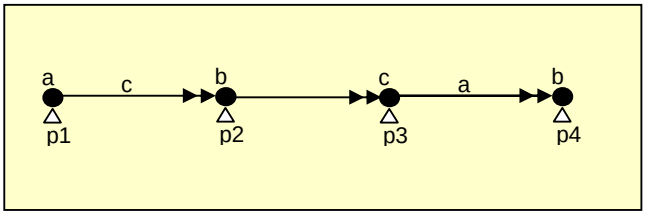


Figura 52: Ejemplo 1 de pasaje de un escenario VTS a TE

y su correspondiente definición formal, se aplican las reglas anteriormente indicadas y se obtiene el siguiente escenario TE:

$S = \langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$ donde:

Σ = {a; b; c}

P = {p1; p2; p3; p4}

ℓ = {p1→{a}; p2→{b}; p3→{c}; p4→{b}}

$\not\equiv$ = ∅

$<$ = {(p1, p2); (p2, p3); (p3, p4)}

$<_F$ =∅

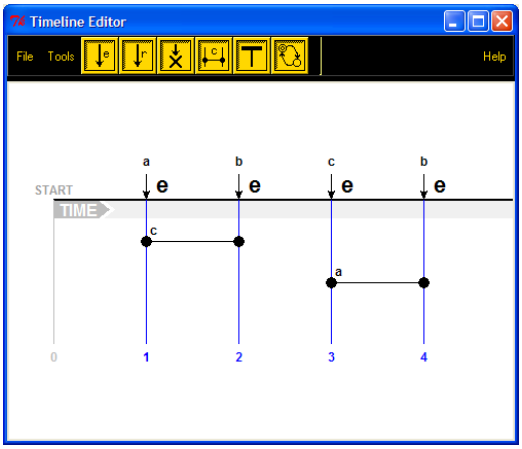
$<_L$ = ∅

γ = {(p1, p2) →{c, b}; (p2, p3) →{c}; (p3, p4) →{a, b}}

δ =∅

Aplicando lo dicho anteriormente, se obtiene el siguiente escenario TE $S = \langle E; M; a; t; S; \alpha \rangle$ de la siguiente manera:
E = {a; b; c }
M = { 0;1; 2; 3; 4}
a = { 0 →start; 1 →a; 2 →b; 3 →c; 4 → b }
t = { 1 →e; 2 →e; 3 →e; 4 →e }
TT = {⟨1,2⟩ ; ⟨3,4⟩ }
TF = ∅
FT = ∅
FF = ∅
S = {⟨1,2⟩ ; ⟨3,4⟩ }
α = {⟨1,2⟩ →c; ⟨3,4⟩→a }

El gráfico de este escenario TE es el siguiente:



Ejemplo 2

Consideremos el escenario de VTS de la Figura 53

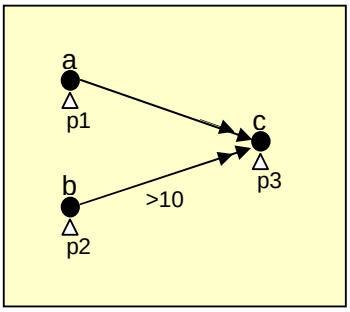


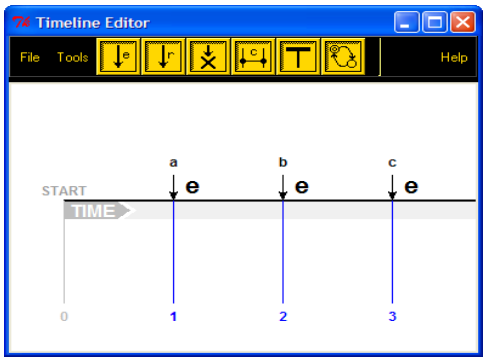
Figura 53: Ejemplo 2 de pasaje de un escenario VTS a TE

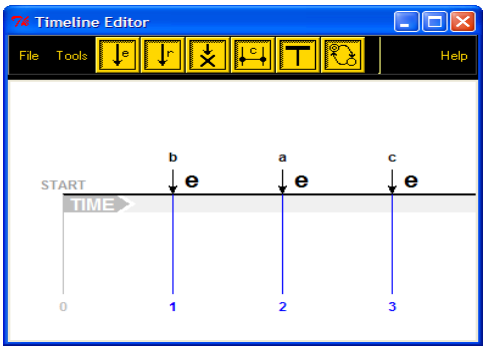
y su definición formal:

$S = \langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$ donde:
Σ = {a; b; c}
P = {p1; p2; p3}
ℓ = {p1→{a}; p2→{b}; p3→{c}}
$\not\equiv$ = $\varnothing$
$<$ = {(p1, p3); (p2, p3)}
$<_F$ =$\varnothing$
$<_L$ = $\varnothing$
γ = {(p1, p3) →{c}; (p2, p3) →{c}}
δ = {(p2, p3) → (10,∝)}

Dado que en $<$ hay dos flechas que llegan al punto p3 (desde p1 y p2) se tendrán dos escenarios: en uno se dará primero el punto p1 y luego el p2 antes de p3 y en el otro se dará primero el punto p2 y luego el p1 antes de p3.

Este escenario traducido a TE serían dos gráficos pero se perdería la información de la restricción de tiempo:





$S = \langle E; M; a; t; S; \alpha \rangle$ donde:

E = {a; b; c }
M = { 0;1; 2; 3}
a = { 0 →start; 1 →a; 2 →b; 3 →c }
t = { 1 →e; 2 →e; 3 →e }
TT = ∅
TF = ∅
FT = ∅
FF = ∅
S = ∅
α = ∅

$S = \langle E; M; a; t; S; \alpha \rangle$ donde:

E = {a; b; c }
M = { 0;1; 2; 3}
a = { 0 →start; 1 →b; 2 →a; 3 →c }
t = { 1 →e; 2 →e; 3 →e }
TT = ∅
TF = ∅
FT = ∅
FF = ∅
S = ∅
α =∅

Ejemplo 3

El escenario condicional VTS de la Figura 54 no puede ser traducido a un escenario TE.

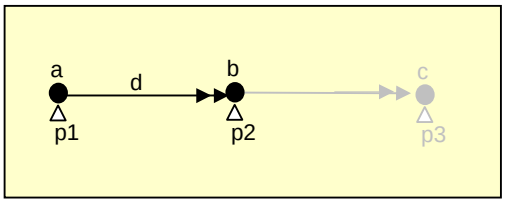


Figura 54: Ejemplo 3 de pasaje de un escenario VTS a TE

Ejemplo 4

Considerando el escenario VTS de la Figura 55 y su definición formal:

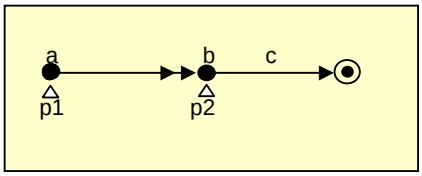


Figura 55: Ejemplo 4 de pasaje de un escenario VTS a TE

$S = \langle E; M; a; t; S; \alpha \rangle$ donde:

Σ = {a; b}

P = {p1; p2}

ℓ = {p1→{a}; p2→{b}}

$\not\equiv$ = $\varnothing$

$<$ = {(p1, p2)}

$<_F$ =$\varnothing$

$<_L$ = $\varnothing$

γ = {(p1, p2) →{b}; (p2,∝) →{c}}

δ = $\varnothing$

Al pasar a TE se aplica lo visto en el ejemplo 1 y se agrega una marca 3 de tipo f si el escenario de VTS debe ser verdadero o una marca 3 de tipo r si debe ser falso. Si se asume en este ejemplo que debe ser verdadero, se tienen entonces la siguiente definición formal del escenario TE:

$S = \langle E; M; a; t; S; \alpha \rangle$ donde:

E = {a; b; c;}

M = {0; 1; 2; 3}

a = {0 →start; 1 →a; 2 →b; 3 →c}

t = {1 →e; 2 →e; 3 →f}

TT = $\varnothing$

TF = $\varnothing$

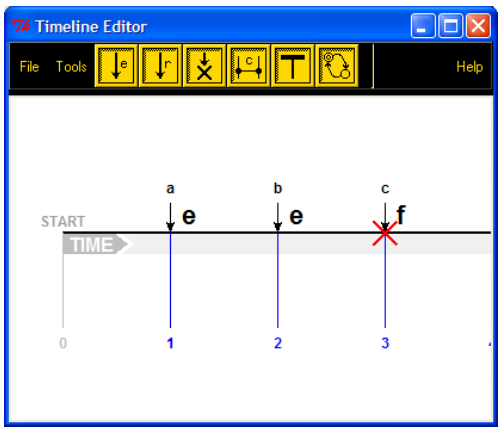
FT = $\varnothing$

FF = $\varnothing$

S = $\varnothing$

α = $\varnothing$

El gráfico que representa este escenario es el siguiente:



Ejemplo 5

Considerando el escenario VTS de la Figura 56 y su definición formal:

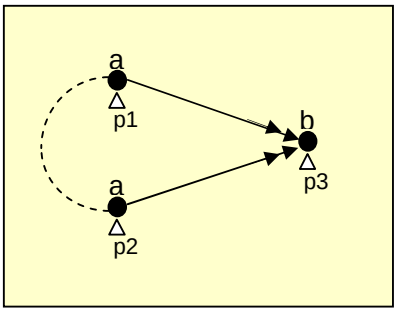


Figura 56: Ejemplo 5 de pasaje de un escenario VTS a TE

$S = \langle E; M; a; t; S; \alpha \rangle$ donde:

Σ = {a; b}

P = {p1; p2; p3}

ℓ = {p1→{a}; p2→{a}; p3→{b}}

$\not\equiv$ = {(p1, p2)}

$<$ = {(p1, p3); (p2, p3)}

$<_F$ =∅

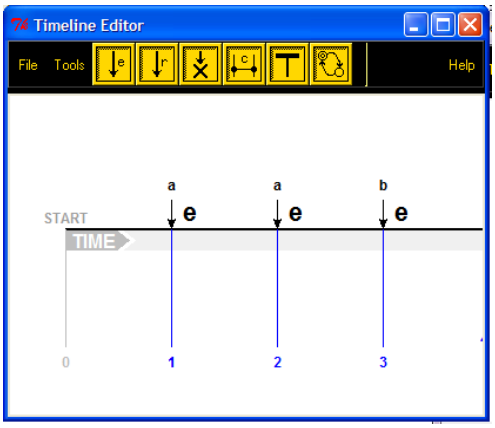
$<_L$ = ∅

γ = {(p1, p3) →{b}; (p2, p3) →{b}}

δ = ∅

En este caso no hay un orden estricto entre el punto p1 y el punto p2; ambos deben darse antes del punto p3 pero no importa cuál se observa antes. Esto implicarían dos escenarios TE: en uno se da primero el punto p1 y en el otro primero es p2. Pero dado que ambos

puntos tienen el mismo evento asociado, estos dos escenarios serían idénticos. O sea que basta un solo escenario TE como el siguiente:



Si se considera un escenario igual al de la Figura 56 pero sin la línea punteada (lo que en su descripción formal sería $\not\equiv = \emptyset$) además del escenario VTS obtenido anteriormente sería necesario un nuevo gráfico TE con solo 2 marcas, una con el evento a asociado y otra con el evento b.

4.2.4 Ventajas y desventajas

Después de realizar distintos ejemplos y casos posibles, se puede decir que no todo lo que se puede representar en un escenario VTS se puede graficar en TE. Se han observado las siguientes desventajas en los resultados de TE con respecto a VTS:

- ✓ En VTS indicamos la ocurrencia de un evento en un punto pero eso no significa que sea el primero de ese evento que ocurre desde el punto anterior. En TE los eventos asociados a las marcas indican que es el primero que ocurre desde la marca anterior.
- ✓ En VTS podemos tener más de un evento (conjunto) asociado a un punto. Esto indica que ocurrirá uno de esos eventos asociados a ese punto. En TE se asocia un determinado evento a una marca por lo que se necesitan más de un gráfico para representar este caso.
- ✓ En VTS podemos tener restricciones temporarias dadas por la relación δ, en TE no existen.
- ✓ En VTS existe la posibilidad de plantear un escenario donde ocurran eventos en cualquier orden antes se seguir con la corrida. Esto sería más de un gráfico para TE. En VTS el orden es parcial mientras que en TE es total, menos flexible.
- ✓ En VTS podemos enfocarnos en el primer o último evento que ocurra dentro de un grupo. En TE no tenemos esta posibilidad pues los eventos ocurren ordenadamente en el tiempo. Esto sería más de un gráfico para TE.
- ✓ En VTS podemos indicar que siempre que se produzca cierta secuencia de eventos (antecedentes) entonces deberá observarse la sucesión de determinados eventos (consecuente). En TE no se puede expresar esto puesto que los eventos suceden una sola vez.

Por lo enumerado anteriormente se puede concluir que:

- ✓ Los gráficos de VTS son mucho más flexibles permitiendo modelar mayor variedades de escenarios posibles en un único gráfico. Además permite incorporar restricciones temporales.

4.3 Similitudes y diferencias entre TE y VTS

Teniendo en cuenta las definiciones de escenarios de VTS y TE y las traducciones de uno a otro se observan las siguientes consideraciones:

Similitudes:

- ✓ En ambas herramientas se definen conjuntos de eventos. (Σ y *E* en VTS y TE respectivamente).
- ✓ En VTS tenemos un conjunto de puntos *P* mientras que en TE se define un conjunto de marcas *M*. Si bien son cosas diferentes las podemos vincular conceptualmente al graficar los escenarios.
- ✓ Tanto en VTS como en TE se define una relación que vincula los eventos a los puntos (ℓ en VTS) y a las marcas (a en TE). En este último caso es un único evento.
- ✓ En ambos casos encontramos relaciones de restricciones de eventos prohibidos (γ en VTS y α en TE).

Diferencias:

- ✓ En TE tenemos un conjunto de restricciones (*S*) que van de marca a marca y son de distintos tipos dependiendo si incluyen o no las marcas de inicio y fin.
- ✓ En TE se puede definir marcas de distintos tipos (e, f y r) a través de la relación *t*. En VTS se pueden interpretar las marcas de tipo f y r como restricciones de eventos prohibidos pero se pierde la información del tipo de marca en la traducción.
- ✓ En VTS tenemos la relación $<$ que vincula a los puntos en un orden parcial permitiendo una mayor flexibilidad en el orden en que se suceden los eventos. Esto permite en VTS mostrar en un único gráfico un escenario que tendría asociadas distintas trazas posibles según la secuencia en que se dan estos eventos. En TE los eventos ocurren ordenadamente en el tiempo lo que queda remarcado al definir *M* como un conjunto totalmente ordenado.
- ✓ En VTS existen escenarios donde no se sabe el orden en que sucederán los eventos (relaciones en $<$, por ejemplo puntos a los que llegan varias flechas).o eventos simultáneos pero distintos (relaciones en $\not\equiv$).
- ✓ En VTS podemos hablar del evento que ocurra primero o último de un grupo de eventos ($<_F$,$<_L$).
- ✓ En VTS se pueden asociar restricciones de tiempo (δ), lo que no es posible en TE.
- ✓ En VTS se pueden mostrar escenarios condicionales a través de antecedentes y consecuentes. Estos consecuentes pueden encontrarse delante, en el medio o al final del antecedente. Esto no es posible expresarlo en TE.

Capítulo 5: Comparación MSC y VTS

En este capítulo se realiza una comparación de MSC con relación a VTS. Antes de comenzar se tendrán en cuenta ciertas consideraciones al observar los gráficos MSC.
Para poder realizar una comparación real de estas dos herramientas debemos diferenciar sus filosofías.
Existen diferencias entre los lenguajes comúnmente utilizados en la industria como MSC y los que se manejan en el ámbito científico como VTS. Las herramientas utilizadas en la industria suelen mostrar, generalmente en forma gráfica, lo que se requiere que cumpla el sistema. Su finalidad principal es describir el sistema permitiendo una mejor comunicación entre los diseñadores y quienes harán la implementación o también como una forma de documentar el sistema desarrollado. Esto no implica que tengan la base formal que permita aplicar técnicas automáticas para su verificación.
Los lenguajes que se manejan en el ámbito académico utilizan notación formal (lógicas, autómatas, algunos lenguajes gráficos) para expresar requerimientos. Luego se analiza el modelo obtenido del sistema para determinar si el mismo satisface o no las propiedades, es decir, se verifica si el sistema cumple o viola los requerimientos especificados. Estas herramientas cuentan con una base formal que permiten una verificación entre el modelo y las propiedades que no suelen tener las utilizadas en la industria.

La ventaja del MSC es que su forma gráfica es muy sencilla de entender pero establecer las corridas que se obtienen del escenario graficado no lo es.
En cambio, VTS al ser un lenguaje formal, permite determinar estas corridas que satisfacen las propiedades especificadas y verificarlas con las que se obtienen del modelo.

Como se vio en el capítulo 2, MSC es un lenguaje gráfico que se utiliza para especificar corridas (trazas) de sistemas que se enfocan en el intercambio de mensajes entre entidades y su entorno. Para poder establecer estas trazas o corridas es imprescindible determinar el orden en que se realizan estos intercambios de mensajes, cuándo se envían y cuándo se reciben los mensajes.

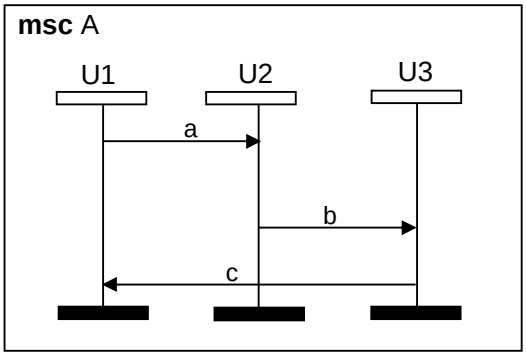


Figura 57: Ejemplo de gráfico MSC

Observando un gráfico MSC se puede establecer un orden “horizontal” y un orden “vertical” en que suceden los eventos. El orden horizontal está dado por mensajes: el envío de un mensaje es anterior a la recepción del mismo. El orden vertical está dado por la secuencia de los eventos en cada instancia.

Si se observa el gráfico de la Figura 57 se puede determinar la siguiente sucesión de eventos observando el orden “horizontal”:

- U1 envía el mensaje a
- U2 recibe el mensaje a
- U2 envía el mensaje b
- U3 recibe el mensaje b
- U3 envía el mensaje c
- U1 recibe el mensaje c

Si se observa la sucesión de eventos en las distintas instancias se puede determinar el siguiente orden “vertical”:

- U1 envía el mensaje a
- U1 recibe el mensaje c
- U2 recibe el mensaje a
- U2 envía el mensaje b
- U3 recibe el mensaje b
- U3 envía el mensaje c

Si ahora se observa el gráfico de la Figura 58 se puede decir con certeza que se puede determinar el siguiente orden horizontal:

- U2 envía el mensaje a
- U1 recibe el mensaje a

- U2 envía el mensaje b
- U3 recibe el mensaje b

- U3 envía el mensaje c
- U1 recibe el mensaje c

También se puede decir el orden en que suceden los eventos en las instancias:

- U2 envía el mensaje a
- U2 envía el mensaje b
- U3 recibe el mensaje b
- U3 envía el mensaje c

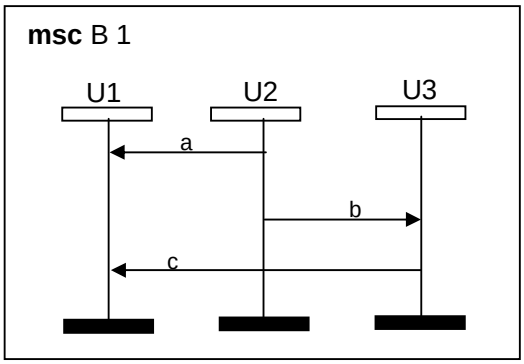


Figura 58: Ejemplo de gráfico MSC

Pero no se puede asegurar que U1 recibe el mensaje ***a*** antes que el mensaje ***c***. Por este motivo se debe determinar antes de analizar un gráfico MSC si se aplica o no la política FIFO sobre los eventos que ocurren en una instancia [AHP96].

Si se analiza la secuencia de los eventos que suceden en la instancia U1 y se aplica política FIFO se asume entonces que se recibe el mensaje **a** antes que el mensaje **c**.
Cuando no se aplica política FIFO, el gráfico del MSC puede ser dibujado con un gráfico donde se observa una zona de co-región (ver punto 2.4.3 del Capítulo 2), esto es, una zona donde sabemos que se dan una serie de eventos pero no el orden estricto en que suceden. El gráfico MSC equivalente al de la Figura 58 con una co-región sería como lo muestra la Figura 59.

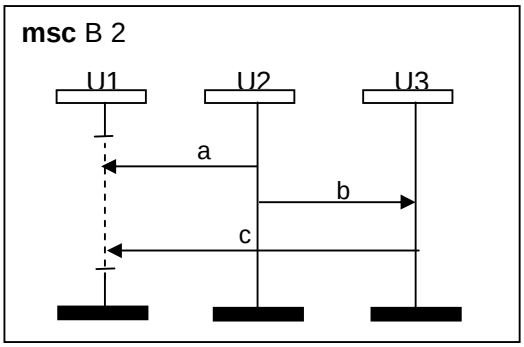
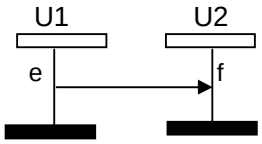


Figura 59: Gráfico MSC con co-región

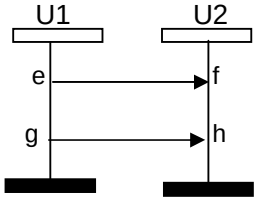
Considerando como eventos distintos el envío de un mensaje y la recepción del mismo, formalmente se asume:

- ✓ El evento envío de un mensaje (evento de tipo send) debe darse antes del evento de recepción de ese mismo mensaje (evento de tipo receive).



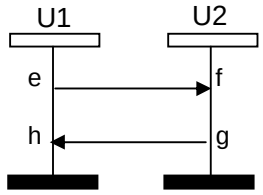
∃ e y ∃ f tal que
T (e) = s
T (f) = r,
m (e) = f y
entonces **e < f.**

- ✓ Cuando en la línea de tiempo de una instancia encontramos un evento ***s1*** del tipo send y luego otro evento ***s2*** del tipo send entonces ***s1 < s2.***



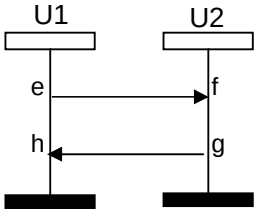
∃ e y ∃ g tal que
T (e) = T (g) = s y
L (e) = L (g) = U1
entonces **e < g.**

✓ Cuando en la línea de tiempo de una instancia encontramos un evento ***s*** del tipo send y luego un evento ***r*** del tipo receive entonces ***s < r.***



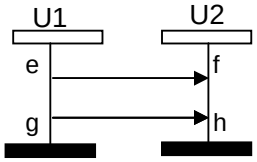
∃ e y ∃ h tal que
T (e) = s,
T (h) = r,
L (e) = L (h) = U1 y
entonces **e < h.**

✓ Cuando en la línea de tiempo de una instancia encontramos un evento ***r*** del tipo receive y luego un evento ***s*** del tipo send entonces ***r < s.***



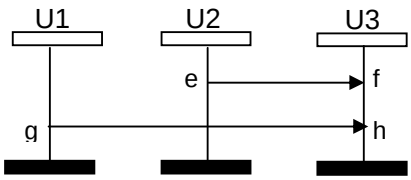
∃ f y ∃ g tal que
T (f) = r,
T (g) = s y
L (f) = L (g) = U2
entonces **f < g.**

✓ Cuando en la línea de tiempo de una instancia encontramos un evento ***r1*** del tipo receive y luego otro evento ***r2*** del tipo receive y ambos eventos tienen eventos send asociados con la misma instancia entonces ***r1 < r2.*** O sea :



∃ f y ∃ h tal que
m (f) = e, m (h) = g,
L (e) = L (g) =U1 con e < g,
T (f) = T (h) =r y
L (f) = L (h) =U2
entonces **f < h.**

✓ Cuando en la línea de tiempo de una instancia encontramos un evento ***r1*** del tipo receive y luego otro evento ***r2*** del tipo receive y ambos eventos tienen eventos send asociados con distintas instancias entonces ***r1 < r2*** si se aplica la política FIFO sino puede darse ***r1 y r2*** en cualquier orden.



∃ f y ∃ h g tal que
m (f) = e, m (h) = g,
L (e) $\neq L$ (g),
T (f) = T (h) =r y
L (f) = L (h) =U3
entonces **no se puede decir el orden de f y h (equivale a una co-región) o se define la aplicación de política FIFO con lo que tenemos que f < h.**

Tomando el siguiente gráfico MSC, se indican los eventos send y receive y se analiza cómo será el orden en que se dan los elementos en las instancias:

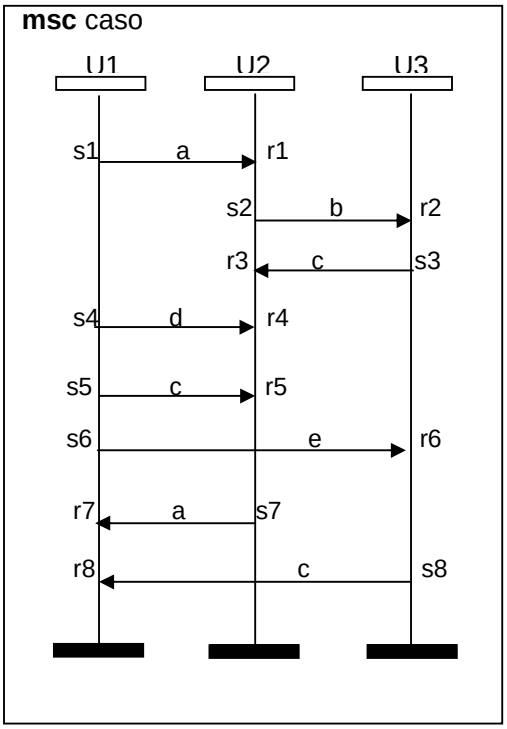


Figura 60: Ejemplo de escenario MSC

Se sabe que el orden horizontal entre los mensajes de tipo send y receive asociados al mismo mensaje es primero el evento send (de enviar) y luego el evento receive (de recibir). Por lo tanto se obtienen las siguientes afirmaciones :

s1 < r1 s5 < r5
s2 < r2 s6 < r6
s3 < r3 s7 < r7
s4 < r4 s8 < r8

Ahora si se observa el orden en que se dan los eventos en las distintas instancias se obtiene lo siguiente:

Para la instancia U3:
Sin dudas se sabe que r2<s3<r5<s8

Para la instancia U2:
Sin dudas se sabe que r1<s2<r3 y que r5<s7.
Por lo explicado anteriormente r4<r5 ya que los eventos send asociados a estos eventos receive provienen de la misma instancia. Entonces el orden será:
r1 < s2 < r3 < r4 < r5 < s7

Para la instancia U1:
Sin dudas se sabe que s1<s2<s4<s5<s6.
Lo que no se puede determinar es el orden en que se dan r7 y r8 ya que los eventos send asociados a estos eventos receive pertenecen a distintas instancias. Se debe indicar si se aplica o no política FIFO. En caso de aplicarse, la secuencia a partir del evento s6 será:
s6 < r7 < r8
Si no se aplica la política FIFO se pueden dar las siguientes secuencias a partir del evento s6:
s6 < r7 < r8
s6 < r8 < r7.

5.1 Pasando de un escenario MSC a uno VTS

5.1.1 Relación entre gráficos

Para un mejor entendimiento de cuáles son los eventos al pasar de un gráfico MSC a un escenario VTS, se podría adoptar alguna de las siguientes formas de nombrarlos:

Opción 1: Propuesto por David Harel y P.S. Thiagarajan en su trabajo *Message Sequence Charts* [HT03]. Se nombran los eventos como una tupla de la siguiente manera:

- ⟨**p!q;m**⟩ cuando son del tipo receive donde p es la instancia que envía el mensaje m y q la que recibe dicho mensaje,
- ⟨**q?p;m**⟩ cuando son del tipo send donde q es la instancia que recibe el mensaje m y p la que envió dicho mensaje y
- ⟨**p;m**⟩ cuando son del tipo local donde p es la instancia que realiza m

Opción 2: Propuesto por S. Mauw en su trabajo *The formalization of Message Sequence Charts* [M96]. Es parecido a cómo se describen los eventos en SDL. Se propone nombrar a los eventos como:

- ***in(p,q,m)*** cuando son del tipo receive donde p es la instancia que recibe el mensaje m y q la que envía dicho mensaje,
- ***out(p,q,m)*** cuando son del tipo send donde q es la instancia que envía el mensaje m y q la que recibe dicho mensaje y
- ***action (i, a)*** cuando son del tipo local donde i es la instancia en que se realiza la acción a.

Se optó por la segunda opción por asemejarse al aspecto textual de MSC.

Para pasar de un gráfico MSC a un escenario VTS se aplicarán las siguientes reglas:

1) Se considerarán gráficos MSC que no están dentro de un HMSC.
2) Para facilitar la identificación de los eventos que serán puntos con sus eventos asociados en el escenario VTS, se marcan los mismos con rojo sobre el gráfico MSC: cuando una instancia o actor envía o recibe mensajes, se realiza una acción, se definen timers, se observa un mensaje perdido o encontrado.

3) Los eventos de recepción de mensaje serán los primeros que ocurran desde el envío del mismo mensaje. Esto se representarán con flechas dobles:

Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):

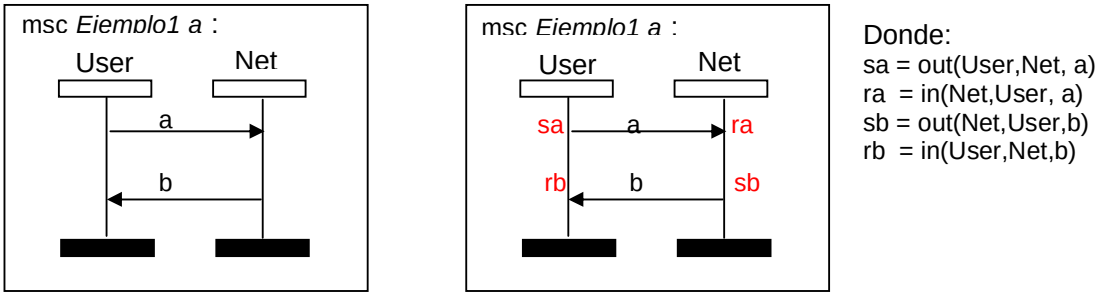
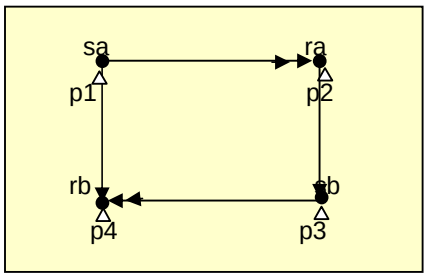
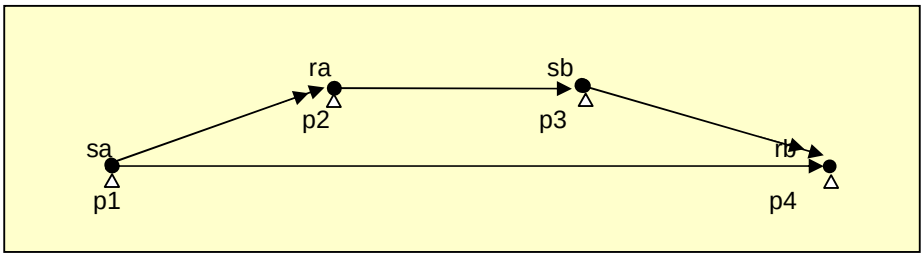


Figura 61: Ejemplo 1 a de MSC – MSC Básico

Se obtiene el siguiente escenario VTS según el orden horizontal y vertical establecido anteriormente:

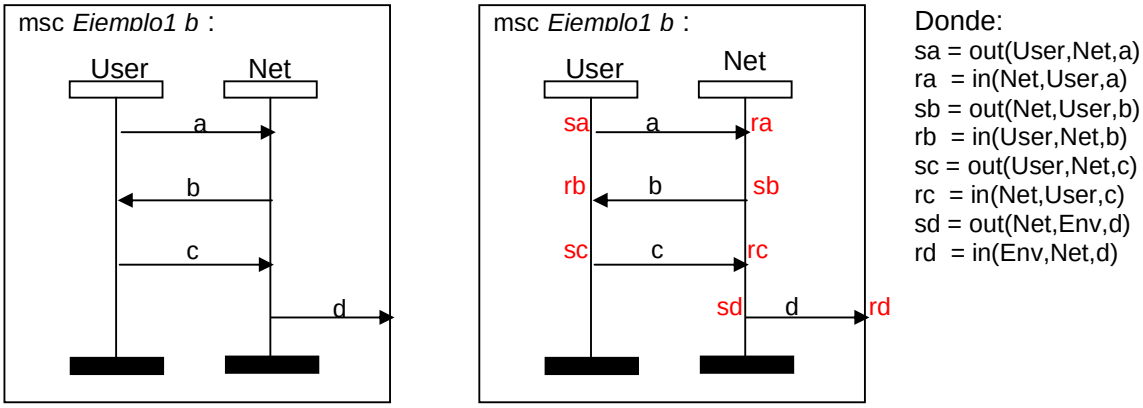


Visto el mismo gráfico pero dibujando los puntos de otra manera permite una interpretación más fácil en la sucesión de los eventos con una lectura de izquierda a derecha:



4) Se toman como eventos el envío y recepción de mensajes desde y hacia el entorno.

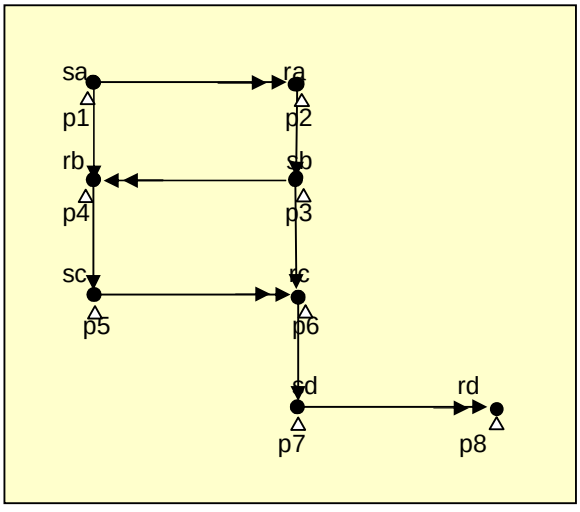
Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):



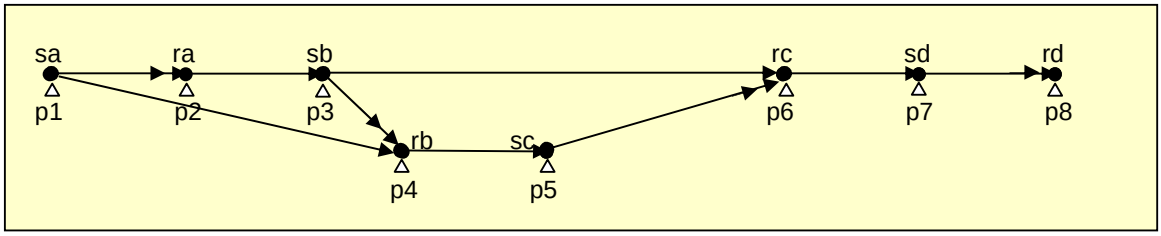
Donde:
sa = out(User,Net,a)
ra = in(Net,User,a)
sb = out(Net,User,b)
rb = in(User,Net,b)
sc = out(User,Net,c)
rc = in(Net,User,c)
sd = out(Net,Env,d)
rd = in(Env,Net,d)

Figura 62: Ejemplo 1 b de MSC - Mensajes hacia y desde el entorno

Se obtiene el siguiente escenario VTS:

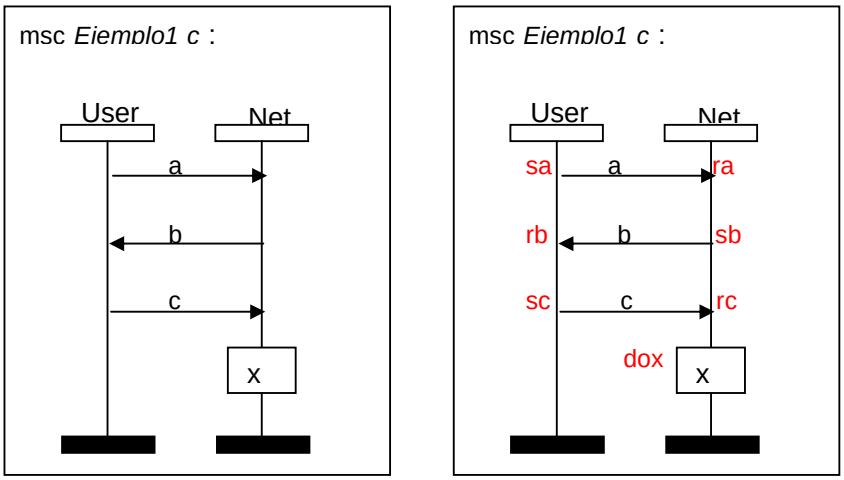


Visto de otra manera:



5) Las acciones realizadas por una instancia se marcan como puntos con la acción como evento asociado en el escenario VTS.

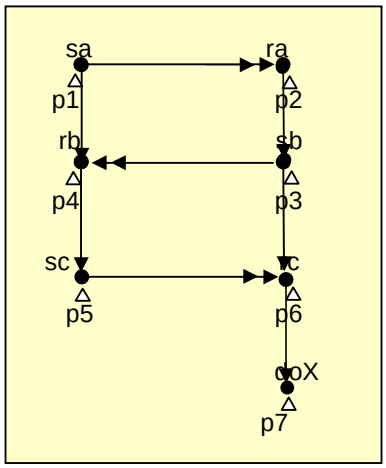
Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):



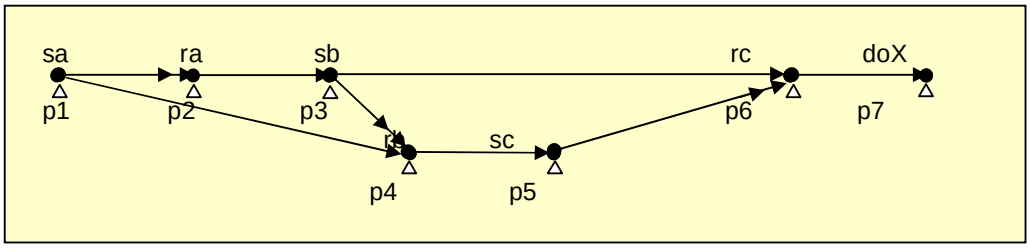
Donde:
sa = out(User,Net,a)
ra = in(Net,User,a)
sb = out(Net,User,b)
rb = in(User,Net,b)
sc = out(User,Net,c)
rc = in(Net,User,c)
doX = action(Net,X)

Figura 63: Ejemplo 1 c de MSC - Acciones

Se obtiene el siguiente escenario VTS:

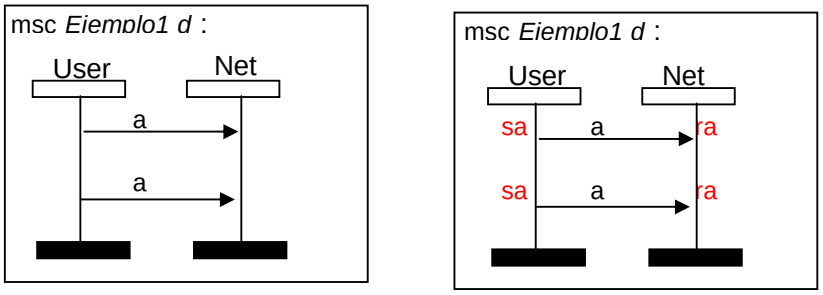


Visto de otra manera:



6) Cuando dos actores o instancias envían el mismo mensaje se puede reflejar en VTS como puntos distintos con el mismo evento asociado.

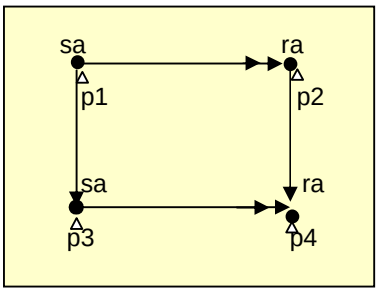
Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):



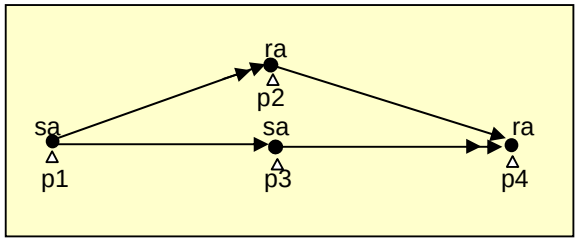
Donde:
sa = out(User,Net,a)
ra = in(Net,User,a)

Figura 64: Ejemplo 1 d de MSC - ¿Mismo mensaje o distintos?

En este caso el primer **sa** y **ra** deben ser distintos al segundo **sa** y **ra** respectivamente debido a que existe una relación de precedencia entre estos eventos (orden vertical). Se obtiene entonces el siguiente escenario VTS:



Visto de otra manera:



Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):

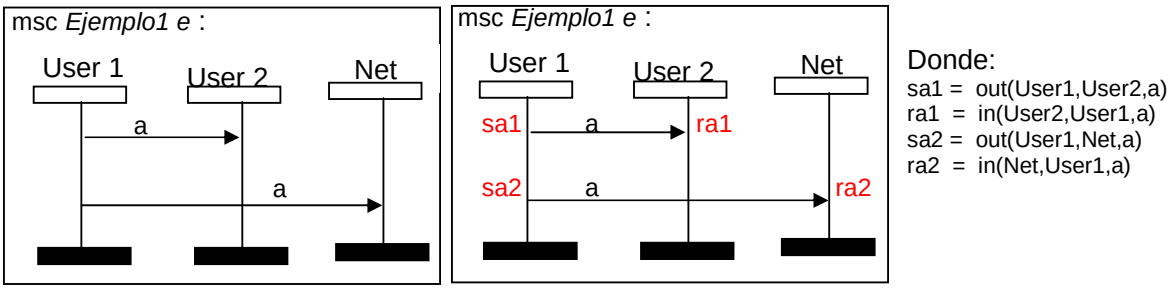
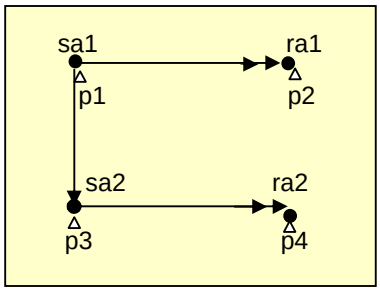
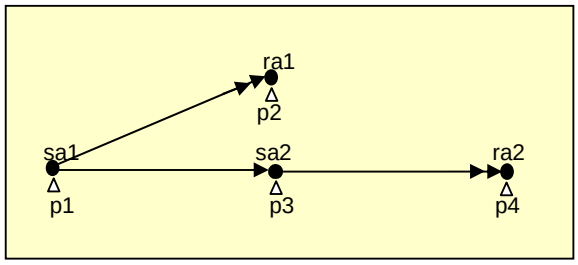


Figura 65: Ejemplo 1 e de MSC - ¿Mismo mensaje o distintos?. Otro caso

En este caso el primer y segundo ***sa*** y ***ra*** deben ser distintos porque a pesar de ser el mismo mensaje ***a*** y la misma instancia que los envía, los destinatarios son distintos. Se obtiene entonces el siguiente escenario VTS:



Visto de otra manera:



7) Cuando hay dos o más eventos que no puede determinarse en qué orden suceden pero corresponden a distintos actores o instancias, se puede expresar sin problemas en un escenario VTS.

Partiendo del escenario MSC de la Figura 66, se identifican los eventos para pasar a VTS (en rojo).

En este escenario VTS las corridas pueden tener el evento **rd** del punto p6 antes o después del evento **rb** del punto p4.
También se observa que se dibuja una flecha que une los puntos p1 y p4 según el orden vertical en la instancia User. En realidad no es necesario que dicha flecha se grafique ya que por la propiedad transitiva que tienen los eventos en un escenario VTS (orden parcial), este orden de p1<p4 ya está dado por p1<p2<p3<p4.

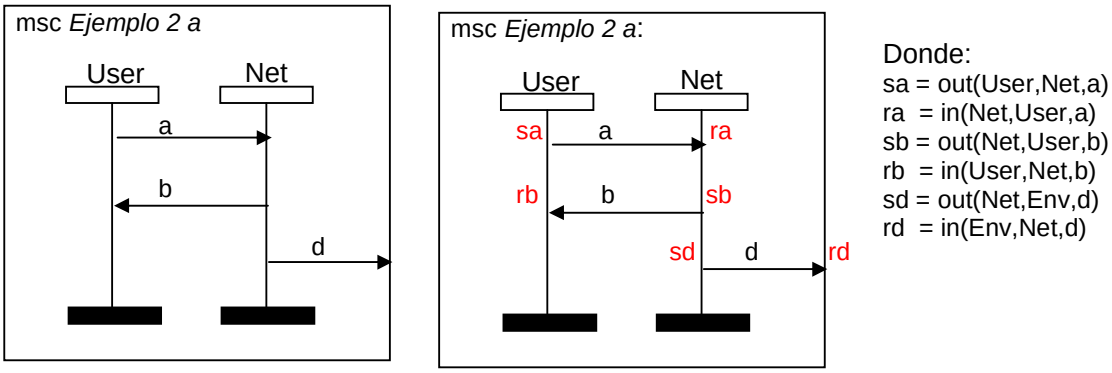
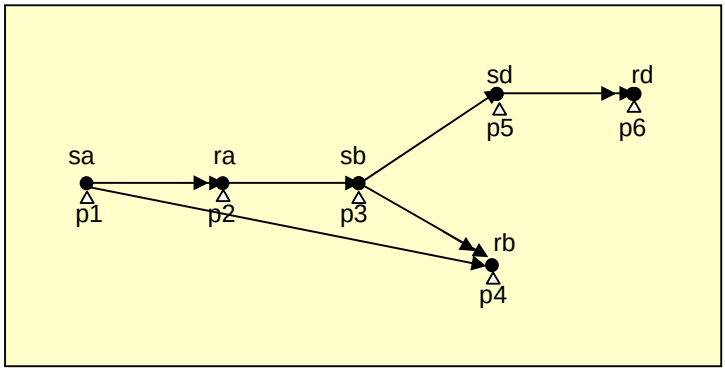


Figura 66: Ejemplo 2 a de MSC - Orden de eventos de distintos actores o instancias

El escenario VTS obtenido es el siguiente:



8) El orden de los eventos en las instancias estará dado por la secuencia en que se sucede, de arriba hacia abajo (ver Figura 61). Cuando no se puede determinar qué evento sucede primero en una instancia o actor (se reciben dos o más mensajes desde distintas instancias o actores), se deberá tener en cuenta si se aplica o no la política FIFO. Según se aplique o no la política FIFO se obtendrán distintos escenarios.

Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):

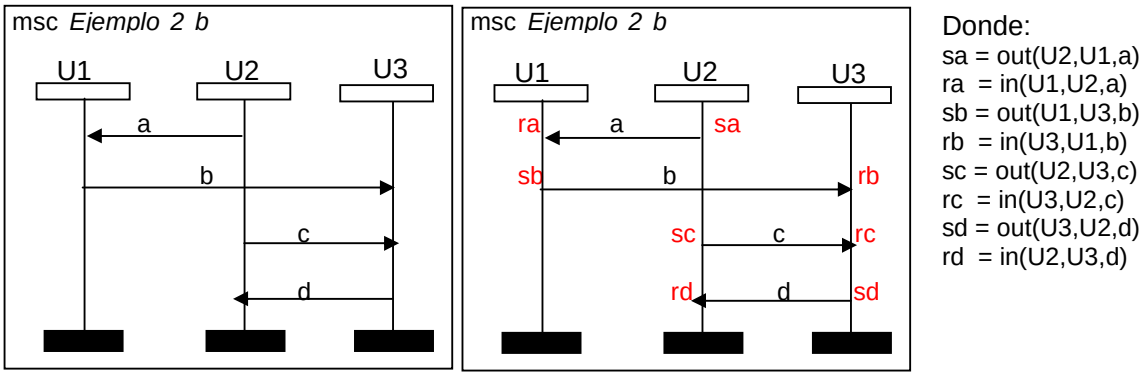
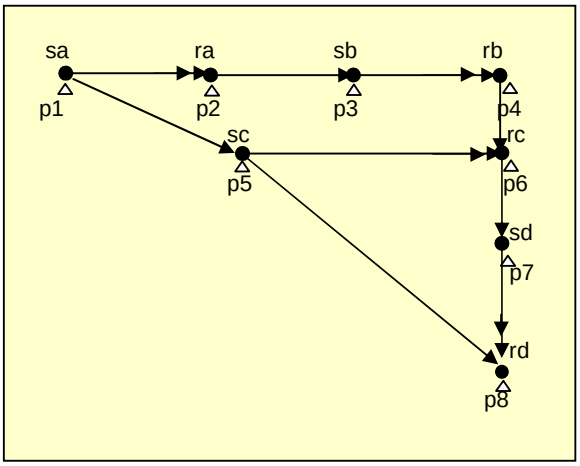


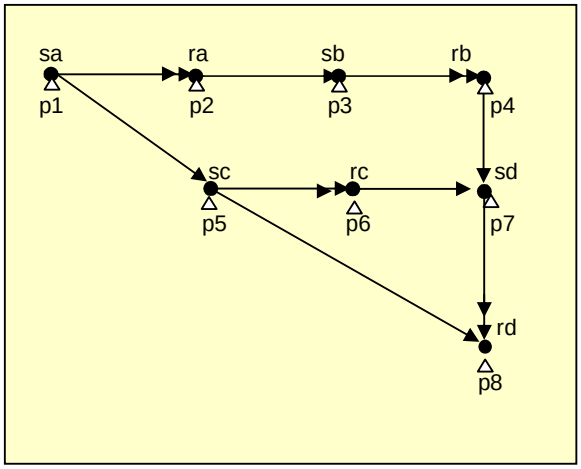
Figura 67: Ejemplo 2 b de MSC - Orden de eventos en un mismo actor o instancia

Se obtienen los siguientes gráficos VTS según se aplique o no la política FIFO:

Caso I : Aplicando política FIFO, o sea, ***rb*** sucede antes que ***rc***

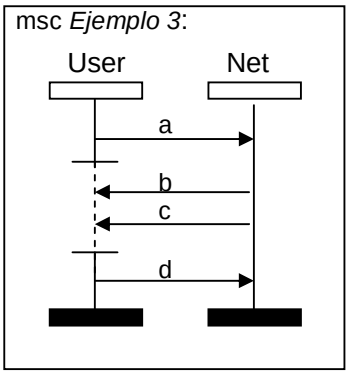


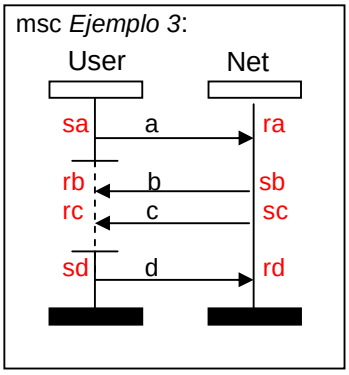
Caso II : Sin aplicar política FIFO. Esto es semejante a pensar que ***rb*** y ***rc*** se dan en una co-región (ver siguiente punto).



9) Cuando nos encontramos con una co-región en una instancia, los eventos se dan pero sin un orden determinado.

Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):

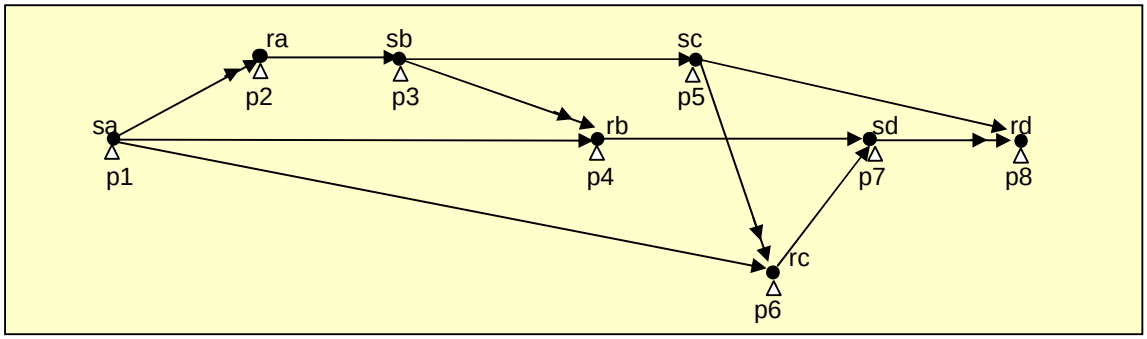




Donde:
sa = out(User,Net,a)
ra = in(Net,User,a)
sb = out(Net,User,b)
rb = in(User, Net,b)
sc = out(Net, User,c)
rc = in(User, Net,c)
sd = out(User, Net,d)
rd = in(Net, User,d)

Figura 68: Ejemplo 3 de MSC - Co-región

Obtenemos el siguiente escenario VTS:



10) Cuando en un gráfico MSC se observa el comienzo y finalización de una instancia, se tomarán el inicio y el final como eventos asociados a puntos que serán graficados en un escenario VTS.

Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):

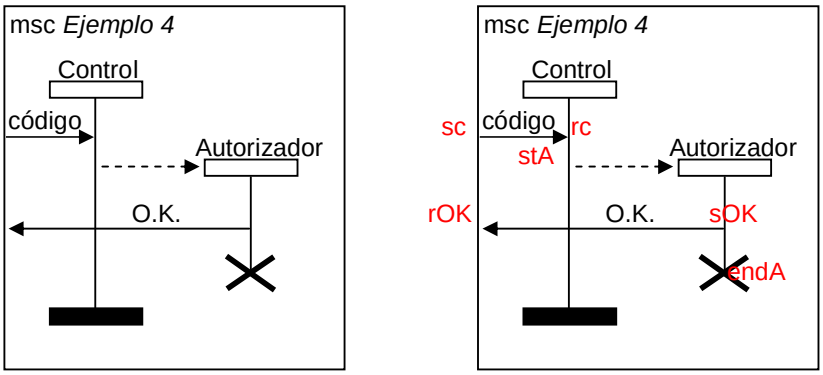
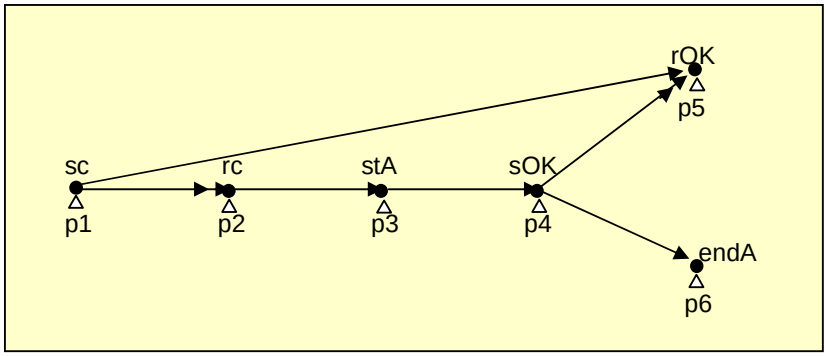


Figura 69: Ejemplo 4 de MSC - Comienzo y finalización de una instancia

Se obtiene el siguiente escenario VTS:



Donde:
sc = out(Env, Control,código)
rc = in(Control, Env, código)
stA = start(Autorizador)
sOK = out(Autorizador, Env, OK)
rOK = in(Env, Autorizador, OK)
endA = end(Autorizador)

11) Cuando en un gráfico MSC se observa mensajes encontrados se asume que el “medio de comunicación” es quién lo envió. Lo mismo sucede con los mensajes perdidos, alguna instancia envió un mensaje que fue recibido por este “medio de comunicación”. Estos eventos se asocian a puntos de un escenario VTS.

Partiendo del escenario MSC de la Figura 70, se identifican los eventos para pasar a VTS (en rojo).

En este caso se incorpora un evento raM para el mensaje encontrado por el medio de comunicación enviado por la instancia User1 y un scM para el mensaje perdido enviado por el medio de comunicación y recibido por la instancia User2.

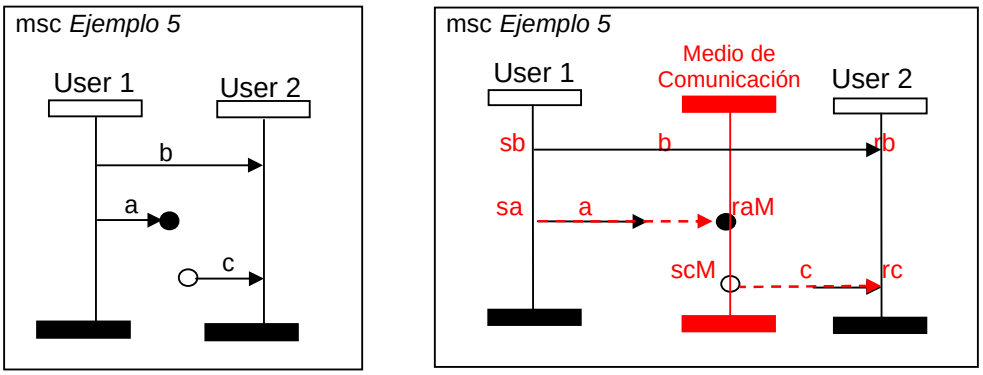
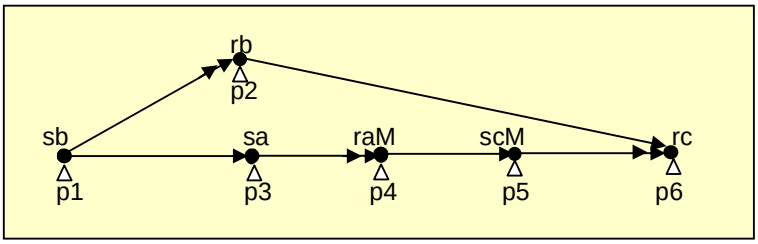


Figura 70: Ejemplo 5 de MSC - Mensajes perdidos y encontrados

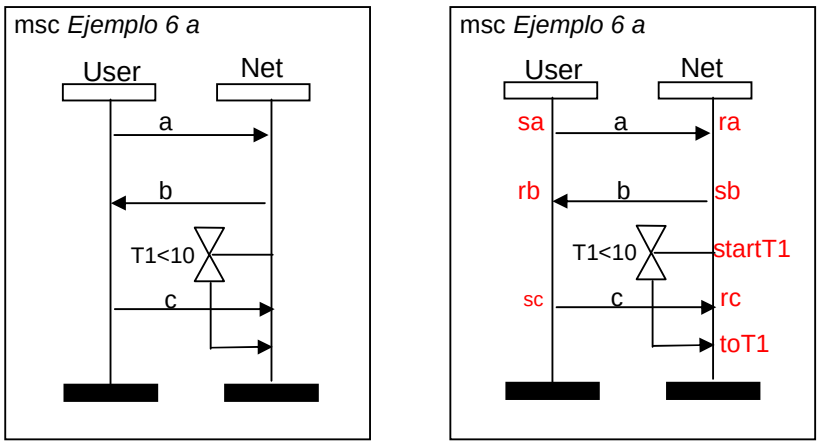
Se obtiene el siguiente escenario VTS:



Donde:
sb = out(User1,User2,b)
rb = in(User2, User1,b)
sa = out(User1,Medio,a)
raM = in(Medio,User1,a)
scM = out(Medio, User2,c)
ra = in(User2, Medio,a)

12) En los gráficos MSC se puede definir un timer. Cuando se comienza, termina o reinicia un timer en un gráfico MSC, se tendrá un punto con dicho evento asociado en el escenario VTS. Se incorporará al escenario VTS una restricción de tiempo si al definir timers en el gráfico MSC se indica el tiempo que debe transcurrir hasta su detención o reseteo.

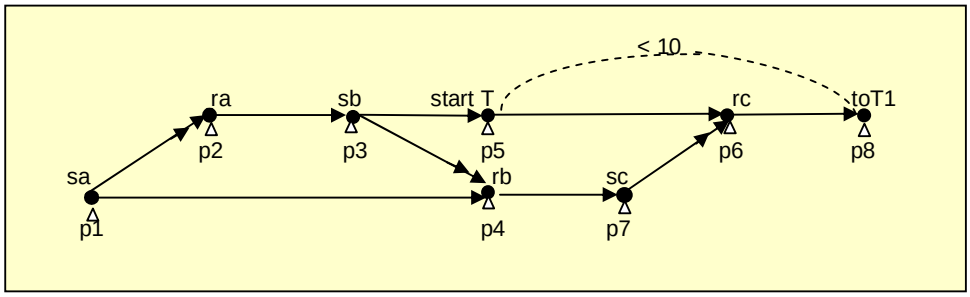
Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):



Donde:
sa = out(User,Net,a)
ra = in(Net,User,a)
sb = out(Net,User,b)
rb = in(User, Net,b)
startT1 = setT1 (Net,<20)
toT1 = stopT1 (Net)
sc = out(User,Net,,c)
rc = in(Net,User,c)

Figura 71: Ejemplo 6 a de MSC - Inicio del timer y detención por tiempo cumplido

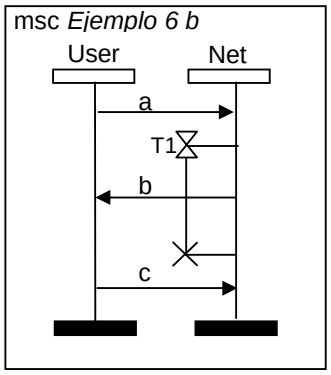
Se obtiene el siguiente VTS:

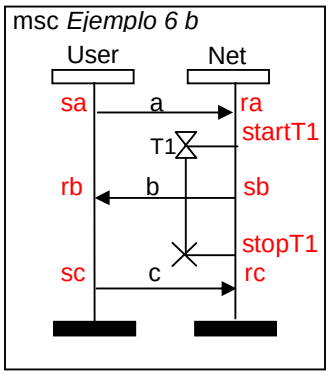


En el caso que se muestra en la Figura 71 se puede observar el inicio del timer, el envío y recepción de un mensaje mientras transcurre el tiempo del timer y la finalización del mismo luego de cumplir como máximo 10 unidades de tiempo (timeout).

En la Figura 72 se observa un gráfico MSC con un timer que no finaliza por haber cumplido el tiempo sino que se detiene por otro motivo. En el caso de timeot (Figura 71) se cumple todo el tiempo. Con el stop se detiene el timer antes de completar todo el tiempo.

Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):

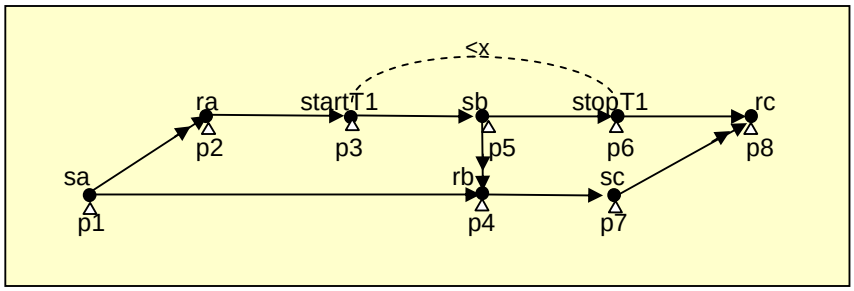




Donde:
sa = out(User,Net,a)
ra = in(Net,User,a)
sb = out(Net,User,b)
rb = in(User, Net,b)
startT1 = setT1 (Net,<20)
stopT1 = stopT1 (Net)
sc = out(User,Net,,c)
rc = in(Net,User,c)

Figura 72: Ejemplo 6 b de MSC - Inicio del timer y detención antes de cumplirse el tiempo

Obtenemos el siguiente VTS:



En la Figura 73 se observa un gráfico MSC con un timer que se reinicia transcurrido cierto tiempo. En este caso se consideran distintos eventos para el escenario VTS el inicio y el reinicio del timer.
En muchos escenarios de MSC con reseteo de timers se puede observar cómo se relacionan distintos gráficos entre sí por medio de timers. Por ejemplo, en un gráfico comienza o se resetea un timer y en otro gráfico se detiene. Para la comparación de MSC y VTS sólo se tomarán en cuenta los casos de timers de MSC que comienzan y terminan en el mismo gráfico. En los casos en que se comienza un timer en un gráfico MSC y se lo detiene o finaliza en otro se deberá graficar ambos gráficos MSC en un solo escenario VTS.

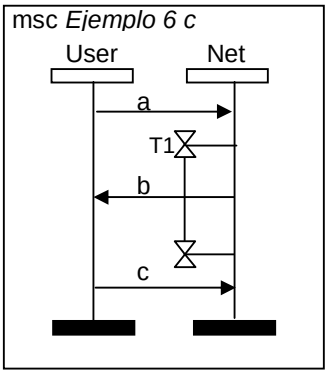
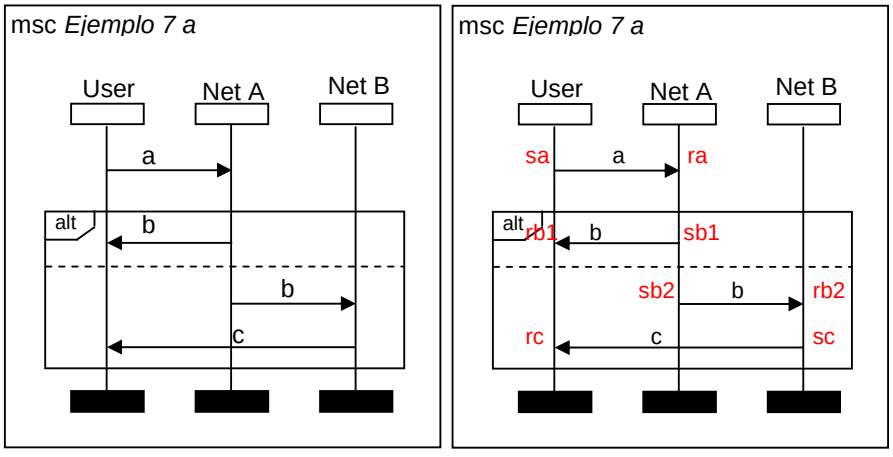


Figura 73: Ejemplo 6 c de MSC - Inicio y reinicio del timer

13) Una expresión ***Alt*** (alternativas) en un gráfico MSC se traduce en tantos escenarios VTS como alternativas ofrezca la sección.

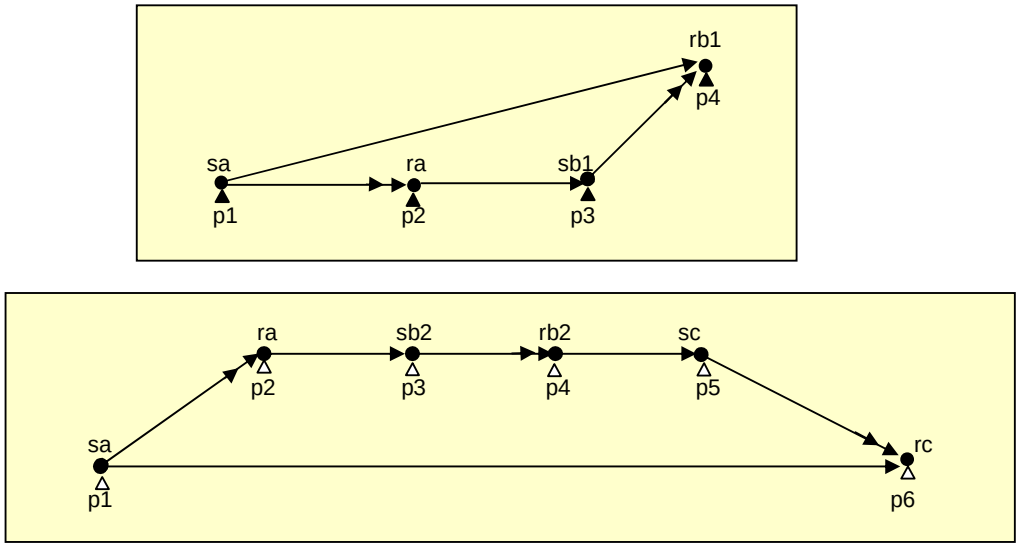
Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):



Donde:
sa = out(User,NetA,a)
ra = in(NetA,User,a)
sb1 = out(NetA,User,b)
rb1 = in(User, NetA,b)
sb2 = out(NetA, NetB,b)
rb2 = in(NetB, NetA,b)
sc = out(NetB, User,c)
rc = in(User, NetB,c)

Figura 74: Ejemplo 7 a de MSC - Alternativas

Se obtienen los siguientes escenarios VTS, cada uno incluyendo una de las alternativas:



En caso de tener una alternativa con otra alternativa anidada se obtendrá un escenario VTS por cada opción alt simple y un escenario por cada alternativa anidada:

Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):

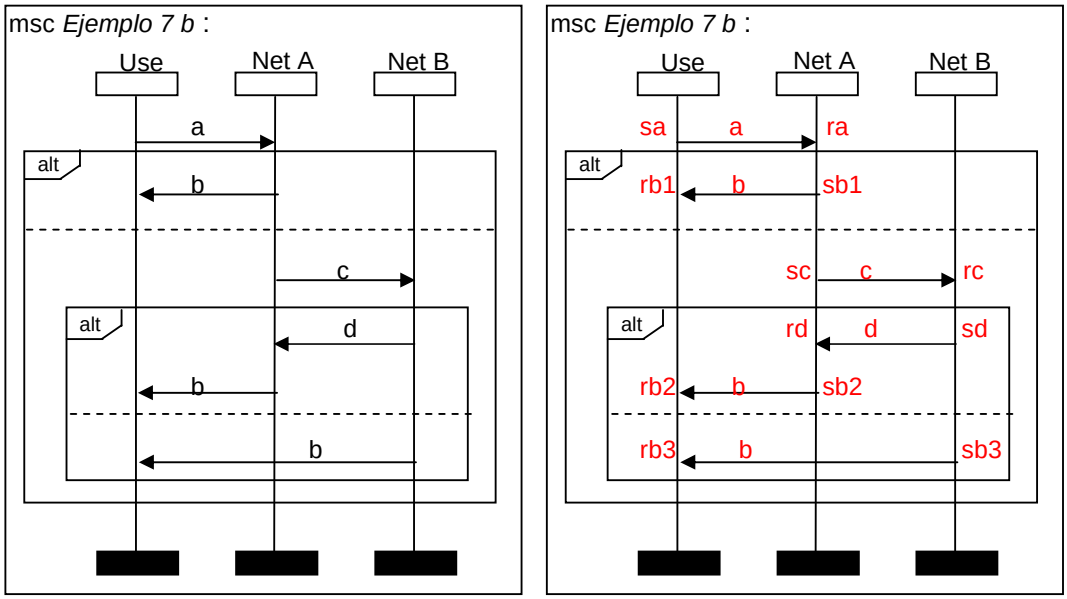
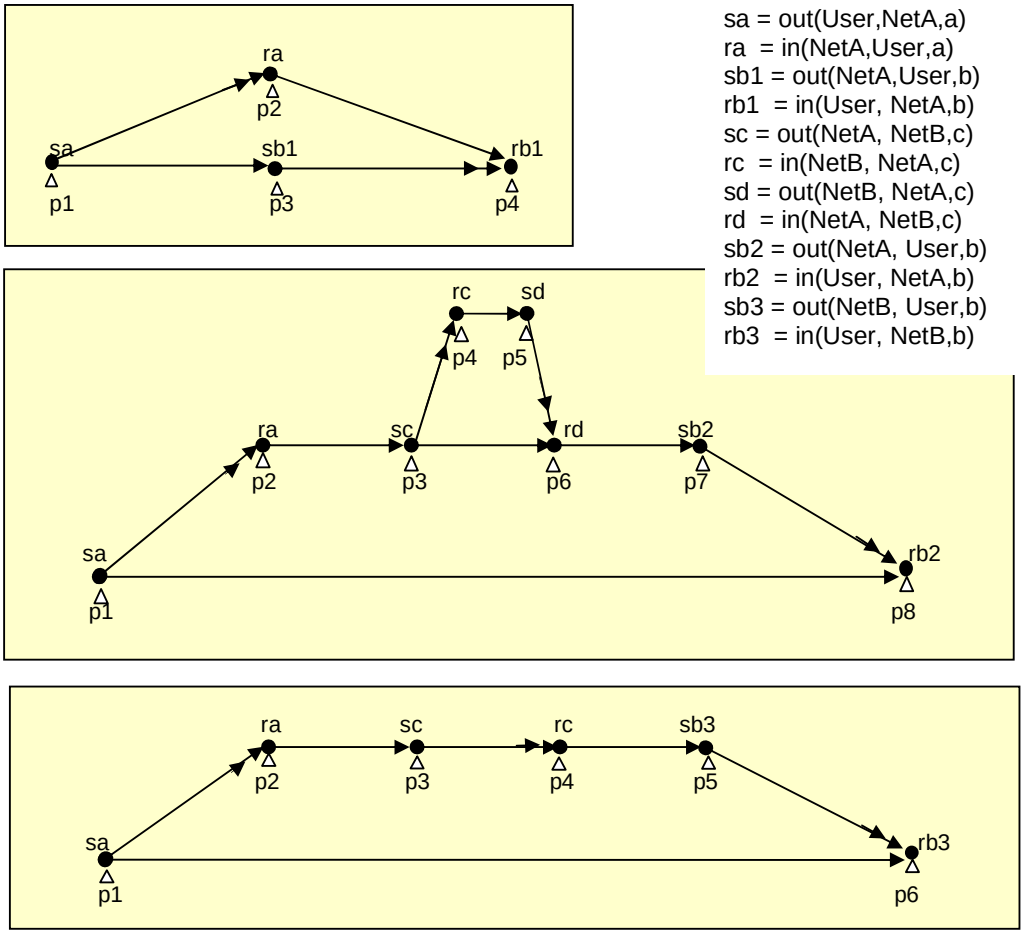


Figura 75: Ejemplo 7 b de MSC - Alternativas anidadas

Se obtienen los siguientes escenarios VTS:



14) Cuando se observa una expresión **Loop** <n, m> (iteraciones) se ejecuta la sección al menos n veces y m veces como máximo en el escenario MSC.

Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):

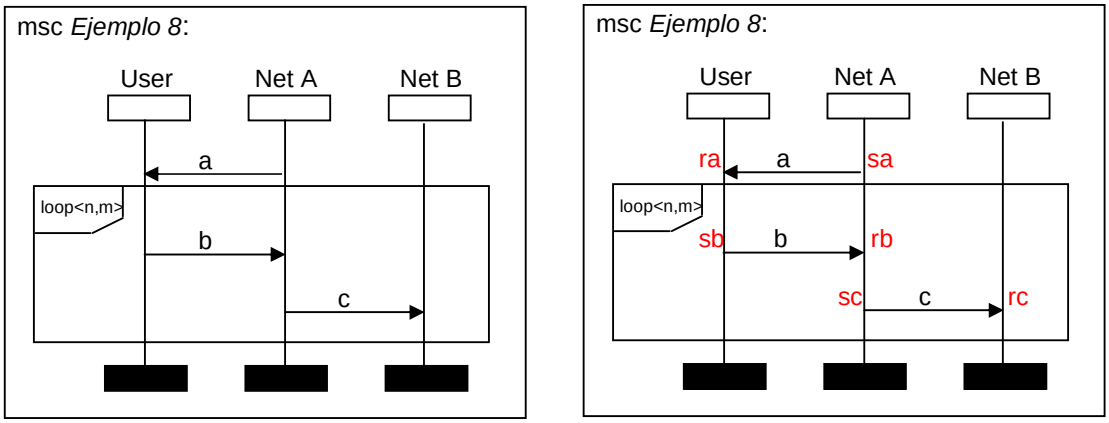
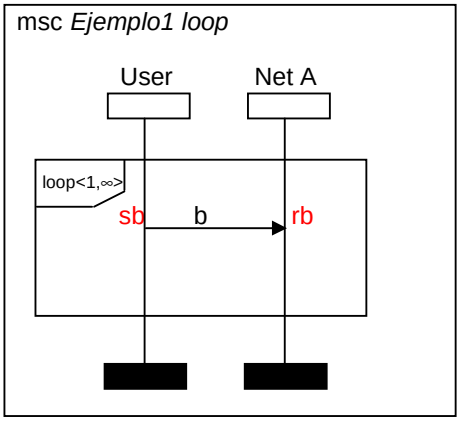


Figura 76: Ejemplo 8 de MSC - Iteraciones

En VTS no se pueden representar loops acotados como el de la Figura 76, es decir, con un mínimo y/o un máximo de iteraciones, ya que no se pueden contar dichas iteraciones.

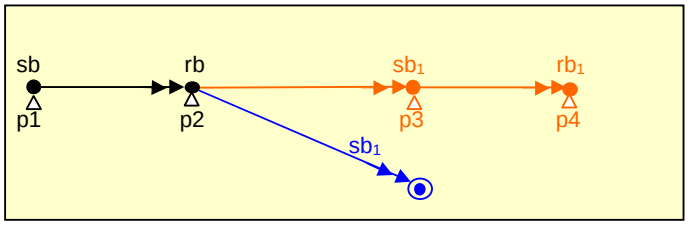
Un loop <1;∞> puede representarse en VTS con escenarios condicionales en los siguientes casos:

- ✓ cuando no haya eventos antes ni después del loop como se muestra en el siguiente gráfico MSC.

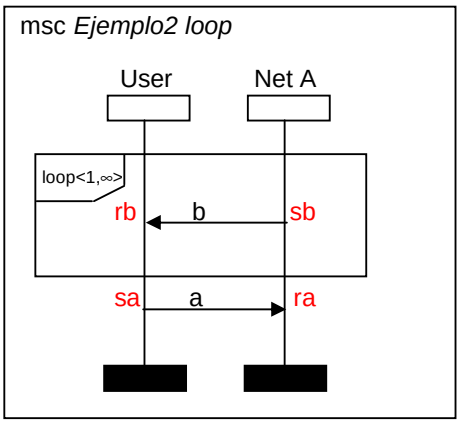


Donde:
sb = out(User, NetA, b)
rb = in(NetA, User, b)

Se obtiene el siguiente escenario VTS:

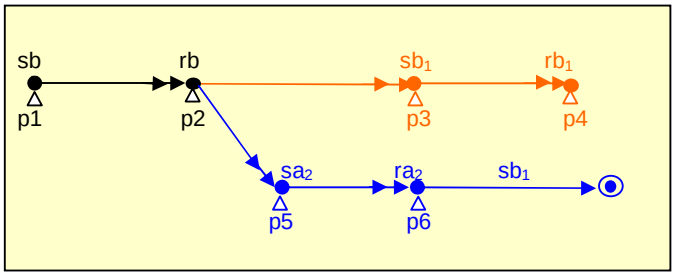


- ✓ cuando no haya eventos antes del loop y no se repitan los eventos del loop a continuación del mismo. El siguiente gráfico MSC sí puede representarse en VTS. Si se observara un envío de un mensaje b desde la instancia Net A hacia la instancia User luego del loop no sería posible expresarlo en VTS.



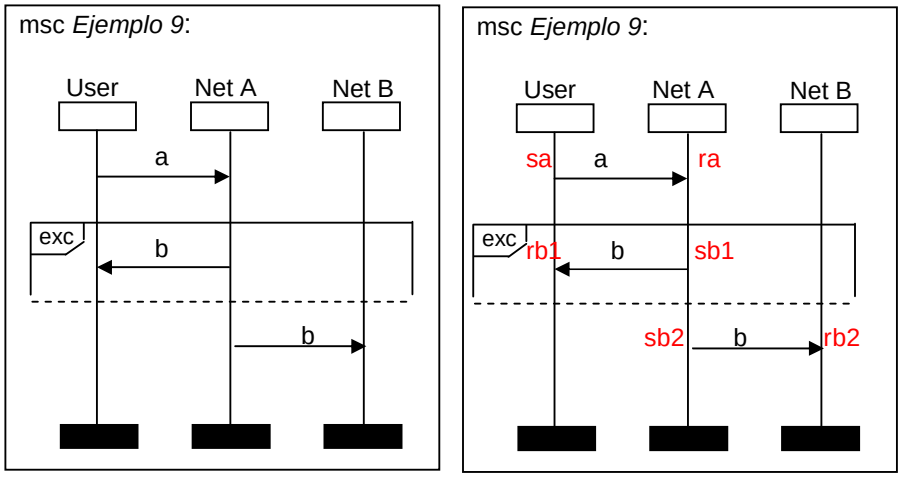
Donde:
sb = out(NetA,User, b)
rb = in(User, NetA, b)
sa = out(User, NetA, a)
ra = in(NetA, User, a)

Se obtiene el siguiente escenario VTS:



15) Una expresión **Exc** (excepción) en un gráfico MSC es equivalente a una expresión alt con una segunda alternativa que es el resto del MSC por lo que se podrá graficar en VTS como dos escenarios, uno que incluye los eventos de la excepción y otro que no (ver Figura 77).

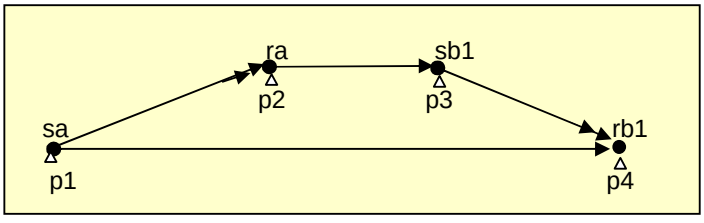
Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):

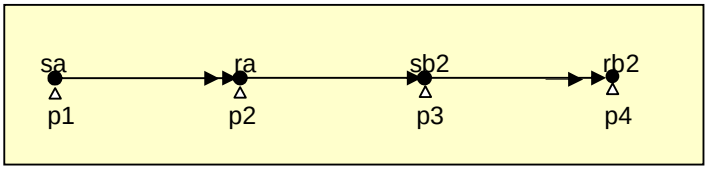


Donde:
sa = out(User,NetA,a)
ra = in(NetA,User,a)
sb1 = out(NetA,User,b)
rb1 = in(User, NetA,b)
sb2 = out(NetA, NetB,b)
rb2 = in(NetB, NetA,b)

Figura 77: Ejemplo 9 de MSC - Excepciones

Se obtienen los siguientes dos escenarios VTS, uno que incluye los eventos de la sección excepción y luego finaliza, y otro que ignora la sección exp.





16) Una expresión **Opt** (opción) en un gráfico MSC es equivalente a una expresión alt con la segunda alternativa vacía por lo que se podrá graficar en VTS como dos escenarios, uno que incluye la opción y otro que no. (ver Figura 78).

Partiendo del siguiente escenario MSC, se identifican los eventos para pasar a VTS (en rojo):

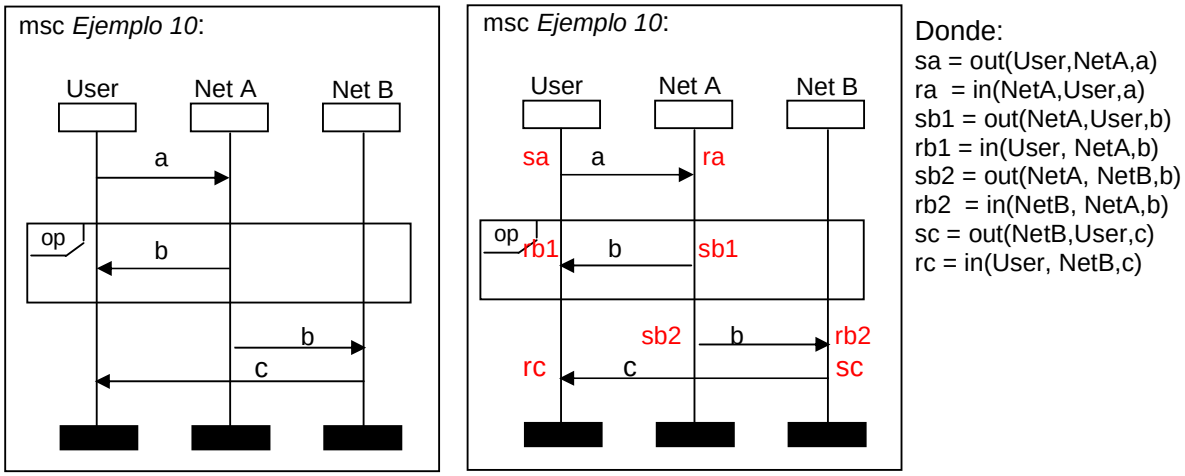
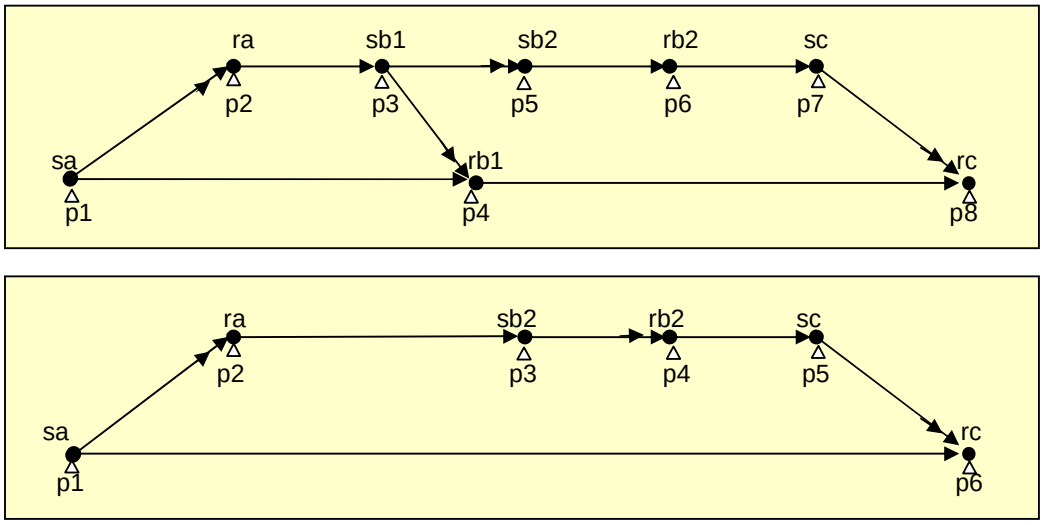


Figura 78: Ejemplo 10 de MSC - Opciones

Se obtienen los siguientes escenarios VTS, uno incluye los eventos optativos y el otro no.



5.1.2 Relación textual

En esta sección se analiza cómo se relacionan textualmente estos formalismos.
Como se expuso en la sección 2.4.3, se tomará el meta lenguaje SDL publicado por ITU en su documento Z.120 como el aspecto textual de MSC. En cuanto a VTS, se tomará el código

XML que se obtiene al graficar un escenario VTS en VISIO con la plantilla correspondiente (ver la sección 2.5.3).

Para pasar de una definición en SDL de un escenario MSC al texto XML que representa a un escenario VTS se tendrán en cuenta las siguientes reglas:

1. **Nombre del escenario**: El nombre del escenario MSC será el mismo para el escenario XML.

2. **Eventos**:
   - ✓ Los eventos ***in***, ***out*** y ***action*** en SDL serán los eventos en XML. Por cada in se obtendrán un evento equivalente send x (***sx***), por cada out un evento equivalente receive x (***rx***) y por cada action un evento equivalente do x (***dox***).
   - ✓ Cuando el que recibe un mensaje es el entorno, se agregará un evento equivalente a un receive. Cuando el que envía el mensaje es el entorno, se agregará un evento equivalente a un send.
   - ✓ Los mensajes perdidos y encontrados serán tratados en XML como los eventos ***in*** y ***out*** de las instancias teniendo en cuenta al medio de comunicación como el que recibe los mensajes perdidos y envía los mensajes encontrados.
   - ✓ Se define un evento en XML cuando se observa un ***set*** timer, ***timeout*** timer o ***reset*** timer en SDL.
   - ✓ Se define un evento en XML cuando se observa un ***create*** instancia (creación de una instancia) o un ***stop*** instancia (finalización de una instancia).

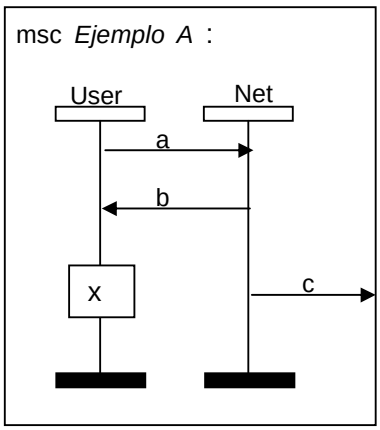
3. **Puntos**: Por cada evento definido según lo expuesto en el punto anterior se define un punto en el texto XML con dicho evento asociado.

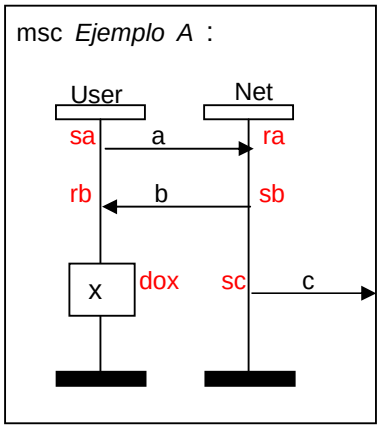
4. **Precedencia**: Teniendo definidos los puntos como se mencionó anteriormente, se tendrán en cuenta las siguientes situaciones para determinar el orden de los eventos:
   - ✓ Después de enviar un mensaje se espera el evento de recepción del mismo. Esto determina el orden horizontal de sucesión de los eventos de un gráfico MSC.
   - ✓ El orden en que se suceden los eventos en las instancias determina el orden vertical de arriba hacia abajo. En caso de no quedar en claro el orden en que se dan los eventos en las instancias se debe determinar si se aplica o no la política FIFO.
   - ✓ Cuando se trata de una co-región, se asume que el evento anterior a esta región en la instancia se debe dar antes que todos los eventos de esta co-región. Esto se refleja en el escenario VTS generando arcos desde el punto del evento anterior a la co-región hacia cada punto asociado a los eventos de la co-región . El primer evento que sucede en la instancia a continuación de la co-región debe darse luego de todos los eventos de la co-región. Esto se refleja en el escenario VTS generando arcos desde cada punto asociado a los eventos de la co-región hacia el punto del siguiente evento en la instancia.

5. **Eventos prohibidos**: Si bien en los gráficos MSC no se observan eventos prohibidos, después de enviar un mensaje se espera que el primer evento de recepción de ese mensaje sea el único que se reciba (en un escenario VTS serían flechas con doble punta).

6. **Restricciones temporales**: Se definirán restricciones de tiempo en XML al definir timers entre los eventos que corresponden a ***set*** y el ***timeout***, ***reset*** o ***stop*** de dicho timer. Se tomará el valor indicado en el SDL o se indicará una restricción variable de <X unidades de tiempo.

Ejemplo 1: MSC básico
Si se onsidera el siguiente gráfco MSC, se pueden detectar los siguientes eventos (marcados en rojo):





Al expresar este gráfico MSC de forma textual, se puede optar por distintas maneras de escribir el texto SDL que define este escenario:

msc ejemplo A ; ***instance*** User ; ***out*** a ***to*** Net ; ***in*** b ***from*** Net ; ***action*** 'c' ; ***endinstance ;*** **instance** Net ; ***in*** a ***from*** User ; ***out*** b ***to*** User ; ***out*** d to env ; ***endinstance ;*** ***endmsc*** ;	***msc*** ejemplo A ; User : ***instance*** ; Net : ***instance*** ; User : ***out*** a ***to*** Net ; Net : ***in*** a ***from*** User ; Net : ***out*** b ***to*** User ; User : ***in*** b ***from*** Net ; User : ***action*** 'c' ; Net : ***out*** d ***to*** env ; User : ***endinstance*** ; Net : ***endinstance*** ; ***endmsc*** ;	***msc*** ejemplo A ; User : ***instance*** ; Net : ***instance*** ; User : ***out*** a ***to*** Net ; User : ***in*** b ***from*** Net ; User : ***action*** 'c' ; Net : ***in*** a ***from*** User ; Net : ***out*** b ***to*** User ; Net : ***out*** d ***to*** env ; User : ***endinstance*** ; Net : ***endinstance*** ; ***endmsc*** ;

Dado que se debe considerar el orden en que se dan los eventos en cada instancia para determinar la precedencia de los eventos en el escenario VTS, se tomará la primera o la última de las propuestas en el cuadro.

En las siguientes tablas se indica entre paréntesis el número de la regla descripta en el punto 5.2 que se aplica a la definición textual del escenario MSC para obtener luego los distintos componentes del escenario VTS y una letra para indicar dónde se refleja en el texto XML que se encuentra a continuación.

En esta primera tabla se aplica la regla 2 para la determinación de los eventos y la regla 3 para la asignación de puntos:

	Eventos **(2)**	Puntos **(3)**
msc ejemplo A ; **(1)**		
User : ***instance*** ;		
Net : ***instance*** ;		
User : ***out*** a ***to*** Net ;	**(2-a)**	**(3-a)**

User : ***in*** b ***from*** Net ;	**(2-d)**	**(3-d)**
User : ***action*** 'c' ;	**(2-e)**	**(3-e)**
Net : ***in*** a ***from*** User ;	**(2-b)**	**(3-b)**
Net : ***out*** b ***to*** User ;	**(2-c)**	**(3-c)**
Net : ***out*** d ***to*** env ;	**(2-f) (2-g)**	**(3-f) (3-g)**
User : ***endinstance*** ;		
Net : ***endinstance*** ;		
endmsc ;		

En la siguiente tabla se aplica la regla 4 para determinar la de precedencia horizontal y vertical entre los puntos y la regla 5 para asignar los eventos prohibidos.

	Precedencia (horizontal) **(4)**	Precedencia (vertical) **(4)**	Eventos prohibidos **(5)**
msc ejemplo A ; **(1)**			
User : ***instance*** ;			
Net : ***instance*** ;			
User : ***out*** a ***to*** Net ;			
User : ***in*** b ***from*** Net ;	**(4-d)**	**(4-c)**	**(5-a)**
User : ***action*** 'c' ;		**(4-e)**	
Net : ***in*** a ***from*** User ;	**(4-a)**		**(5-b)**
Net : ***out*** b ***to*** User ;		**(4-b)**	
Net : ***out*** d ***to*** env ;	**(4-g)**	**(4-f)**	**(5-c)**
User : ***endinstance*** ;			
Net : ***endinstance*** ;			
endmsc ;			

El texto XML que representa un escenario VTS equivalente es el siguiente

```
<scenario id=' ejemplo A'> (1)
        <events> (2)
                <event>sa</event> (2-a)
                <event>ra</event> (2-b)
                <event>sb</event> (2-c)
                <event>rb</event> (2-d)
                <event>doc</event> (2-e)
                <event>sd</event> (2-f)
                <event>rd</event> (2-g)
        </events>
        <points> (3)
                <point id='p1'> (3-a)
                        <event>sa</event>
                </point>
                <point id='p2'> (3-b)
                        <event>ra</event>
                </point>
                <point id='p3'> (3-c)
                        <event>sb</event>
                </point>
                <point id='p4'> (3-d)
```

```
                <event>rb</event>
            </point>
            <point id='p5'> (3-e)
                <event>doc</event>
            </point>
            <point id='p6'> (3-f)
                <event>sd</event>
            </point>
            <point id='p7'> (3-g)
                <event>rd</event>
            </point>
        </points>
        <precedence> (4)
            <precedes marks='next'> (4-a) (5-a)
                <point-ref id='p1'/>
                <point-ref id='p2'/>
            </precedes>
            <precedes> (4-b)
                <point-ref id='p2'/>
                <point-ref id='p3'/>
            </precedes>
            <precedes> (4-c)
                <point-ref id='p1'/>
                <point-ref id='p4'/>
            </precedes>
            <precedes marks='next'> (4-d) (5-b)
                <point-ref id='p3'/>
                <point-ref id='p4'/>
            </precedes>
            <precedes> (4-e)
                <point-ref id='p4'/>
                <point-ref id='p5'/>
            </precedes>
            <precedes> (4-f)
                <point-ref id='p3'/>
                <point-ref id='p6'/>
            </precedes>
            <precedes marks='next'> (4-g) (5-c)
                <point-ref id='p6'/>
                <point-ref id='p7'/>
            </precedes>
        </precedence>
</scenario>
```

Ejemplo 2: Orden de eventos según la aplicación o no de la política FIFO
Teniendo en cuenta el gráfico MSC que se muestra en la Figura 67 se pueden observar dos escenarios distintos dependiendo de si se aplica o no la política FIFO. Las expresiones textuales que se obtienen son las siguientes:

Igual que en el ejemplo anterior, se indica entre paréntesis el número de la regla descripta en el punto 5.2 que se aplica a la definición del escenario MSC para obtener luego los distintos componentes del escenario VTS en XML.

	Eventos **(2)**	Puntos **(3)**
msc ejemplo2b ; **(1)**		
U1 : ***instance*** ;		

U3 : ***instance*** ;		
U2 : ***instance*** ;		
U1 : ***in*** a ***from*** U2 ;	**(2-b)**	**(3-b)**
U1 : ***out*** b ***to*** U3 ;	**(2-c)**	**(3-c)**
U2 : ***out*** a ***to*** U1 ;	**(2-a)**	**(3-a)**
U2 : ***out*** c ***to*** U3 ;	**(2-e)**	**(3-e)**
U2 : ***in*** d ***from*** U3 ;	**(2-h)**	**(3-h)**
U3 : ***in*** c ***from*** U2 ;	**(2-d)**	**(3-d)**
U3 : ***out*** d ***to*** U2 ;	**(2-f)**	**(3-f)**
U3 : ***in*** b ***from*** U1 ;	**(2-g)**	**(3-g)**
U1 : ***endinstance*** ;		
U2 : ***endinstance*** ;		
U3 : ***endinstance*** ;		
endmsc ;		

	Precedencia (horizontal) **(4)**	Precedencia (vertical) **(4)**	Eventos prohibidos **(5)**
msc ejemplo2b ; **(1)**			
U1 : ***instance*** ;			
U3 : ***instance*** ;			
U2 : ***instance*** ;			
U1 : ***in*** a ***from*** U2 ;	**(4-a)**		**(5-a)**
U1 : ***out*** b ***to*** U3 ;		**(4-e)**	
U2 : ***out*** a ***to*** U1 ;			
U2 : ***out*** c ***to*** U3 ;		**(4-f)**	
U2 : ***in*** d ***from*** U3 ;	**(4-d)**	**(4-g)**	**(5-d)**
U3 : ***in*** c ***from*** U2 ;	**(4-b)**		**(5-b)**
U3 : ***out*** d ***to*** U2 ;		**(4-h1) (4-h2)**	
U3 : ***in*** b ***from*** U1 ;	**(4-c)**	**(4-i)**	**(5-c)**
U1 : ***endinstance*** ;			
U2 : ***endinstance*** ;			
U3 : ***endinstance*** ;			
endmsc ;			

Se obtienen dos gráficos VTS distintos según se aplique o no la política FIFO. La definición de los eventos y los puntos es la misma para ambos casos. La diferencia surge al momento de determinar la precedencia de los eventos teniendo en cuenta los eventos en la instancia U3.
Los eventos y los puntos se determinan de la siguiente manera:

<scenario id=' ejemplo2b'> **(1)**
<events> **(2)**
<event>sa</event> **(2-a)**
<event>ra</event> **(2-b)**
<event>sb</event> **(2-c)**
<event>rb</event> **(2-d)**
<event>sc</event> **(2-e)**
<event>rc</event> **(2-f)**
<event>sd</event> **(2-g)**
<event>rd</event> **(2-h)**
</events>

```
<points> (3)
        <point id='p1'> (3-a)
                <event>sa</event>
        </point>
        <point id='p2'> (3-b)
                <event>ra</event>
        </point>
        <point id='p3'> (3-c)
                <event>sb</event>
        </point>
        <point id='p4'> (3-d)
                <event>rb</event>
        </point>
        <point id='p5'> (3-e)
                <event>sc</event>
        </point>
        <point id='p6'> (3-f)
                <event>rc</event>
        </point>
        <point id='p7'> (3-g)
                <event>sd</event>
        </point>
        <point id='p8'> (3-h)
                <event>rd</event>
        </point>
</points>
```

La precedencia entonces será diferente según se aplique o no la política FIFO (texto en color rojo):

XML del escenario VTS obtenido aplicando política FIFO	XML del escenario VTS obtenido sin aplicar política FIFO
<precedence> **(4)** <precedes marks='next'>**(4-a) (5-a)** <point-ref id='p1'/> <point-ref id='p2'/> </precedes> <precedes>**(4-e)** <point-ref id='p2'/> <point-ref id='p3'/> </precedes> <precedes>**(4-f)** <point-ref id='p1'/> <point-ref id='p5'/> </precedes> <precedes marks='next'>**(4-b)(5-b)** <point-ref id='p5'/> <point-ref id='p6'/> </precedes> <precedes>**(4-g)** <point-ref id='p5'/> <point-ref id='p8'/> </precedes> <precedes marks='next'>**(4-c) (5-c)** <point-ref id='p3'/> <point-ref id='p4'/> </precedes> <precedes>**(4-h1)** <point-ref id='p4'/>	<precedence> **(4)** <precedes marks='next'>**(4-a) (5-a)** <point-ref id='p1'/> <point-ref id='p2'/> </precedes> <precedes>**(4-e)** <point-ref id='p2'/> <point-ref id='p3'/> </precedes> <precedes>**(4-f)** <point-ref id='p1'/> <point-ref id='p5'/> </precedes> <precedes marks='next'>**(4-b) (5-b)** <point-ref id='p5'/> <point-ref id='p6'/> </precedes> <precedes>**(4-g)** <point-ref id='p5'/> <point-ref id='p8'/> </precedes> <precedes marks='next'>**(4-c) (5-c)** <point-ref id='p3'/> <point-ref id='p4'/> </precedes> <precedes>**(4-h2)** <point-ref id='p4'/>

```
            <point-ref id='p6'/>
        </precedes>
        <precedes>(4-i)
            <point-ref id='p6'/>
            <point-ref id='p7'/>
        </precedes>
        <precedes marks='next'>(4-d)(5-d)
            <point-ref id='p7'/>
            <point-ref id='p8'/>
        </precedes>
    </precedence>
</scenario>
```

```
            <point-ref id='p7'/>
        </precedes>
        <precedes>(4-i)
            <point-ref id='p6'/>
            <point-ref id='p7'/>
        </precedes>
        <precedes marks='next'>(4-d) (5-d)
            <point-ref id='p7'/>
            <point-ref id='p8'/>
        </precedes>
    </precedence>
</scenario>
```

Ejemplo 3:Co-región

En el caso que se muestra en la Figura 68 se observa una co-región en la instancia User. Teniendo en cuenta el texto SDL correspondiente al escenario se han asignado puntos siguiendo el esquema del escenario VTS obtenido en el ejemplo a partir de dicha figura.

msc ejemplo3 ;	Asignación de puntos
User : ***instance*** ;	
Net : ***instance*** ;	
User : ***out*** a ***to*** Net ;	p1
User : ***concurrent*** ;	
in b ***from*** Net ;	p4
in c ***from*** Net ;	p6
endconcurrent ;	
User : ***out*** d ***to*** Net ;	p7
Net : ***in*** a ***from*** User ;	p2
Net : ***out*** b ***to*** User ;	p3
Net : ***out*** c ***to*** User ;	p5
Net : ***in*** d ***from*** User ;	p8
User : ***endinstance*** ;	
Net : ***endinstance***;	
endmsc ;	

La precedencia en este caso sigue observando el orden horizontal (relacionado con los mensajes recibidos) y el orden vertical de la misma manera que en los casos anteriores salvo en el segmento de la co-región.

Para determinar la precedencia en este segmento se tendrá en cuenta lo explicado para este caso en la regla 4: el evento anterior a la co-región en SDL es User:***out*** a ***to*** Net asignado en este caso al punto 1, y el evento que sigue a la co-región es User:***out*** d ***to*** Net asignado al punto 7.

Se determina que el evento anterior a la co-región debe darse antes que los eventos de esta co-región por lo que determina que p1<p4 y p1<p6.

Además cada evento de la co-región debe darse antes que el evento que sucede en la instancia a continuación de la co-región. Esto determina que p4<p7 y p6<p7.

Por lo tanto las precedencias correspondientes a la zona de co-región para el escenario VTS son: (p1, p4), (p1, p6), (p4, p7) y (p6, p7).

Ejemplo 4: Comienzo y finalización de una instancia
Cuando se observa el comienzo y la finalización de una instancia en un gráfico MSC como se muestra en la Figura 69, se transcribe en SDL como un ***create*** y un ***stop*** asociados a dicha instancia.
En VTS simplemente se toma el comienzo y la finalización de una instancia como eventos que se asignan a puntos.

Ejemplo 5: Timers
Cuando se observan definiciones de timers en un gráfico MSC (como se muestra en Figura 71 y Figura 72) que incluyen declaraciones del tipo ***set*** timer, ***timeout*** timer o ***reset*** timer en SDL, se representan en VTS simplemente como eventos asignados a puntos.

Ejemplo 6: Condiciones
Cuando se observan alternativas, excepciones u opciones en un gráfico MSC, se traducen en varios escenarios VTS según estas condiciones.
Una forma sencilla de obtener estos escenarios es reescribiendo el texto SDL de las distintas posibilidades que ofrecen las condiciones ***alt***, ***exc*** y ***opc***. De esta forma se obtendrán MSC básicos y se podrán aplicar las reglas vistas en el punto 5.1.2 para derivar estos textos SDL al texto XML que representa un escenario VTS.

Por ejemplo, partiendo del siguiente texto SDL correspondiente al escenario de la Figura 74

```
msc ejemplo7a ;
    User  : instance ;
    Net A: instance ;
    Net B: instance ;
        User : out a to Net A;
        Net A: in a from User ;
        User, Net A, Net B: alt begin ;
                            Net A: out b to User ;
                            User  : in b from Net A ;
                alt ;
                            Net A: out b to Net B ;
                            Net B: in b from Net A ;
                            Net B: out c to User;
                            User  : in c from Net B ;
                alt end;
    User  : endinstance ;
    Net A: endinstance ;
    Net B: endinstance ;
endmsc ;
```

se pueden asumir dos gráficos MSC distintos, ***msc*** ejemplo7a 1 y ***msc*** ejemplo7a 2, cada uno tomando una parte del ***alt***. A continuación se transcriben los textos SDL que representan estos dos gráficos MSC.

El texto XML que expresa estos escenarios VTS se obtiene de la misma manera que en los casos anteriores según la definición de eventos, de puntos, precedencia y prohibidos explicados en el punto 5.1.2.

```
msc ejemplo7a 1;
    User  : instance ;
    Net A: instance ;
    Net B: instance ;
        User : out a to Net A;
        Net A: in a from User ;
        Net A: out b to User ;
        User  : in b from Net A ;
    User  : endinstance ;
    Net A: endinstance ;
    Net B: endinstance ;
endmsc ;
```

```
msc ejemplo7a 2;
    User  : instance ;
    Net A: instance ;
    Net B: instance ;
        User : out a to Net A;
        Net A: in a from User ;
        Net A: out b to Net B ;
        Net B: in b from Net A ;
        Net B: out c to User;
        User  : in c from Net B ;
    User  : endinstance ;
    Net A: endinstance ;
    Net B: endinstance ;
endmsc ;
```

5.1.3 Relación formal

Como se mencionó en el punto 2.4.4 del capítulo 2, se optó por la definición formal de MSC propuesta por A. Muscholl y D. Peled [MP00]. También cabe recordar que esta definición corresponde a un escenario básico de MSC donde no se definen co-regiones, inicio y finalización de instancias, timers ni escenarios condicionales como los que surgen al tener en cuenta las expresiones de alternativas, iteraciones, excepciones u opciones.

El escenario $\boldsymbol{S}_{VTS}=\langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$ que resulta a partir de un escenario $\boldsymbol{S}_{\text{MSC}}=\langle V, <, P, \mathcal{N}, L, T, N, m \rangle$ se obtiene de la siguiente manera:	
Σ	• Se relaciona con V. Son el mismo conjunto.
P	• La cantidad de puntos será igual a la cantidad de elementos de Σ (eventos).
ℓ	• Simplemente se asocia un evento a cada punto.
$\not\equiv$	No tiene su similar en MSC. Por lo tanto resulta =∅. Aunque se observen el mismo evento (ver Figura 64 y Figura 65) estos estarán asignados a distintos puntos dado que existe una relación de precedencia vertical en la instancia.
$<$	• Tiene su similar en $<$ del escenario MSC. Pero en este punto hay que determinar si se aplica siempre la política FIFO o no cuando en una instancia se reciben mensajes de distintos origenes. Es decir, en caso de encontrar dos eventos **f** y **h** tales que: T **(f) =** T **(h) =r**, L **(f) =** L **(h)** y m **(f) ≠** m **(h)** (como se detalló al comienzo de este capítulo) entonces se puede decidir aplicar o no política FIFO. • Si se aplica política FIFO se relaciona con < de MSC cambiando los nombres de los eventos por los puntos a los que están asociados. • Si no se aplica política FIFO es equivalente a decir que estos eventos pertenecen a una corregión donde no se puede establecer el orden en que se darán. Dado que en esta descripción formal no se incluye la definición de co-regiones se debe realizar lo siguiente: ✓ No tener en cuenta el par del conjunto < de la definición de MSC que indica

	el orden entre estos eventos de la misma instancia. ✓ Se agregarán al conjunto < de VTS pares ***(p,q)*** con ***p*** igual punto asociado al evento anterior a esta co-región en la instancia y ***q*** igual a cada punto asociado a los eventos de esta co-región. ✓ Se agregarán al conjunto < de VTS pares ***(p,q)*** con ***p*** igual a cada punto asociado a los eventos de esta co-región y ***q*** igual al punto asociado al evento siguiente a esta co-región en la instancia.
$<_F$	No tiene su similar en MSC. Por lo tanto resulta =∅.
$<_L$	No tiene su similar en MSC. Por lo tanto resulta = ∅.
γ	• En los gráficos MSc no se muestran eventos prohibidos entre los envíos o recepción de los mensajes. Pero γ se relaciona con los eventos ***receive*** ya que los eventos de recepción de mensaje serán los primeros eventos de recepción de esos mensajes que ocurran desde el envío de dichos mensajes. Esto se representarán con flechas dobles en el escenario VTS. Formalmente: ✓ se toman de m las relaciones s $\rightarrow$ r, ✓ se identifica el par ***(p,q)*** con ***p*** igual al punto asociado al evento de s y ***q*** igual al punto que tiene asociado este evento r y ✓ se les asigna el evento r como evento prohibido para asegurar sea el primero que ocurrió desde el envío de dicho mensaje.
δ	• En la definición formal adoptada no se ven definidos los timers. La forma de incorporar las restricciones de tiempo impuestas por los timers es por observación del gráfico de escenario MSC.

Ejemplo 1: pasando de MSC a VTS conociendo sólo la definición formal sin problemas en el orden de los eventos.

Si se toma la Figura 63 se tiene la siguiente definición formal del escenario MSC:

$\boldsymbol{S}_{MSC}$ ejemplo1.c = $\langle V, <, P, \mathcal{N}, L, T, N, m \rangle$

V = {sa, ra, sb, rb, sc, rc, dox}

$<$ = {(sa, ra), (sb, rb), (sc, rc), (sa, rb), (rb, sc), (ra, sb), (sb, rc), (rc, dox)}

P = {User, Net}

$\mathcal{N}$ = {a, b, c, x}

L = {sa → User , ra → Net , sb → Net , rb → User , sc → User , rc → Net, dox → Net}

T = {sa → s, ra → r, sb → s, rb → r, sc → s, rc → r, dox → l}

N = {sa → a, ra → a, sb → b, rb → b, sc → c, rc → c, dox → x}

m = {sa → ra, ra → sa, sb → rb, rb → sb, sc → rc, rc → sc}

El escenario VTS que se obtiene es el siguiente:

$\boldsymbol{S}_{VTS}$ *ejemplo1.c* $= \langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$
Σ = { sa, ra , sb , rb , sc , rc , dox}
P = {p1 ; p2 ; p3 ; p4 ; p5 ; p6 ; p7 }
ℓ = { p1→{sa}; p2→{ra}; p3→{sb}; p4→{rb} ; p5→{sc} ; p6→{rc} ; p7→{dox} }
$\not\equiv$ = $\varnothing$
$<$ = { (p1, p2) ; (p3, p4) ; (p5, p6) ; (p1, p4) ; (p4, p5) ; (p2, p3) ; (p3, p6) ; (p6, p7) }
$<_F$ =$\varnothing$
$<_L$ = $\varnothing$
γ = { (p1, p2) →{ra } ; (p3, p4) →{rb} ; (p5, p6) →{rc} }
δ =$\varnothing$

Ejemplo 2: pasando de MSC a VTS conociendo sólo la definición formal con problemas en el orden de los eventos.

Si se toma la Figura 67 se tiene la siguiente definición formal del escenario MSC:

$\boldsymbol{S}_{\text{MSC}}$ ejemplo2.b $= \langle V, <, P, \mathcal{N}, L, T, N, m \rangle$

V = {sa, ra, sb, rb, sc, rc, sd, rd}

$<$ = {(sa, ra), (sb, rb), (sc, rc), (sd, rd), (ra, sb), (sa, sc), (sc, rd), (rb, rc), (rc, sd)}

P = {U1, U2, U3}

$\mathcal{N}$ = {a, b, c, d}

L = {sa → U2 , ra → U1 , sb → U1 , rb → U3 , sc → U2 , rc → U3 , sd → U3, rd → U2}

T = { sa → s , ra → r , sb → s , rb → r , sc → s , rc → r , sd → s , rd → r }

N = {sa → a, ra → a, sb → b, rb → b, sc → c, rc → c, sd → d, rd → d}

m = {sa → ra, ra → sa, sb → rb, rb → sb, sc → rc, rc → sc, sd → rd, rd → sd}

El escenario VTS que se obtiene es el siguiente:

$\boldsymbol{S}_{VTS}$ *ejemplo1.c* $= \langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$
Σ = {sa, ra, sb, rb, sc, rc, sd, rd}
P = {p1 ; p2 ; p3 ; p4 ; p5 ; p6 ; p7; p8 }
ℓ = {p1→{sa}; p2→{ra}; p3→{sb}; p4→{rb}; p5→{sc}; p6→{rc}; p7→{sd}; p8→{rd}}
$\not\equiv = \varnothing$
$<$ = { (p1, p2); (p2, p3); (p3, p4); (p1, p5); (p5, p6); (p7, p8); (p5, p8) ; (p4, p6) ; (p6, p7) }
$<_F = \varnothing$
$<_L = \varnothing$
γ = {(p1, p2) →{ra }; (p3, p4) →{rb}; (p5, p6) →{rc}; (p7, p8) →{rd}}
$\delta = \varnothing$

Pero si se observan los elementos de L correspondientes a **U3**, se puede observar que ***rb*** y ***rc*** son eventos de recepción de mensajes enviados por distintos procesos o instancias. O sea que se cumple que T **(rb) =** T **(rc) =r**, L **(rb) =** L **(rc) = U3** y L **(*m* (rb)) ≠** L **(*m* (rc)).** Entonces se podría obtener otro escenario VTS sin aplicar la política FIFO. En este escenario VTS lo que cambia es $<$ de la siguiente manera:

- ✓ se descarta el par ***(rb, rc)*** de $<$ de MSC que representa el orden entre estos dos eventos de la instancia U3, por lo que no estará el par ***(p4, p6)*** en $<$ de VTS,
- ✓ se deben agregan flechas desde cada uno de estos puntos al punto del evento siguiente en dicha instancia. Los eventos ***e*** tales que $L(e) = U3$ son ***rb***, ***rc*** y ***sd***, lo que nos queda ***sd*** como evento siguiente a ***rb*** y ***rc***. Sabiendo que el punto asociado a ***sd*** es ***p7***, se agregarán a $<$ de VTS, en este caso, sólo el par ***(p4, p7)*** ya que también se debería agregar ***(p6, p7)*** pero ya estaba anteriormente. Entonces queda el siguiente escenario:

S_{VTS} *ejemplo2.b* $= \langle \Sigma, P, \ell, \not\equiv, <, <_F, <_L, \gamma, \delta \rangle$

Σ = {sa, ra, sb, rb, sc, rc, sd, rd}
P = {p1; p2; p3; p4; p5; p6; p7; p8}
ℓ = {p1→{sa}; p2→{ra}; p3→{sb}; p4→{rb}; p5→{sc}; p6→{rc}; p7→{sd}; p8→{rd}}
$\not\equiv = \varnothing$
$<$ = { (p1, p2) ; (p3, p4) ; (p5, p6) ; (p7, p8) ; (p2, p3) ; (p1, p5) ; (p5, p8) ; (p4, p7) ; (p6, p7) }
$<_F = \varnothing$
$<_L = \varnothing$
γ = {(p1, p2) →{ra}; (p3, p4) →{rb}; (p5, p6) →{rc}, (p7, p8) →{rd}}
$\delta = \varnothing$

5.2 Ventajas y desventajas

Después de analizar distintos ejemplos y casos posibles, se puede decir que en general se puede expresar con VTS la mayoría de los escenarios MSC. En cambio no todo lo que se puede hacer en un escenario VTS se lo puede mostrar en un gráfico MSC.

Se han observado las siguientes desventajas en los resultados de MSC con respecto a VTS:

- En VTS se puede indicar la ocurrencia de un evento en un punto indicando que es el primero de esos eventos que ocurre. En MSC sólo los eventos asociados a la recepción de un mensaje cumplirían con este criterio.
- En VTS podemos tener más de un evento (conjunto) asociado a un punto. Esto indica que ocurrirá uno de esos eventos asociado. En MSC se asocia un determinado evento a un instante en una instancia por lo que se necesitan más de un gráfico MSC para representar este caso.
- En VTS se puede tener restricciones temporarias en todos sus aspectos (gráfico, XML y formal), en MSC se pueden definir timers en los gráficos y en SDL, no así formalmente cuando se trata de un bMSC.
- En VTS se puede enfocar en el primer o último evento que ocurra dentro de un grupo. Esta facilidad no se observa en MSC. En MSC sería más de un gráfico o un gráfico bastante complejo con distintas expresiones de alternativas según suceda cada primero o último.
- En VTS se puede indicar eventos prohibidos entre la ocurrencia de dos eventos. Esto no se observa en MSC.

También se ha observado la siguiente desventaja en los resultados de VTS con respecto a MSC:

- En MSC se pueden graficar procesos iterativos de una manera muy sencilla. En cambio en VTS no siempre se pueden expresar iteraciones. Se pueden conseguir iteraciones en VTS con antecedentes y consecuentes pero para poder hacerlo se deben cumplir ciertas condiciones.

Capítulo 6: Comparación cualitativa entre TE, MSC y VTS

6.1 Comparando los lenguajes

6.1.1 Amigabilidad: Facilidad para entender el lenguaje

- **TE**

Es el lenguaje más “amigable” de los tres. Esto se debe en gran parte a que su expresividad se limita a escenarios muy simples, como una sucesión de eventos en el tiempo. Hay que recordar que este lenguaje gráfico fue creado específicamente para definir propiedades sin tener que involucrarse con el aspecto formal de lenguajes existentes. Como consecuencia de esto se observa que es un lenguaje fácil de utilizar. Una vez que se conocen sus elementos y qué representan, se puede interpretar sin dificultad el escenario representado por un gráfico TE.
No se pueden evaluar los aspectos textuales ni formales ya que TE es un lenguaje puramente gráfico y no ofrece estas facilidades.

- **MSC**

Al trabajar con gráficos MSC hay que tener en cuenta la complejidad del escenario que se está describiendo. Si se trata de un MSC básico, no se necesitan conocimientos previos para poder interpretar un gráfico MSC. En cambio si se incorporan elementos como eventos perdidos y encontrados, co-regiones, condicionales y timers, se necesita conocer cómo se dibujan y qué representan para poder comprender el escenario graficado. Estos conocimientos previos deben ser ampliados más aún si se trata de un HMSC.
Si se considera la definición textual de un escenario MSC, se puede decir que si bien no es difícil de entender, es necesario el estudio previo del documento Z.120 de ITU para comprender la gramática y la definición de los elementos del SDL.
Generalmente entender las definiciones formales presupone un entendimiento previo y no suelen ser intuitivas o sencillas de interpretar. Teniendo en cuenta que se han analizado distintos trabajos donde se proponían diferentes formalizaciones de MSC y ninguna presentaba una definición completa (sólo expresan un bMSC), podemos considerar que este aspecto no sólo no es muy “amigable” sino que tampoco está completo.

- **VTS**

Si bien para poder entender un escenario VTS es necesario un conocimiento previo de los elementos que se pueden incorporar y qué significan en la representación de un escenario, la interpretación de un escenario VTS es muy intuitiva, incluso en escenarios condicionales donde se plantean en un mismo gráfico distintos escenarios posibles que tienen una parte de sus trazas en común (antecedentes) y otras partes de trazas distintas (consecuentes) que deberán observarse para cada subescenario.
Debido a que la forma de expresar un escenario VTS en forma de texto se realiza en XML, basta con comprender su gramática general y la forma de definir los elementos específicos del escenario VTS.

En cuanto a su parte formal, VTS cuenta con una descripción completa. Pero como se mencionó anteriormente, las formalizaciones en general no son fáciles de entender sin conocimientos previo y en el caso de VTS la interpretación del escenario a partir de su descripción formal no es intuitiva ni fácil de interpretar sin estos conocimientos.

6.1.2 Expresividad: Facilidad para mostrar distintas propiedades del escenario

- **TE**

Al ser el leguaje más sencillo de los tres, hace que también sea el más limitado para mostrar distintas variantes de un escenario al momento de expresarlo gráficamente. Este lenguaje fue pensado para mostrar de una manera muy simple cómo deben sucederse los eventos en el tiempo.
Por ese motivo, este lenguaje no permite situaciones sencillas y muy comunes como la observación de eventos sin un orden estricto entre ellos, antes de un evento determinado (lo que se define como una co-región en MSC o dos flechas o más que llegan a un punto en VTS). Tampoco se pueden expresar condiciones ni restricciones de tiempo. Es el menos expresivo de los tres lenguajes analizados.

- **MSC**

Si bien este lenguaje fue desarrollado específicamente para representar escenarios de problemas de comunicación, permite graficar una gran variedad de situaciones incluso variantes de condiciones a través de las in-line expressions. Un aspecto que no muestra MSC, y sí lo hacen TE y VTS, es la posibilidad de indicar eventos que no son aceptados o son prohibidos entre la ocurrencia de otros eventos.

- **VTS**

Es el más expresivo de los tres. Permite plasmar casi todas las opciones que ofrece MSC a excepción de condiciones cíclicas de tipo “loop” donde se indica una cantidad máxima y mínima de repeticiones. VTS permite ciclos infinitos utilizando escenarios condicionales.
Además incorpora la posibilidad de considerar el primero o último evento dentro de un conjunto y los escenarios condicionales que permiten mostrar distintos escenarios que comparten una sección de sus trazas en un único gráfico.
VTS también posibilita expresar ordenes parciales entre los eventos mientras que en MSC no es tan completa (no siempre se puede determinar el orden de los eventos, se pueden obtener distintos escenarios según se aplique o no la política FIFO) y en TE directamente no existe ya que se trata de un gráfico donde los eventos suceden en un cierto orden en una línea de tiempo.
En VTS se pueden expresar restricciones temporales (en MSC se las indican con timers mientras que no son posibles en TE) y también eventos prohibidos (en TE se pueden indicar restricciones entre las marcas mientras que en MSC no es posible).

6.1.3 Popularidad: Cuánto se lo utiliza y en qué ámbito

- **TE**

Debido a que fue desarrollado para un problema específico, y teniendo en cuenta los escasos ejemplos de uso publicados, no se puede considerar que sea un lenguaje muy conocido tanto en el ámbito industrial como en el académico.

- **MSC**

Es el lenguaje gráfico más conocido y utilizado de los tres. Es muy utilizado para especificar escenarios relacionados con sistemas de comunicación en el ámbito industrial.

- **VTS**

Este lenguaje fue desarrollado en el ámbito académico y si bien tiene asociado varias traducciones que permiten utilizar varios verificadores, por ejemplo a autómatas de tiempo o redes de Petri, su uso aún no se ha extendido de forma considerable en el ámbito industrial.

6.2 Comparando las herramientas

6.2.1 Disponibilidad: Facilidad para acceder al editor o graficador

- **TE**

Cuando se comenzó este trabajo se pudo acceder a una versión limitada del editor desde un sitio web luego de solicitar autorización para su uso (ver punto 2.3.2.1 del capítulo 2). Actualmente esta herramienta no está accesible de forma gratuita.

- **MSC**

Al ser este lenguaje muy utilizado en la industria, se han desarrollado distintas herramientas que permiten realizar gráficos MSC (ver punto 2.4.2.1 del capítulo 2). Al momento de realizar este trabajo no se encontró un editor o graficador MSC gratuito.
Como se mencionó en el capítulo 2 (ver punto 2.4.2.1), existe una plantilla para VISIO que se puede obtener de forma gratuita. Dado que se debe tener el VISIO (que no es de distribución gratuita) para poder graficar un escenario MSC con esta plantilla, se puede asumir que no es posible acceder a un graficador MSC totalmente gratuito.

- **VTS**

Como se mencionó en el capítulo 2 (ver punto 2.5.2.1), existe una plantilla para VISIO que se puede obtener de forma gratuita (http://lafhis.dc.uba.ar/vts/). También se puede obtener un graficador de VTS básico como plugin de Eclipse en http://lafhis.dc.uba.ar/vintime .

6.2.2 Utilización: Dificultad en el uso de la herramienta

- **TE**

El editor presentado por los laboratorios Bell es muy simple de usar.
Su instalación no requiere de grandes requerimientos de hardware ni de software. Para este trabajo se instaló la herramienta bajo Windows XP y luego en Windows 7 sin problemas.
Leyendo la ayuda o guía que acompaña la herramienta se pueden conocer los elementos básicos de TE y utilizar esta herramienta fácilmente.

- **MSC**

Dado que existen distintas herramientas en el mercado, no se puede generalizar ni indicar cuál es más simple o más compleja en su uso, ni los requerimientos de instalación.
Para este trabajo se instaló la planilla de MSC para VISIO bajo Windows XP sin problema.

- **VTS**

Una vez descargada la plantilla del sitio correspondiente y siguiendo las indicaciones que allí se enumeran, no hay inconvenientes para su instalación. Para este trabajo se instaló la herramienta bajo Windows XP sin problemas.
Teniendo conocimientos previos de los distintos elementos que se pueden integrar en un escenario VTS, el uso del editor es simple e intuitivo. Se van agregando los componentes del escenario que se desea representar como en cualquier dibujo de VISIO.

6.2.3 Flexibilidad: Distintas salidas que permite la herramienta

- **TE**

El editor TE genera un archivo de tipo VIP. Si se desea guardar la imagen del gráfico se puede guardar como un archivo .GIF o .PS.
También se puede obtener el gráfico del autómata que representa al escenario (facilidad del editor) en un archivo con extensión AUT. Si bien representa a un autómata, este archivo no guarda la imagen del autómata sino que es editable con cualquier editor de texto y contiene una descripción del mismo en PROMELA para luego utilizar SPIN para la verificación de los requerimientos (ver punto 3.1.1).

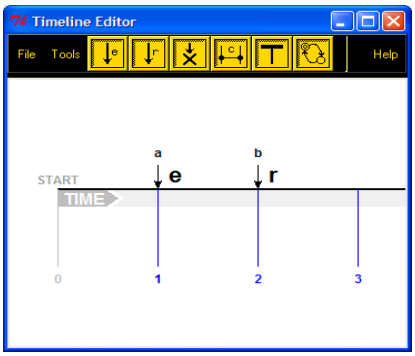


Figura 79: Ejemplo de archivo VIP generado con el editor TE

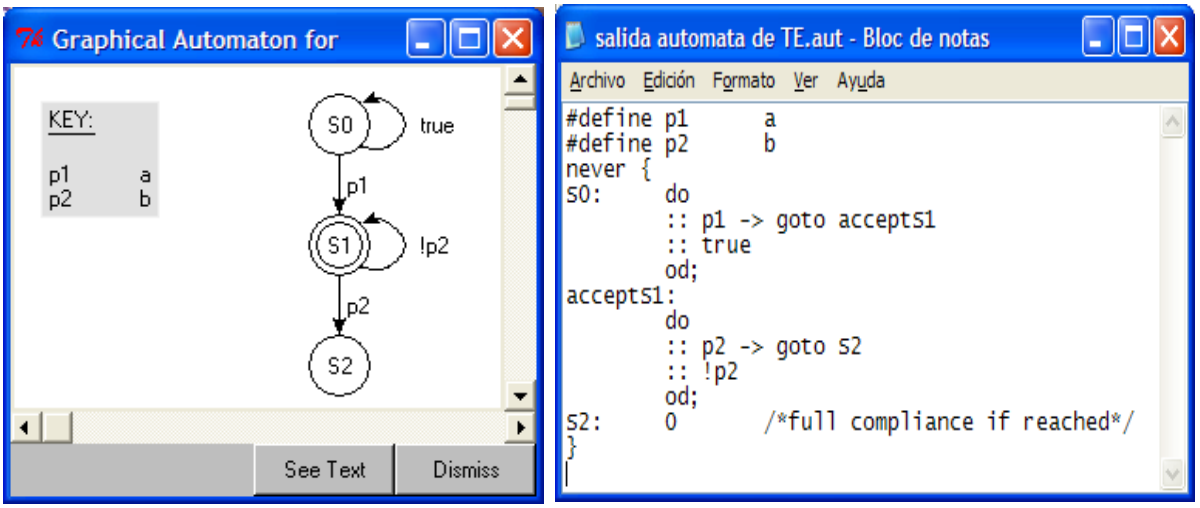


Figura 80: Ejemplo de autómata que se obtiene con el gráfico de la figura 1 y el archivo AUT generado con el editor TE

- **MSC**

Dado que existen distintas herramientas en el mercado, no se puede generalizar ni indicar cuáles son las distintas salidas que permiten cada herramienta.
Si tomamos la plantilla para VISIO que permite generar gráficos, se puede obtener un archivo VSD como así también los distintos tipos de archivos que permite este software como .gif, .jpg, .bmp, etc.
Un tipo de salida que permite el VISIO es un archivo VDX. A diferencia del archivo que se obtiene con la plantilla para VISIO para realizar gráficos VTS, este archivo si bien es editable

por cualquier editor de texto, no permite identificar claramente cómo se compone el grafico MSC.
Cabe recordar, como se menciona en el capítulo 3, que hay distintos trabajos que presentan traducciones de gráficos MSC a redes de Petri, CFM y UML.

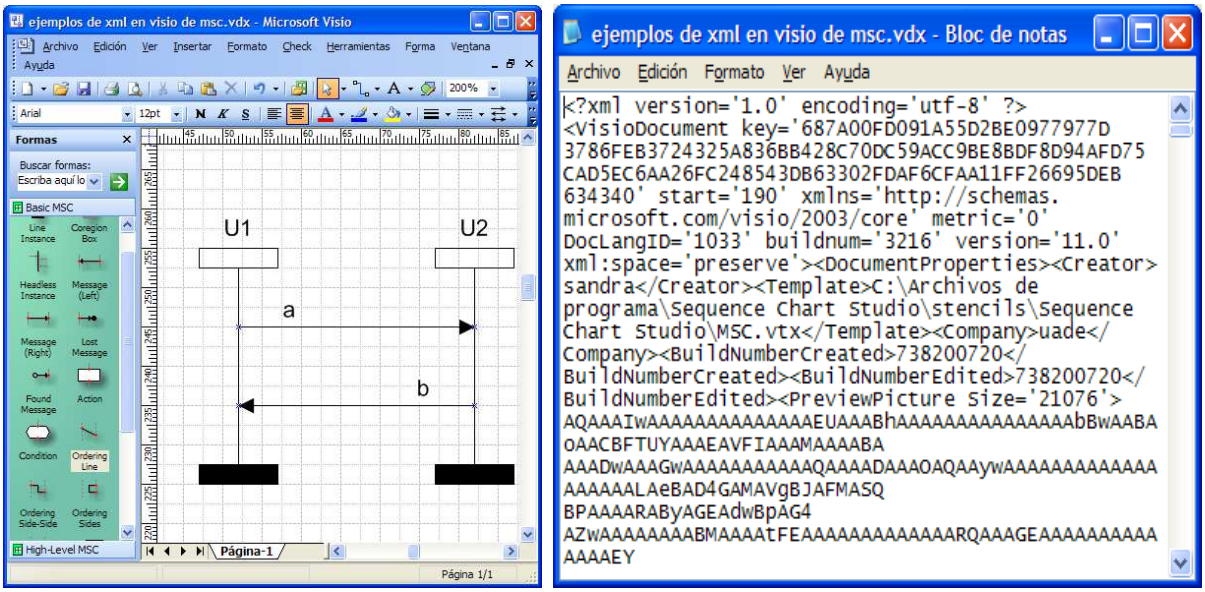


Figura 81: Ejemplo de archivo VSD y VDX generado con VISIO usando la plantilla para gráficos MSC

- **VTS**

Como con todos los dibujos de VISIO, se obtiene un archivo de extensión VSD y los demás tipos de salidas que permite este software. Esta extensión de VTS para VISIO permite además obtener un archivo de extensión VDX que es editable con cualquier editor de texto y que en su parte final se puede observar la definición XML del escenario dibujado con la gramática vista en el capítulo 2 punto 2.5.3 que al leerla permite recrear y entender el escenario VTS sin necesidad de tener dicho gráfico.
También pueden obtenerse otras salidas utilizando los traductores de VTS a redes de Petri y a SAL como se menciona en el capítulo 3.

6.3 Bondades y falencias

6.3.1 Bondades

- **TE**
  - ✓ Es un lenguaje simple y fácil de interpretar.
  - ✓ Su editor es sencillo y no requiere de demasiados conocimientos previos.
  - ✓ La herramienta gráfica genera un autómata equivalente y un archivo con el mismo que permite ser usado por SPIN.

- **MSC**
  - ✓ Es el más utilizado y conocido en el desarrollo de software para comunicaciones.
  - ✓ Existen distintas herramientas que permiten obtener gráficos MSC.
  - ✓ Permite distintos niveles de profundidad de especificación (bMSC, MSC, HMSC).
  - ✓ Cuenta con un documento explicativo del SDL como es el Z.120 de ITU.

- ✓ Existen distintos trabajos que presentan traducciones de MSC a distintas herramientas para permitir su verificación (ver punto 3.1.2).

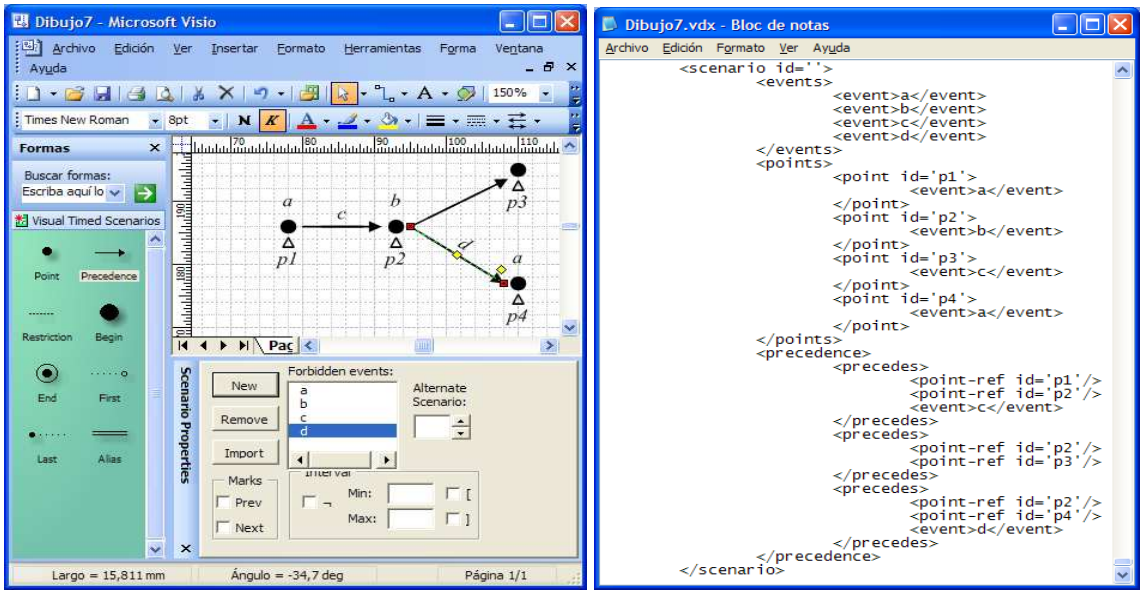


Figura 82: Ejemplo de archivo VSD y XML generado con VISIO usando la plantilla para gráficos VTS

- **VTS**
  - ✓ Cuenta con su herramienta gráfica que permite generar un autómata equivalente, un archivo XML entendible y cuenta con su formalización completa.
  - ✓ Es un lenguaje con alto nivel de expresividad. Permite graficar una gran variedad de escenarios con distintas complejidades, como por ejemplo el primer o último elemento de un conjunto.
  - ✓ Permite en un mismo gráfico mostrar las distintas opciones posibles (escenarios condicionales) gracias a los antecedentes y consecuentes.
  - ✓ Cuenta con una herramienta que genera el autómata equivalente que permite el uso de verificadores como Kronos, Uppaal o Zeus. También es posible traducir un escenario VTS a redes de Petri para utilizar el verificador TINA, y otros verificadores de TPN. Recientemente se definió una traducción a SAL que posibilita diversas maneras de análisis (ver punto 3.1.3).

6.3.2 Falencias

- **TE**
  - ✓ No permite mostrar restricciones o intervalos de tiempo.
  - ✓ No permite mostrar la ocurrencia de ciertos eventos en cualquier orden antes de la aparición de un determinado evento.
  - ✓ Sólo cuenta con el aspecto gráfico (ni textual ni formal).

- **MSC**
  - ✓ No permite observar eventos no deseados o prohibidos en la secuencia de eventos de un escenario.
  - ✓ No cuenta con una definición formal completa.

- ✓ En ciertos casos, como se vio en el capítulo 5, un gráfico MSC puede ser ambiguo en el orden en que se suceden los eventos; se debe especificar entonces si se aplica o no la política FIFO para evitar esta ambigüedad.

- **VTS**
  - ✓ No permite especificar ciclos de eventos con una determinada cantidad de iteraciones.

Capítulo 7: Conclusiones y trabajos futuros

7.1 Conclusiones

En esta tesis se presenta un estudio comparado de tres lenguajes gráficos que tienen como objetivo especificar las propiedades que debe cumplir un sistema de software que se está desarrollando.
Se han elegido TE, MSC y VTS, lenguajes con distintas características pero que permitan especificar propiedades relacionadas con la ocurrencia de eventos (aunque en MSC los eventos no están directamente determinados y la mayoría de los mismos están referenciados al envío y recepción de mensajes o actividades).

En la comparación se ha tenido en cuenta el aspecto gráfico, el aspecto textual y su formalización.

Para poder comparar formalmente se ha debido formalizar el lenguaje TE y se ha adoptado una de las varias formalizaciones que existen de MSC.

Se ha comparado TE y VTS observando que todos los escenarios especificados con TE pueden ser expresados con VTS. En cambio no resulta de igual manera cuando se trata de pasar de un escenario VTS a uno TE. En muchos casos son necesarios varios escenarios TE para representar un único escenario VTS. Esta comparación permitió determinar las reglas para pasar de un escenario TE a un escenario VTS y viceversa.

También se ha comparado MSC y VTS observando que todos los escenarios especificados con MSC pueden ser expresados con VTS a excepción de la posibilidad de MSC de mostrar ciertas iteraciones a través de Loop. El resto de las expresiones condicionales de MSC pueden expresarse en VTS aunque con más de un gráfico.
Al igual que con TE, surge de esta comparación la definición de las reglas necesarias para traducir un escenario MSC a un escenario VTS.

Aplicando las traducciones de un gráfico MSC hacia uno VTS se puede obtener la representación del mismo escenario con posibilidades de ser verificado luego de obtener el autómata correspondiente con la herramienta que VTS ofrece. La traducción de un gráfico MSC (muy utilizado en la industria) a un escenario VTS (formalismo utilizado mayormente en el ámbito académico) propone una manera de extender las bondades de los lenguajes formales (por ejemplo la verificación del modelo) fuera de lo académico sin dejar de utilizar un formalismo intuitivo y muy conocido en el ámbito industrial, principalmente en telecomunicaciones.

Estos lenguajes tienen traducciones a autómatas temporales equivalentes y permiten una forma de ser verificados utilizando herramientas como SPIN, KRONOS o UPPAAL.
Teniendo en cuenta la especificación formal de VTS se generó un código que permite representar las corridas que satisfacen el escenario especificado. Si bien las corridas que

satisfacen un escenario son infinitas, con este código se las muestra en forma de expresiones regulares donde E (o ∑) representa todas las combinaciones posibles de sucesión de eventos (o sea infinitas combinaciones) lo que corresponde a infinitas trazas.
Al poder traducir formalmente un escenario TE o MSC a un escenario VTS se puede utilizar también este generador de corridas para obtener una representación finita de las trazas que satisfagan a estos escenarios.

7.2 Trabajos futuros

A partir de esta tesis, se podrían considerar los siguientes trabajos futuros:

- En este trabajo se han considerado los gráficos MSC básicos para analizar la traducción formal debido que no se ha encontrado una formalización completa en las bibliografías que abordaban este tema. Se podría considerar la posibilidad de incorporar elementos como la co-región, los timers, las inline expression (condicionales), creación y finalización de instancias, y mensajes incompletos a las definiciones formales de MSC. Una vez que se cuente con esta ampliación a la especificación formal de un MSC se podría completar la traducción a un escenario VTS.
- En este trabajo se pudo traducir un MSC a VTS pero no se determinó cómo pasar de un HMSC a VTS. Si bien se ha realizado una traducción de gráficos de bMSC o MSC, a gráficos VTS, queda analizar la posibilidad de extender VTS para que permita soportar jerarquías de gráficos semejantes a un HMSC.
- Extender el generador de corridas para que permita incluir las restricciones temporarias. Este aspecto del código no se ha incorporado en este trabajo y su solución no surge fácilmente, ya que no es trivial.
- Adaptar el generador de corridas para que permita como archivo de entrada el texto XML que se obtiene como salida de VISIO al utilizar la plantilla de VTS.

Siglas:

CCITT	Comité Consultivo Internacional Telegráfico y Telefónico
CFM	Communicating Finite-state Machine
CTL	Computing Tree Logic
DER	Diagrama de Entidad-Relación
DFD	Diagrama de Flujo de Datos
FIFO	First In Fist Out
HMSC	High_level MSC
ITU	International Telecomunication Union
LTL	Linear Temporal Logic
MSC	Message Sequence Charts
OO	Object Oriented
PROMELA	Process or Protocol Meta Language
SCs	Sequence Charts
SDL	System Design Languages
TCTL	Timed Computation Tree Logic
TE	TimeEdit
UML	Unified Modeling Language
VTS	Visual Timed event Scenarios
XML	Extensible Markup Language

Referencias:

[ABKO04] A. Alfonso, V. Braberman, N. Kicillof, A. Olivero, "*Visual Timed Event Scenarios*", Proceedings of the 26th International Conference on Software Engineering, 2004.

[AHP96] Rajeev Alur, Gerard J. Holzmann, Doron Peled, "*An Analyzer for Message Sequence Charts*", Proceeding: TACAs '96 Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems, 1996.

[AY99] Rajeev Alur, Mihalis Yannakakis, "*Model Checking of Message Sequence Charts*", CONCUR '99 Proceedings of the 10th International Conference on Concurrency Theory, 1999.

[BDMOTY98] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, Sergio Yovine, "*Kronos: A Model-Checking Tool for Real-Time Systems*", Proceedings of the 10th International Conference on Computer Aided Verification, 1998.

[BKO05] Víctor Braberman, Nicolas Kicillof, Alfredo Olivero, "*A Scenario-Matching Approach to the Description and Model Checking of Real-Time Properties*", IEEE Transactions on Software Engineering, Volume 31, 2005.

[BOB03] Víctor Braberman, Fernando Oliveto, Matías Blaunstein, "*Scenario-based Validation and Verification for Real-Time Software: On Run Conformance and Coverage for MSC-Graphs*", 2nd International Workshop on Scenarios and State Machines: Models, Algorithms, and Tools - may 2003.

[BY03] Johan Bengtsson, Wang Yi, "*Timed Automata: Semantics, Algorithms and Tools*", In Proceedings of Lectures on Concurrency and Petri Nets'2003. pp.87~124, 2003.

[DGH08] Philippe Darondeau, Blaise Genest, Loïc Hélouët, "*Products of message sequence charts*", Proceeding FOSSACS'08/ETAPS'08 Proceedings of the Theory and practice of software, 11th international conference on Foundations of software science and computational structures, 2008.

[GGR93] Jens Grabowski, Peter Graubmann, Ekkart Rudolph, "*The Standardization of Message Sequence Charts*", Computer Networks and ISDN Systems, 1993.

[GHM03] B. Genest, L. Helouet, A. Muscholl. "*High-Level Message Sequence Charts and Projections* ", *Proc. of CONCUR'03,* Lecture Notes in Computer Science 2761, pp. 308-322, 2003.

[GM05] Blaise Genest, Anca Muscholl. "*Message Sequence Charts: A Survey*". In Fifth International Conference on Application of Concurrency to System Design (ACSD 2005), 6-9 June 2005, St. Malo, France. Pages 2-4, IEEE Computer Society, 2005.

[GMP03] B. Genest, A. Muscholl and D. Peled. "*Message Sequence Charts*", Lectures on Concurrency and Petri Nets: Advances in Petri nets (Tutorial

volume from the 4th Advanced Course on Petri Nets, ACPN 2003), pp. 537-558, 2003.

[GPS07] E. Elkind, B. Genest, D. Peled and P. Spoletini, "*Quantifying the Discord: Order Discrepancies in Message Sequence Charts".* In: 5th International Symposium on Automated Technology for Verification and Analysis (ATVA'07)*,* Tokyo, Japan, 2007.

[H01] Oystein Haugen, "*From MSC-2000 to UML 2.0 –The Future of Sequence Diagrams*", SDL 2001 Meeting UML 10th International SDL Forum Copenhagen Denmark June 2729 2001 Proceedings, 2001.

[H04] Oystein Haugen, "*Comparing UML 2.0 Interactions and MSC-2000*", published in System Analysis and Modeling Volume: 3319, Publisher: LNCS, Pages: 65 – 79, 2004.

[HT03] David Harel, P. S. Thiagarajan, "*Message sequence charts*", Published in book UML for real Kluwer Academic Publishers Norwell, MA, USA, 2003.

[IEEE98] IEEE Recommended Practice for Software Requirements Specifications, 1998.

[ITU04] International Telecommunication Union, "Message Sequence Chart (MSC). *ITU-T Recommendation Z.120*", 2004.

[KML03] S. Kryvyy, L. Matvyeyeva, M. Lopatina, "*Automatic Translation Of MSC Diagrams Into Petri Nets*", International Journal "Information Theories & Applications" Vol.10, 2003.

[LL97] Stefan Leue, Peter B. Ladkin, "*Implementing and Verifying MSC Specifications Using PROMELA/XSPIN*", Proceedings of the DIMACS Workshop SPIN96, the 2nd International Workshop on the SPIN Verification System, volume 32 of DIMACS Series, 1997.

[M01] Stephan Merz, "*Model checking: a tutorial overview* ", Published in: Modeling and verification of parallel processes, 2001.

[M96] S. Mauw, "*The formalization of message sequence charts*", Computer Networks and ISDN Systems - Special issue on SDL and MSC, Volume 28 Issue 12, June 1996.

[MOYB08] Daniel Monteverde, Alfredo Olivero, Sergio Yovine, Víctor Braberman, "*VTS-based Specification and Verification of Behavioral Properties of AADL Models*", Model Based Architecting and Construction of Embedded Systems, 2008.

[MP00] Anca Muscholl, Doron Peled, "*Analyzing Message Sequence Charts*", In Proc. 7 th Intl. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01), volume 2031 of Lect. Notes in Comp. Sci, 2000.

[MR97] Sjouke Mauw, Michel A. Reniers: "*High-level message sequence charts*". SDL Forum, 1997.

[OY10] Alfredo Olivero, Sergio Yovine, “GAMBETAS - *Ginga-oriented Automated Methodology for Better Embedded Television Application Software*”, artículo de divulgación, 2010.

[P02] Doron Peled, **"***Specification and Verification using Message Sequence Charts*”, Electronic Notes in Theoretical Computer Science Volume 65, Issue 7, May 2002, Pages 51-64 VISS 2002, Validation and Implementation of Scenario-based Specifications, 2002.

[P04] Girish Palshikar, “*An introduction to model checking*”, published in Embedded System Programming, 2004.

[R98] Michel Adriaan Reniers “*Message Sequence Chart: Syntax and Semantics*”, Faculty of Mathematics and Computing, 1998.

[RGG96] Ekkart Rudolph, Peter Graubmann, Jens Grabowski, “*Tutorial on message sequence charts*”, Published in: Journal Computer Networks and ISDN Systems - Special issue on SDL and MSC, Volume 28 Issue 12, June 1996.

[RSC97] Rational Software Corporation, ”*UML Notation Guide*”, 1997.

[S98] J. M. Spivey, “*The Z Notation: A Reference Manual”*, Second Edition. Programming Research Group, University of Oxford, Published 1998.

[SHE01] Margaret H. Smith, Gerard J. Holzmann, Kousha Etessami, “*Events and Constraints: A Graphical Editor for Capturing Logic Requirements of Programs*”, Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, IEEE Computer Society Washington, DC, USA 2001.

Anexo A
Casos que se incluyen:

Creación y finalización de instancias

Tomando como ejemplo la Figura 10, se obtiene la siguiente descripción textual teniendo en cuenta el orden horizontal y vertical:

msc ejemplo2 ;
Control : ***instance*** ;
Authorizer : ***instance*** ;
Control: ***in*** code ***from env*** ;
Control: ***create*** Authorizer;
Authorizer : ***out*** o.k. ***to env*** ;
Authorizer : stop
Control : ***endinstance ;***
Authorizer : ***endinstance ;***
endmsc ;

Mensajes perdidos y mensajes encontrados

Tomando como ejemplo la Figura 11, se obtiene la siguiente descripción textual teniendo en cuenta el orden horizontal y vertical:

msc ejemplo3 ;
UserA : ***instance*** ;
UserB : ***instance*** ;
UserA : ***out*** b ***to*** UserB ;
UserB: ***in*** b ***from*** UserA ;
UserA: ***out*** a ***to lost*** UserB ;
UserB: ***in*** c ***from found*** UserA ;
UserA : ***endinstance ;***
UserB: ***endinstance***;
endmsc ;

Co-región

Tomando como ejemplo la Figura 12, se obtiene la siguiente descripción textual. En esta descripción sólo se observa el orden vertical de cada instancia.

msc ejemplo4 ;
User : ***instance*** ;
Net : ***instance*** ;
User : ***out*** a ***to*** Net ;
User : ***concurrent*** ;
in b ***from*** Net ;
in c ***from*** Net ;
endconcurrent ;
User : ***out*** d ***to*** Net ;
Net : ***in*** a ***from*** User ;
Net : ***out*** b ***to*** User ;
Net : ***out*** c ***to*** User ;
Net : ***in*** d ***from*** User ;
User : ***endinstance ;***
Net : ***endinstance***;
endmsc ;

Timers

Iniciar el timer y detenerlo

Tomando como ejemplo la Figura 13, se obtiene la siguiente descripción textual.

msc ejemplo5a ;
    User : ***instance*** ;
    Net : ***instance*** ;
    User : ***out*** a ***to*** Net ;
    Net : ***in*** a ***from*** User ;
    Net : ***out*** b ***to*** User ;
    User : ***in*** b ***from*** Net ;
    Net : ***set*** T1 (<20);
    Net : ***timeout*** T1;
    User : ***out*** c ***to*** Net ;
    Net : ***in*** c ***from*** User ;
    User : ***endinstance ;***
    Net : ***endinstance ;***
endmsc ;

Iniciar el timer y resetearlo

Tomando como ejemplo la Figura 14, se obtiene la siguiente descripción textual.

msc ejemplo5b ;
    User : ***instance*** ;
    Net : ***instance*** ;
    User : ***out*** a ***to*** Net ;
    Net : ***in*** a ***from*** User ;
    Net : ***set*** T1;
    Net : ***out*** b ***to*** User ;
    User : ***in*** b ***from*** Net ;
    Net : ***reset*** T1;
    User : ***out*** c ***to*** Net ;
    Net : ***in*** c ***from*** User ;
    User : ***endinstance ;***
    Net : ***endinstance ;***
endmsc ;

Inline expressions

Alt (alternativas):

Tomando como ejemplo la Figura 15, se obtiene la siguiente descripción textual.

msc ejemplo6 ;
    User : ***instance*** ;
    NetA : ***instance*** ;
    NetB : ***instance*** ;
    User : ***out*** a ***to*** NetA ;
    NetA : ***in*** a ***from*** User ;
    User, NetA, NetB: ***alt begin*** ;
        NetA: ***out*** b ***to*** User ;

```
                User : in b from NetA ;
                alt;
                NetA: out b to NetB ;
                NetB : in b from NetA ;
                NetB: out b to User ;
                User : in b from NetB ;
                alt end;
        User : endinstance ;
        NetA: endinstance;
        NetB: endinstance;
endmsc ;
```

Loop (iteraciones):

Tomando como ejemplo la Figura 16, se obtiene la siguiente descripción textual.

```
msc ejemplo7 ;
        User : instance ;
        NetA : instance ;
        NetB : instance ;
        User : out a to NetA ;
        NetA : in a from User ;
        User, NetA, NetB: loop <2,5> begin ;
                NetA: out b to User ;
                User : in b from NetA ;
                NetA: out b to NetB ;
                NetB : in b from NetA ;
        loop end;
        NetB: out b to User ;
        User : in b from NetB ;
        User : endinstance ;
        NetA: endinstance;
        NetB: endinstance;
endmsc ;
```

Exc (excepción):

Tomando como ejemplo la Figura 17, se obtiene la siguiente descripción textual.

```
msc ejemplo8 ;
        User : instance ;
        NetA : instance ;
        NetB : instance ;
        User : out a to NetA ;
        NetA : in a from User ;
        User, NetA, NetB: exc begin ;
                NetA: out b to User ;
                User : in b from NetA ;
        exc end;
        NetA: out c to NetB ;
        NetB : in c from NetA ;
        User : endinstance ;
```

NetA: ***endinstance***;
NetB: ***endinstance***;
endmsc ;

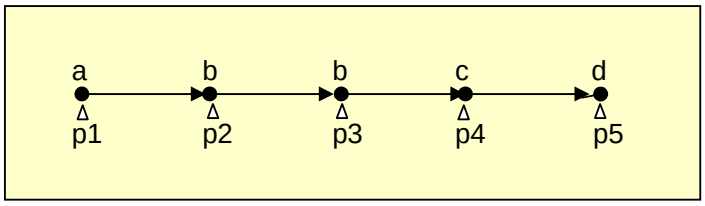
Opt (opción):

Tomando como ejemplo la Figura 18, se obtiene la siguiente descripción textual.

msc ejemplo9 ;
User : ***instance*** ;
NetA : ***instance*** ;
NetB : ***instance*** ;
User : ***out*** a ***to*** NetA ;
NetA : ***in*** a ***from*** User ;
User, NetA, NetB: ***opt begin*** ;
NetA: ***out*** b ***to*** User ;
User : ***in*** b ***from*** NetA ;
opt end;
NetA: ***out*** c ***to*** NetB ;
NetB : ***in*** c ***from*** NetA ;
User : ***endinstance ;***
NetA: ***endinstance***;
NetB: ***endinstance***;
endmsc ;

Anexo B
Casos que se incluyen:

Caso 1: puntos con el mismo evento asignado



Archivo de entrada:

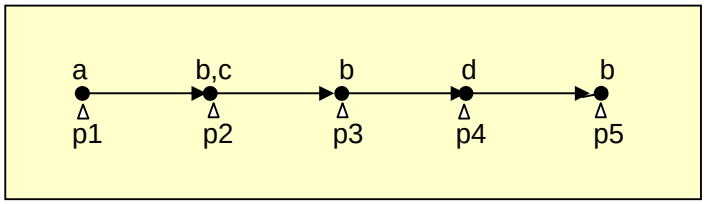
e,a
e,b
e,c
e,d
x,x
d,1,2
d,2,3
d,3,4
d,4,5
r,1,a
r,2,b
r,3,b
r,4,c
r,5,d

Archivo de salida:

"Dado el siguiente escenario VTS: S={E, P, l, d, <, <f, <l, ep, rt} donde:"
"E: conjunto finito de eventos"
"P: conjunto finito de puntos"
"l: función donde a cada punto de P le corresponde un subconjunto de eventos de E"
"d: relación asimétrica entre los puntos de desigualdad"
"<: relación de precedencia entre los puntos"
"<f: relación que representa el primer punto de un conjunto"
"<l: relación que representa el último punto de un conjunto"
"ep: función que asigna a cada par de puntos un conjunto de eventos prohibidos entre ellos"
"rt: función que asigna a cada par de puntos una restricción de tiempo entre ellos"

"y que se asumen los siguientes valores:"
"E = { a ; b ; c ; d } "
"P = { 1 ; 2 ; 3 ; 4 ; 5 } "
"l = { 1->a ; 2->b ; 3->b ; 4->c ; 5->d } "
"d = { }"
"< = { (1,2) ; (2,3) ; (3,4) ; (4,5) } "
"<f = { }"
"<l = { }"
"ep = { }"
"rt = { }"
""
""
"Corridas correspondientes al escenario :"
" corrida: 1---> 1 2 3 4 5   eventos: E* a E* b E* b E* c E* d E* "

Caso 2: más de un evento asociado a un punto



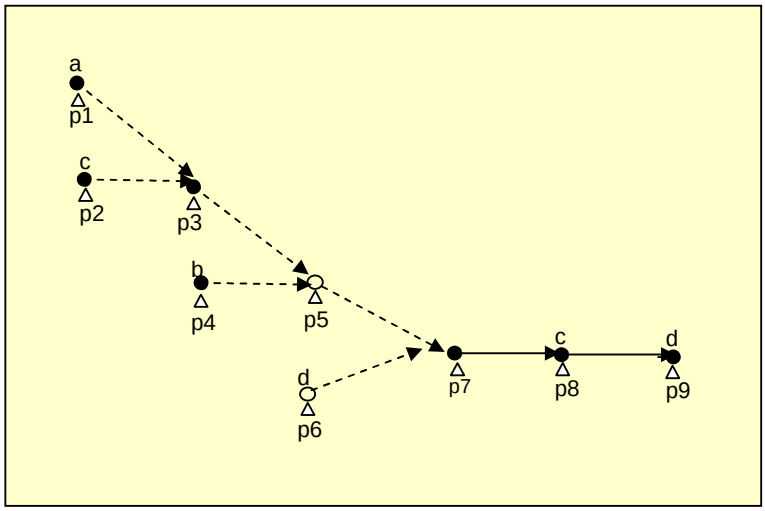
Archivo de entrada:

e,a
e,c
e,b
e,d
x,x
d,1,2
d,2,3
d,3,4
d,4,5
r,1,a
r,2,b
r,2,c
r,3,b
r,4,d
r,5,b

Archivo de salida:
Desde este caso en adelante se omite la primera parte del archivo de salida ya que es meramente explicativa y no agrega información a los ejemplos.

"y que se asumen los siguientes valores:"
"E = { a ; c ; b ; d } "
"P = { 1 ; 2 ; 3 ; 4 ; 5 } "
"I = { 1->a ; 2-> { b,c } ; 3->b ; 4->d ; 5->b } "
"< = { (1,2) ; (2,3) ; (3,4) ; (4,5) } "
"<f = { }"
"<I = { }"
"ep = { }"
"rt = { }"
""
""
"Corridas correspondientes al escenario :"
" corrida: 1---> 1 2 3 4 5 eventos: E* a E* (b|c) E* b E* d E* b E* "

Caso 3: primeros y últimos



Archivo de entrada:

e,a
e,b
e,c
e,d
x,x
d,7,8
d,8,9
l,1,3
l,2,3
l,3,5
l,4,5
f,5,7
f,6,7
r,8,c
r,9,d
r,1,a
r,2,c
r,4,b
r,6,d

Archivo de salida:

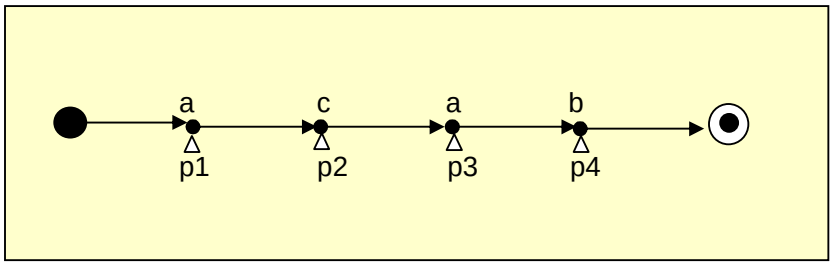
"y que se asumen los siguientes valores:"
"E = { a ; b ; c ; d } "
"P = { 7 ; 8 ; 9 ; 5 ; 6 ; 1 ; 3 ; 2 ; 4 } "
"l = { 8->c ; 9->a ; 6->d ; 1->a ; 2->c ; 4->b } "
"< = { (7,8) ; (8,9) } "
"<f = { (5,7) ; (6,7) } "
"<l = { (1,3) ; (2,3) ; (3,5) ; (4,5) } "
"ep = { }"
"rt = { }"

""
""
"Corridas correspondientes al escenario :"
" corrida: 1---> 4 2 1 8 9 6 eventos: E* b E* c E* a E* c E* a E* d E* "
" corrida: 2---> 2 4 1 8 9 6 eventos: E* c E* b E* a E* c E* a E* d E* "
" corrida: 3---> 4 2 1 6 8 9 eventos: E* b E* c E* a E* d E* c E* a E* "
" corrida: 4---> 2 4 1 6 8 9 eventos: E* c E* b E* a E* d E* c E* a E* "
" corrida: 5---> 4 2 1 8 6 9 eventos: E* b E* c E* a E* c E* d E* a E* "
" corrida: 6---> 2 4 1 8 6 9 eventos: E* c E* b E* a E* c E* d E* a E* "
" corrida: 7---> 6 8 4 2 9 1 eventos: E* d E* c E* b E* c E* a E* a E* "
" corrida: 8---> 4 6 8 2 9 1 eventos: E* b E* d E* c E* c E* a E* a E* "
" corrida: 9---> 2 6 8 4 9 1 eventos: E* c E* d E* c E* b E* a E* a E* "
" corrida: 10---> 6 4 8 2 9 1 eventos: E* d E* b E* c E* c E* a E* a E* "
" corrida: 11---> 6 2 8 4 9 1 eventos: E* d E* c E* c E* b E* a E* a E* "
" corrida: 12---> 4 2 6 8 9 1 eventos: E* b E* c E* d E* c E* a E* a E* "
" corrida: 13---> 2 4 6 8 9 1 eventos: E* c E* b E* d E* c E* a E* a E* "
" corrida: 14---> 6 8 2 4 9 1 eventos: E* d E* c E* c E* b E* a E* a E* "
" corrida: 15---> 6 8 9 4 2 1 eventos: E* d E* c E* a E* b E* c E* a E* "
" corrida: 16---> 4 6 2 8 9 1 eventos: E* b E* d E* c E* c E* a E* a E* "
" corrida: 17---> 2 6 4 8 9 1 eventos: E* c E* d E* b E* c E* a E* a E* "
" corrida: 18---> 6 4 2 8 9 1 eventos: E* d E* b E* c E* c E* a E* a E* "
" corrida: 19---> 6 2 4 8 9 1 eventos: E* d E* c E* b E* c E* a E* a E* "
" corrida: 20---> 6 8 4 9 2 1 eventos: E* d E* c E* b E* a E* c E* a E* "
" corrida: 21---> 4 6 8 9 2 1 eventos: E* b E* d E* c E* a E* c E* a E* "
" corrida: 22---> 2 6 8 9 4 1 eventos: E* c E* d E* c E* a E* b E* a E* "
" corrida: 23---> 6 4 8 9 2 1 eventos: E* d E* b E* c E* a E* c E* a E* "
" corrida: 24---> 6 2 8 9 4 1 eventos: E* d E* c E* c E* a E* b E* a E* "
" corrida: 25---> 4 2 6 1 8 9 eventos: E* b E* c E* d E* a E* c E* a E* "
" corrida: 26---> 2 4 6 1 8 9 eventos: E* c E* b E* d E* a E* c E* a E* "
" corrida: 27---> 6 8 2 9 4 1 eventos: E* d E* c E* c E* a E* b E* a E* "
" corrida: 28---> 6 8 9 2 4 1 eventos: E* d E* c E* a E* c E* b E* a E* "
" corrida: 29---> 4 6 2 1 8 9 eventos: E* b E* d E* c E* a E* c E* a E* "
" corrida: 30---> 2 6 4 1 8 9 eventos: E* c E* d E* b E* a E* c E* a E* "
" corrida: 31---> 6 4 2 1 8 9 eventos: E* d E* b E* c E* a E* c E* a E* "
" corrida: 32---> 6 2 4 1 8 9 eventos: E* d E* c E* b E* a E* c E* a E* "
" corrida: 33---> 6 8 4 2 1 9 eventos: E* d E* c E* b E* c E* a E* a E* "
" corrida: 34---> 4 6 8 2 1 9 eventos: E* b E* d E* c E* c E* a E* a E* "
" corrida: 35---> 2 6 8 4 1 9 eventos: E* c E* d E* c E* b E* a E* a E* "
" corrida: 36---> 6 4 8 2 1 9 eventos: E* d E* b E* c E* c E* a E* a E* "
" corrida: 37---> 6 2 8 4 1 9 eventos: E* d E* c E* c E* b E* a E* a E* "
" corrida: 38---> 4 2 6 8 1 9 eventos: E* b E* c E* d E* c E* a E* a E* "
" corrida: 39---> 2 4 6 8 1 9 eventos: E* c E* b E* d E* c E* a E* a E* "
" corrida: 40---> 6 8 2 4 1 9 eventos: E* d E* c E* c E* b E* a E* a E* "
" corrida: 41---> 4 6 2 8 1 9 eventos: E* b E* d E* c E* c E* a E* a E* "
" corrida: 42---> 2 6 4 8 1 9 eventos: E* c E* d E* b E* c E* a E* a E* "
" corrida: 43---> 6 4 2 8 1 9 eventos: E* d E* b E* c E* c E* a E* a E* "
" corrida: 44---> 6 2 4 8 1 9 eventos: E* d E* c E* b E* c E* a E* a E* "
" corrida: 45---> 2 1 4 8 9 6 eventos: E* c E* a E* b E* c E* a E* d E* "
" corrida: 46---> 2 1 4 6 8 9 eventos: E* c E* a E* b E* d E* c E* a E* "
" corrida: 47---> 2 1 4 8 6 9 eventos: E* c E* a E* b E* c E* d E* a E* "
" corrida: 48---> 6 8 2 1 9 4 eventos: E* d E* c E* c E* a E* a E* b E* "
" corrida: 49---> 2 6 8 1 9 4 eventos: E* c E* d E* c E* a E* a E* b E* "
" corrida: 50---> 6 2 8 1 9 4 eventos: E* d E* c E* c E* a E* a E* b E* "
" corrida: 51---> 2 1 6 8 9 4 eventos: E* c E* a E* d E* c E* a E* b E* "
" corrida: 52---> 6 8 9 2 1 4 eventos: E* d E* c E* a E* c E* a E* b E* "

" corrida: 53---> 2 6 1 8 9 4 eventos: E* c E* d E* a E* c E* a E* b E* "
" corrida: 54---> 6 2 1 8 9 4 eventos: E* d E* c E* a E* c E* a E* b E* "
" corrida: 55---> 6 8 2 9 1 4 eventos: E* d E* c E* c E* a E* a E* b E* "
" corrida: 56---> 2 6 8 9 1 4 eventos: E* c E* d E* c E* a E* a E* b E* "
" corrida: 57---> 6 2 8 9 1 4 eventos: E* d E* c E* c E* a E* a E* b E* "
" corrida: 58---> 2 1 6 4 8 9 eventos: E* c E* a E* d E* b E* c E* a E* "
" corrida: 59---> 2 6 1 4 8 9 eventos: E* c E* d E* a E* b E* c E* a E* "
" corrida: 60---> 6 2 1 4 8 9 eventos: E* d E* c E* a E* b E* c E* a E* "
" corrida: 61---> 6 8 2 1 4 9 eventos: E* d E* c E* c E* a E* b E* a E* "
" corrida: 62---> 2 6 8 1 4 9 eventos: E* c E* d E* c E* a E* b E* a E* "
" corrida: 63---> 6 2 8 1 4 9 eventos: E* d E* c E* c E* a E* b E* a E* "
" corrida: 64---> 2 1 6 8 4 9 eventos: E* c E* a E* d E* c E* b E* a E* "
" corrida: 65---> 2 6 1 8 4 9 eventos: E* c E* d E* a E* c E* b E* a E* "
" corrida: 66---> 6 2 1 8 4 9 eventos: E* d E* c E* a E* c E* b E* a E* "
" corrida: 67---> 4 1 2 8 9 6 eventos: E* b E* a E* c E* c E* a E* d E* "
" corrida: 68---> 1 4 2 8 9 6 eventos: E* a E* b E* c E* c E* a E* d E* "
" corrida: 69---> 4 1 2 6 8 9 eventos: E* b E* a E* c E* d E* c E* a E* "
" corrida: 70---> 1 4 2 6 8 9 eventos: E* a E* b E* c E* d E* c E* a E* "
" corrida: 71---> 4 1 2 8 6 9 eventos: E* b E* a E* c E* c E* d E* a E* "
" corrida: 72---> 1 4 2 8 6 9 eventos: E* a E* b E* c E* c E* d E* a E* "
" corrida: 73---> 6 8 4 1 9 2 eventos: E* d E* c E* b E* a E* a E* c E* "
" corrida: 74---> 4 6 8 1 9 2 eventos: E* b E* d E* c E* a E* a E* c E* "
" corrida: 75---> 1 6 8 4 9 2 eventos: E* a E* d E* c E* b E* a E* c E* "
" corrida: 76---> 6 4 8 1 9 2 eventos: E* d E* b E* c E* a E* a E* c E* "
" corrida: 77---> 6 1 8 4 9 2 eventos: E* d E* a E* c E* b E* a E* c E* "
" corrida: 78---> 4 1 6 8 9 2 eventos: E* b E* a E* d E* c E* a E* c E* "
" corrida: 79---> 1 4 6 8 9 2 eventos: E* a E* b E* d E* c E* a E* c E* "
" corrida: 80---> 6 8 1 4 9 2 eventos: E* d E* c E* a E* b E* a E* c E* "
" corrida: 81---> 6 8 9 4 1 2 eventos: E* d E* c E* a E* b E* a E* c E* "
" corrida: 82---> 4 6 1 8 9 2 eventos: E* b E* d E* a E* c E* a E* c E* "
" corrida: 83---> 1 6 4 8 9 2 eventos: E* a E* d E* b E* c E* a E* c E* "
" corrida: 84---> 6 4 1 8 9 2 eventos: E* d E* b E* a E* c E* a E* c E* "
" corrida: 85---> 6 1 4 8 9 2 eventos: E* d E* a E* b E* c E* a E* c E* "
" corrida: 86---> 6 8 4 9 1 2 eventos: E* d E* c E* b E* a E* a E* c E* "
" corrida: 87---> 4 6 8 9 1 2 eventos: E* b E* d E* c E* a E* a E* c E* "
" corrida: 88---> 1 6 8 9 4 2 eventos: E* a E* d E* c E* a E* b E* c E* "
" corrida: 89---> 6 4 8 9 1 2 eventos: E* d E* b E* c E* a E* a E* c E* "
" corrida: 90---> 6 1 8 9 4 2 eventos: E* d E* a E* c E* a E* b E* c E* "
" corrida: 91---> 4 1 6 2 8 9 eventos: E* b E* a E* d E* c E* c E* a E* "
" corrida: 92---> 1 4 6 2 8 9 eventos: E* a E* b E* d E* c E* c E* a E* "
" corrida: 93---> 6 8 1 9 4 2 eventos: E* d E* c E* a E* a E* b E* c E* "
" corrida: 94---> 6 8 9 1 4 2 eventos: E* d E* c E* a E* a E* b E* c E* "
" corrida: 95---> 4 6 1 2 8 9 eventos: E* b E* d E* a E* c E* c E* a E* "
" corrida: 96---> 1 6 4 2 8 9 eventos: E* a E* d E* b E* c E* c E* a E* "
" corrida: 97---> 6 4 1 2 8 9 eventos: E* d E* b E* a E* c E* c E* a E* "
" corrida: 98---> 6 1 4 2 8 9 eventos: E* d E* a E* b E* c E* c E* a E* "
" corrida: 99---> 6 8 4 1 2 9 eventos: E* d E* c E* b E* a E* c E* a E* "
" corrida: 100---> 4 6 8 1 2 9 eventos: E* b E* d E* c E* a E* c E* a E* "
" corrida: 101---> 1 6 8 4 2 9 eventos: E* a E* d E* c E* b E* c E* a E* "
" corrida: 102---> 6 4 8 1 2 9 eventos: E* d E* b E* c E* a E* c E* a E* "
" corrida: 103---> 6 1 8 4 2 9 eventos: E* d E* a E* c E* b E* c E* a E* "
" corrida: 104---> 4 1 6 8 2 9 eventos: E* b E* a E* d E* c E* c E* a E* "
" corrida: 105---> 1 4 6 8 2 9 eventos: E* a E* b E* d E* c E* c E* a E* "
" corrida: 106---> 6 8 1 4 2 9 eventos: E* d E* c E* a E* b E* c E* a E* "
" corrida: 107---> 4 6 1 8 2 9 eventos: E* b E* d E* a E* c E* c E* a E* "

" corrida: 108---> 1 6 4 8 2 9 eventos: E a E* d E* b E* c E* c E* a E* "*
" corrida: 109---> 6 4 1 8 2 9 eventos: E d E* b E* a E* c E* c E* a E* "*
" corrida: 110---> 6 1 4 8 2 9 eventos: E d E* a E* b E* c E* c E* a E* "*
" corrida: 111---> 1 2 4 8 9 6 eventos: E a E* c E* b E* c E* a E* d E* "*
" corrida: 112---> 1 2 4 6 8 9 eventos: E a E* c E* b E* d E* c E* a E* "*
" corrida: 113---> 1 2 4 8 6 9 eventos: E a E* c E* b E* c E* d E* a E* "*
" corrida: 114---> 6 8 1 2 9 4 eventos: E d E* c E* a E* c E* a E* b E* "*
" corrida: 115---> 1 6 8 2 9 4 eventos: E a E* d E* c E* c E* a E* b E* "*
" corrida: 116---> 6 1 8 2 9 4 eventos: E d E* a E* c E* c E* a E* b E* "*
" corrida: 117---> 1 2 6 8 9 4 eventos: E a E* c E* d E* c E* a E* b E* "*
" corrida: 118---> 6 8 9 1 2 4 eventos: E d E* c E* a E* a E* c E* b E* "*
" corrida: 119---> 1 6 2 8 9 4 eventos: E a E* d E* c E* c E* a E* b E* "*
" corrida: 120---> 6 1 2 8 9 4 eventos: E d E* a E* c E* c E* a E* b E* "*
" corrida: 121---> 6 8 1 9 2 4 eventos: E d E* c E* a E* a E* c E* b E* "*
" corrida: 122---> 1 6 8 9 2 4 eventos: E a E* d E* c E* a E* c E* b E* "*
" corrida: 123---> 6 1 8 9 2 4 eventos: E d E* a E* c E* a E* c E* b E* "*
" corrida: 124---> 1 2 6 4 8 9 eventos: E a E* c E* d E* b E* c E* a E* "*
" corrida: 125---> 1 6 2 4 8 9 eventos: E a E* d E* c E* b E* c E* a E* "*
" corrida: 126---> 6 1 2 4 8 9 eventos: E d E* a E* c E* b E* c E* a E* "*
" corrida: 127---> 6 8 1 2 4 9 eventos: E d E* c E* a E* c E* b E* a E* "*
" corrida: 128---> 1 6 8 2 4 9 eventos: E a E* d E* c E* c E* b E* a E* "*
" corrida: 129---> 6 1 8 2 4 9 eventos: E d E* a E* c E* c E* b E* a E* "*
" corrida: 130---> 1 2 6 8 4 9 eventos: E a E* c E* d E* c E* b E* a E* "*
" corrida: 131---> 1 6 2 8 4 9 eventos: E a E* d E* c E* c E* b E* a E* "*
" corrida: 132---> 6 1 2 8 4 9 eventos: E d E* a E* c E* c E* b E* a E* "*

Caso 4: puntos de inicio y final



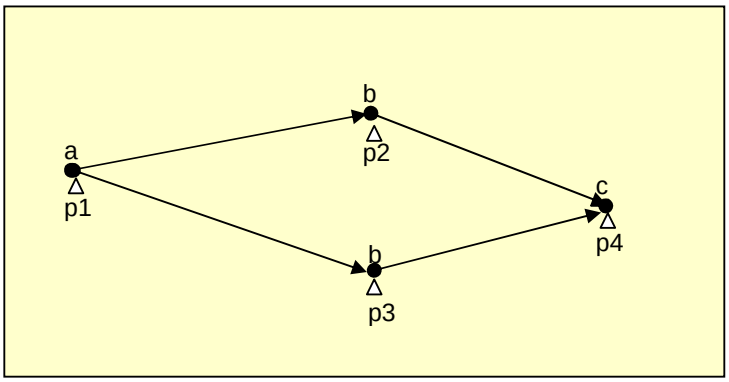
Archivo de entrada:

e,a
e,b
e,c
x,x
d,start,1
d,1,2
d,2,3
d,3,4
d,4,end
r,1,a
r,2,c
r,3,a
r,4,b

Archivo de salida:

"y que se asumen los siguientes valores:"
"E = { a ; b ; c } "
"P = { 1 ; 2 ; 3 ; 4 } "
"I = { 1->a ; 2->c ; 3->a ; 4->b } "
"d = { }"
"< = { (1,2) ; (2,3) ; (3,4) } "
"<f = { }"
"<I = { }"
"ep = { }"
"rt = { }"
""
""
"Corridas correspondientes al escenario :"
" corrida: 1---> 1 2 3 4 eventos: E* a E* c E* a E* b E* "

Caso 5: puntos con misma etiqueta



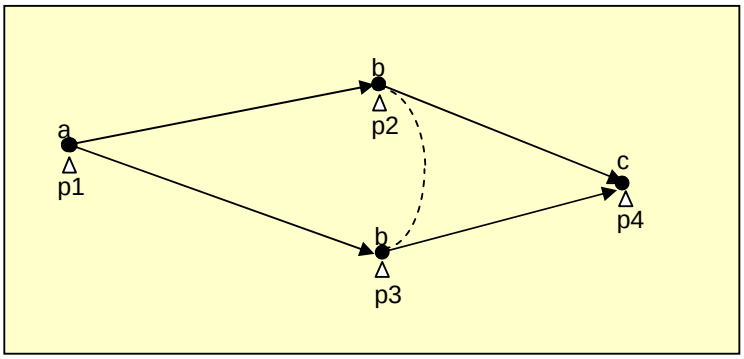
Archivo de entrada:

e,a
e,b
e,c
x,x
d,1,2
d,1,3
d,2,4
d,3,4
r,1,a
r,2,b
r,3,b
r,4,c

Archivo de salida:

"y que se asumen los siguientes valores:"
"E = { a ; b ; c } "
"P = { 1 ; 2 ; 3 ; 4 } "
"l = { 1->a ; 2->b ; 3->b ; 4->c } "
"d = { }"
"< = { (1,2) ; (1,3) ; (2,4) ; (3,4) } "
"<f = { }"
"<l = { }"
"ep = { }"
"rt = { }"
""
""
"Corridas correspondientes al escenario :"
" corrida: 1---> 1 2 3 4 eventos: E* a E* b E* b E* c E* "
" corrida: 2---> 1 3 2 4 eventos: E* a E* b E* b E* c E* "

Caso 6: relación de asimetría



En estos casos en que hay una relación de asimétrica de desigualdad entre los puntos, no se generarán corridas extras como en el ejemplo anterior.

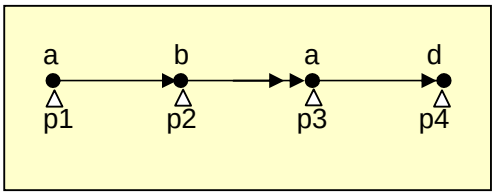
Archivo de entrada:

e,a
e,b
e,c
x,x
d,1,2
d,1,3
d,2,4
d,3,4
r,1,a
r,2,b
r,3,b
r,4,c
a,2,3
a,3,2

Archivo de salida:

"y que se asumen los siguientes valores:"
"E = { a ; b ; c } "
"P = { 1 ; 2 ; 3 ; 4 } "
"I = { 1->a ; 2->b ; 3->b ; 4->c } "
"d = { (2,3) ; (3,2) } "
"< = { (1,2) ; (1,3) ; (2,4) ; (3,4) } "
"<f = { }"
"<I = { }"
"ep = { }"
"rt = { }"
""
""
"Corridas correspondientes al escenario :"
" corrida: 1---> 1 2 3 4 eventos: E* a E* b E* b E* c E* "
" corrida: 2---> 1 3 2 4 eventos: E* a E* b E* b E* c E* "

Caso 7: evento prohibido



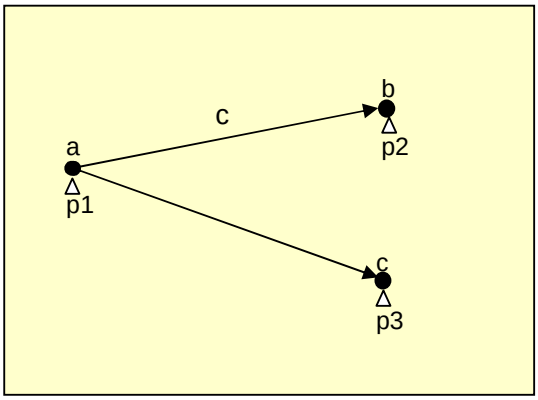
Archivo de entrada:

```
e,a
e,b
e,c
e,d
x,x
d,1,2
d,2,3
d,3,4
r,1,a
r,2,b
r,3,a
r,4,d
x,x,x
p,2,3,a
```

Archivo de salida:

```
"y que se asumen los siguientes valores:"
"E  = { a ; b ; c ; d } "
"P  = { 1 ;  2 ;  3 ;  4 } "
"l  = { 1->a ; 2->b ; 3->a ; 4->d } "
"d  = { }"
"<  = { (1,2) ; (2,3) ; (3,4) } "
"<f = { }"
"<l = { }"
"ep = { (2,3)->a } "
"rt = { }"
""
""
"Corridas correspondientes al escenario :"
" corrida:   1---> 1 2 3 4      eventos:    E*   a   E*   b   (E-{a})*   a   E*   d   E*   "
```

Caso 8: evento prohibido



Archivo de entrada:

e,a
e,b
e,c
e,d
x,x
d,1,2
d,1,3
r,1,a
r,2,b
r,3,c
x,x,x
p,1,2,c

Archivo de salida:

"y que se asumen los siguientes valores:"
"E = { a ; b ; c ; d } "
"P = { 1 ; 2 ; 3 } "
"I = { 1->a ; 2->b ; 3->c } "
"d = { }"
"< = { (1,2) ; (1,3) } "
"<f = { }"
"<I = { }"
"ep = { (1,2)-> { c }}"
"rt = { }"
""""
""""
"Corridas correspondientes al escenario :"
" corrida: 1---> 1 2 3 eventos: E* a (E-{c})* b E* c E* "