UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T Z.120

(11/1999)

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

SERIE Z: LENGUAJES Y ASPECTOS GENERALES DE SOPORTE LÓGICO PARA SISTEMAS DE TELECOMUNICACIÓN

Técnicas de descripción formal – Gráficos de secuencias de mensajes

Gráficos de secuencias de mensajes

Recomendación UIT-T Z.120

(Anteriormente Recomendación del CCITT)

RECOMENDACIONES UIT-T DE LA SERIE Z

LENGUAJES Y ASPECTOS GENERALES DE SOPORTE LÓGICO PARA SISTEMAS DE TELECOMUNICACIÓN

TÉCNICAS DE DESCRIPCIÓN FORMAL	
Lenguaje de especificación y descripción	Z.100–Z.109
Aplicación de técnicas de descripción formal	Z.110–Z.119
Gráficos de secuencias de mensajes	**Z.120–Z.129**
LENGUAJES DE PROGRAMACIÓN	
CHILL: el lenguaje de programación del UIT-T	Z.200–Z.209
LENGUAJE HOMBRE-MÁQUINA	
Principios generales	Z.300–Z.309
Sintaxis básica y procedimientos de diálogo	Z.310–Z.319
LHM ampliado para terminales con pantalla de visualización	Z.320–Z.329
Especificación de la interfaz hombre-máquina	Z.330–Z.399
CALIDAD DE SOPORTES LÓGICOS DE TELECOMUNICACIONES	Z.400–Z.499
MÉTODOS PARA VALIDACIÓN Y PRUEBAS	Z.500–Z.599

Para más información, véase la Lista de Recomendaciones del UIT-T.

Recomendación UIT-T Z.120

Gráficos de secuencias de mensajes

Resumen

Alcance/objetivo

El objetivo de recomendar el gráfico de secuencias de mensajes (MSC, *message sequence chart*) es proporcionar un lenguaje de trazos para la especificación y descripción del comportamiento de comunicación de componentes de sistema y su entorno mediante el intercambio de mensajes. Dado que en los MSC el comportamiento de comunicación se presenta de una manera muy intuitiva y transparente, especialmente en la representación gráfica, el lenguaje MSC se aprende, utiliza e interpreta con facilidad. En relación con otros lenguajes puede utilizarse para el soporte de metodologías de especificación, diseño, simulación, prueba y documentación de sistemas.

Cobertura

En esta Recomendación se presenta una definición de sintaxis para los gráficos de secuencias de mensajes en su representación textual y gráfica. Se ofrece una descripción de semántica informal.

Aplicación

El MSC tiene un vasto campo de aplicación. No está diseñado para un dominio de aplicación única. Un importante campo de aplicación de MSC es una especificación general del comportamiento de comunicación de los sistemas en tiempo real, en particular, los sistemas de conmutación de telecomunicaciones. Mediante gráficos MSC pueden especificarse dibujos de sistema seleccionados, primordialmente los casos 'normalizados'. Los casos no normalizados que comprenden comportamientos excepcionales se pueden describir a partir de los normalizados. De este modo, pueden utilizarse los MSC para la especificación de requisitos, la especificación de interfaces, la simulación y validación, la especificación de prueba y la documentación de sistemas en tiempo real. Los MSC pueden utilizarse junto con otros lenguajes de especificación, en particular el lenguaje especificación y descripción SDL (*specification and description language*). En este contexto, los MSC proporcionan también una base para el diseño de sistemas SDL.

Situación/estabilidad

MSC es estable. Esta Recomendación es una continuación natural de la versión de 1996 a la que se han añadido los conceptos de:

- datos;
- tiempo;
- flujo de control;
- orientación a objeto en documento MSC.

Trabajo asociado

– Recomendación UIT-T Q.65 (1997), *Métodología funcional unificada para la caracterización de servicios y capacidades de red*.

– Recomendación UIT-T X.210 (1993) | ISO/CEI 10731:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Modelo de referencia básico: Convenios para la definición de servicios en la interconexión de sistemas abiertos*.

– Recomendación UIT-T X.292 (1998), *Metodología y marco de las pruebas de conformidad para interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del UIT-T – Notación combinada arborescente y tabular*.

– Recomendación UIT-T Z.100 (1999), *Lenguaje de especificación y descripción*.

– UML 1.2, OMG 1999.

Orígenes

La Recomendación UIT-T Z.120, revisada por la Comisión de Estudio 10 (1997-2000) del UIT-T, fue aprobada por el procedimiento de la Resolución 1 de la CMNT el 19 de noviembre de 1999.

PREFACIO

La UIT (Unión Internacional de Telecomunicaciones) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones. El UIT-T (Sector de Normalización de las Telecomunicaciones de la UIT) es un órgano permanente de la UIT. Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Conferencia Mundial de Normalización de las Telecomunicaciones (CMNT), que se celebra cada cuatro años, establece los temas que han de estudiar las Comisiones de Estudio del UIT-T, que a su vez producen Recomendaciones sobre dichos temas.

La aprobación de Recomendaciones por los Miembros del UIT-T es el objeto del procedimiento establecido en la Resolución 1 de la CMNT.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI.

NOTA

En esta Recomendación, la expresión "Administración" se utiliza para designar, en forma abreviada, tanto una administración de telecomunicaciones como una empresa de explotación reconocida de telecomunicaciones.

PROPIEDAD INTELECTUAL

La UIT señala a la atención la posibilidad de que la utilización o aplicación de la presente Recomendación suponga el empleo de un derecho de propiedad intelectual reivindicado. La UIT no adopta ninguna posición en cuanto a la demostración, validez o aplicabilidad de los derechos de propiedad intelectual reivindicados, ya sea por los miembros de la UIT o por terceros ajenos al proceso de elaboración de Recomendaciones.

En la fecha de aprobación de la presente Recomendación, la UIT [ha recibido/no ha recibido] notificación de propiedad intelectual, protegida por patente, que puede ser necesaria para aplicar esta Recomendación. Sin embargo, debe señalarse a los usuarios que puede que esta información no se encuentre totalmente actualizada al respecto, por lo que se les insta encarecidamente a consultar la base de datos sobre patentes de la TSB.

ÍNDICE

Página

Recomendación UIT-T Z.120

Gráficos de secuencias de mensajes

(revisada en 1999)

1 Introducción

1.1 Objetivos del MSC

Gráficos de secuencias de mensajes (MSC, *message sequence chart*) es un lenguaje para describir la interacción de ejemplares de traspaso de mensaje independientes. Las principales características del lenguaje MSC son las siguientes:

- MSC es un lenguaje de escenario. Un MSC describe el orden en que tienen lugar las comunicaciones y otros eventos. Además, permite imponer restricciones a los valores de datos transmitidos y a la temporización de eventos.
- MSC soporta especificaciones completas e incompletas. Tiene la posibilidad de describir comportamientos incompletos utilizados en análisis efectuados en fases iniciales y para fines de documentación.
- MSC es un lenguaje gráfico. Los diagramas bidimensionales dan una visión de conjunto del comportamiento de ejemplares en comunicación. La forma textual del MSC está prevista principalmente para el intercambio entre herramientas y como una base para el análisis formal automático.
- MSC es un lenguaje formal. La definición del lenguaje se da en lenguaje ordinario, así como en una notación formal.
- MSC es un lenguaje práctico. Se utiliza en todo el proceso de ingeniería. Su campo de aplicación se extiende desde el análisis de dominio y la generación de ideas, pasando por las fases de formulación de requisitos y diseño, hasta las pruebas. MSC se utiliza de maneras algo diferentes en las diversas fases, y es importante que este lenguaje tenga un poder de expresión formal, así como una apariencia intuitiva.
- MSC tiene un amplio campo de aplicación. No está consagrado a un solo dominio de aplicación.
- MSC soporta el diseño estructurado. Escenarios simples (descritos por gráficos de secuencias de mensajes básicos) pueden combinarse para formar especificaciones más completas por medio de gráficos de secuencias de mensajes de alto nivel. Los gráficos MSC se reúnen en un documento MSC. Un diseño modular de escenarios está soportado por mecanismos para descomposición y reutilización.
- MSC se utiliza a menudo con otros métodos y lenguajes. Su definición formal permite la validación formal y automatizada de un MSC con respecto a un modelo descrito en un lenguaje diferente. MSC puede utilizarse, por ejemplo, con SDL (*specification and description language*) y TTCN (*tree and tabular combined notation*).
- Un escenario especificado en un MSC se interpreta usualmente en el sentido de que la implementación debe, al menos, presentar el comportamiento expresado en el escenario. Otras interpretaciones son también posibles. Un MSC puede utilizarse, por ejemplo, para especificar escenarios desautorizados.

1.2 Organización de la Recomendación

Esta Recomendación está estructurada de la manera siguiente. En la cláusula 2 se formulan las reglas generales sobre sintaxis, dibujo y paginación. En la cláusula 3 se presenta una definición del documento MSC. En la cláusula 4 se encuentra la definición de gráficos de secuencias de mensajes y los constituyentes básicos, esto es, ejemplar, mensaje, ordenación general, condición, temporizador, acción, creación de ejemplar y terminación de ejemplar. La cláusula 5 contiene los conceptos de datos y la cláusula 6 define los conceptos de tiempo. En la cláusula 7 se presentan los conceptos de alto nivel relativos a la estructuración y modularización. Estos conceptos soportan una especificación "de arriba a abajo" y permite un afinamiento de ejemplares individuales mediante la corregión (véase 7.1) y la descomposición de ejemplar (véase 7.4). Las expresiones en línea se definen en 7.2 y las referencias de MSC en 7.3. El MSC de alto nivel (véase 7.5) permite la composición de MSC y la reutilización de gráficos MSC o de partes de éstos en diferentes niveles. En la parte II se presentan ejemplos de todos los constructivos de MSC. El anexo A contiene un índice de las palabras clave, escritas entre paréntesis angulares (< >) y de los símbolos no terminales.

1.3 Metalenguaje para la gramática textual

En la forma Backus-Naur (BNF, *Backus-Naur form*) se indica un símbolo terminal no encerrándolo en paréntesis angulares (que son los signos menor que, y mayor que, < and >), o mediante una de las dos representaciones <name> y <character string>.

Los paréntesis angulares y las palabras interiores a los mismos pueden ser, o bien un símbolo no terminal, o uno de los dos símbolos terminales <character string> o <name>. Las categorías sintácticas son no terminales indicados por una o más palabras encerradas entre paréntesis angulares. Para cada símbolo no terminal se da una regla de producción en forma de gramática textual concreta o en forma de gramática gráfica concreta. Por ejemplo:

```
<instance parameter decl> ::=
                    inst <instance parm decl list> <end>
```

Una regla de producción para un símbolo no terminal se compone del símbolo no terminal situado a la izquierda del símbolo ::=, y de uno o más constructivos constituidos por uno o más símbolos no terminales y/o terminales colocados a la derecha. Por ejemplo, <instance parameter decl> e <instance parm decl list> y <end> en el ejemplo anterior son símbolos no terminales; **inst** es un símbolo terminal.

A veces un símbolo incluye una parte subrayada. Esta parte subrayada destaca un aspecto semántico de ese símbolo, por ejemplo <msc name> es sintácticamente idéntico a <name> pero semánticamente requiere que el nombre sea un nombre de gráfico de secuencias de mensajes.

A la derecha del símbolo ::= pueden darse varias producciones alternativas para el no terminal, separadas por barras verticales (|). Por ejemplo:

```
<incomplete message area> ::=
                    <lost message area>
                    |       <found message area>
```

expresa que un <incomplete message area> es un <lost message area> o un <found message area>.

Pueden agruparse conjuntamente elementos sintácticos utilizando llaves ({ y }). Un grupo encerrado entre llaves puede contener una o más barras verticales, que indican elementos sintácticos alternativos. Por ejemplo:

```
<msc document body> ::=
                    { <message sequence chart> | <msc diagram> }*
```

La repetición de grupos encerrados entre llaves se representa mediante un asterisco (*) o un signo más (+). Un asterisco indica que el grupo es facultativo y puede repetirse ulteriormente cualquier número de veces; un signo más expresa que el grupo tiene que estar presente y puede repetirse

ulteriormente cualquier número de veces. El ejemplo anterior indica que un <msc document body> puede estar vacío pero puede también contener cualquier número de <message sequence chart> y/o <msc diagram>.

Si los elementos sintácticos están agrupados mediante corchetes ([y]), el grupo es facultativo. Por ejemplo:

```
<identifier> ::=
                        [ <qualifier> ] <name>
```

expresa que un <identifier> puede, pero no tiene necesariamente que, contener <qualifier>.

Los operadores de metalenguaje tienen el orden de presidencia que sigue. El operador que vincula más débil es el operador alternativo "|". A continuación viene la secuenciación de izquierda a derecha. Los operadores "+" y "*" vinculan más fuerte que la secuenciación, y finalmente los corchetes "[...]" y las llaves "{ ... }" vinculan lo máximo.

1.4 Metalenguaje para la gramática gráfica

El metalenguaje para la gramática gráfica se basa en algunos constructivos que se describen informalmente a continuación. La sintaxis gráfica no es lo suficientemente precisa para describir los gráficos de tal manera que no haya variaciones gráficas. Se permiten pequeñas variaciones de las formas reales de los símbolos terminales gráficos, que incluyen por ejemplo, el oscurecimiento de símbolos rellenados, la forma de una cabeza de flecha y el tamaño relativo de elementos gráficos. Cuando sea necesario, se completará la sintaxis gráfica con la explicación informal del aspecto de las construcciones.

El metalenguaje consiste en una notación similar a la forma Backus-Naur (BNF) con las metaconstrucciones especiales: ***contains***, ***is followed by***, ***is associated with***, ***is attached to***, ***above*** y ***set***. Estos constructivos se comportan como reglas de producción BNF normales pero implican, adicionalmente, alguna relación geométrica o lógica entre los argumentos. El constructivo ***is attached to*** se comporta de modo algo diferente, como se explica a continuación. Al lado izquierdo de todos los constructivos, salvo ***above***, debe ser un símbolo. Un símbolo es un no terminal que produce en cada secuencia de producción exactamente un terminal gráfico. Consideraremos que un símbolo que ***is attached to*** (**está ligado a**) otras zonas o que ***is associated with*** (**está asociado con**) una cadena de texto es también un símbolo. La explicación es informal y el metalenguaje no describe con precisión las dependencias geométricas.

contains

"<area1> ***contains*** <area2>" significa que <area2> está contenida en <area1>, en principio geométricamente, pero también lógicamente.

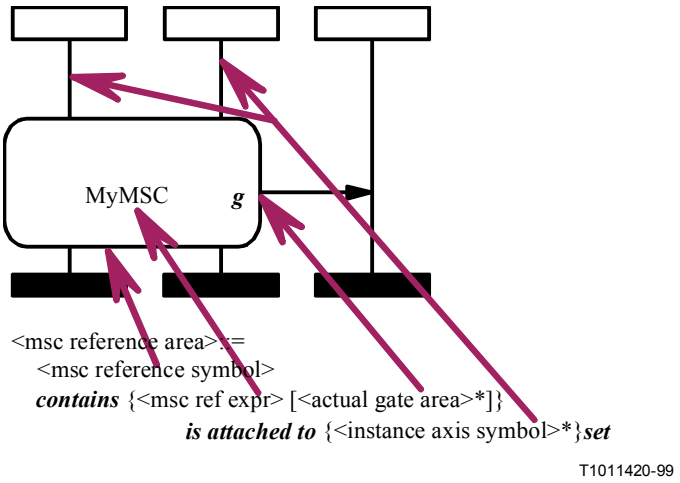


Figura 1/Z.120 – Ejemplo correspondiente a 'contains'

is followed by

"<area1> ***is followed by*** <area2>" significa que <area2> está relacionada lógica y geométricamente con <area1>.

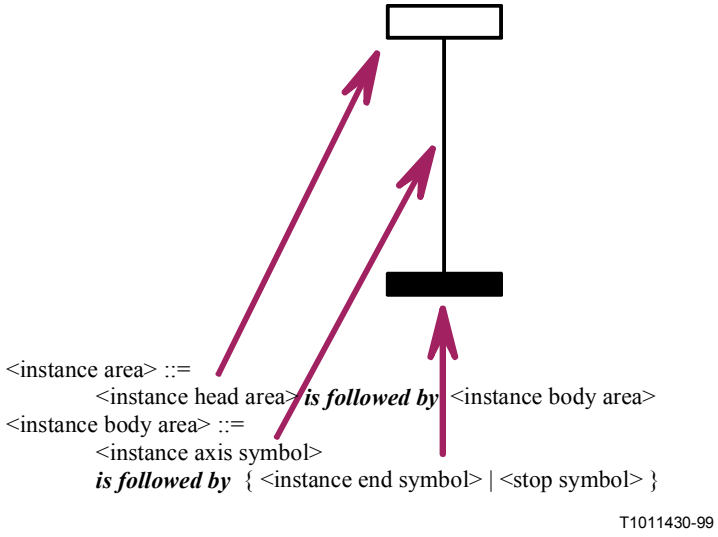


Figura 2/Z.120 – Ejemplo correspondiente a 'is followed by'

is associated with

"<area1> ***is associated with*** <area2>" significa que <area2> se expandirá a una cadena de texto que está afiliada con <area1>. No hay ninguna asociación geométrica más estrecha entre las áreas que la que indica que deben considerarse asociadas entre sí.

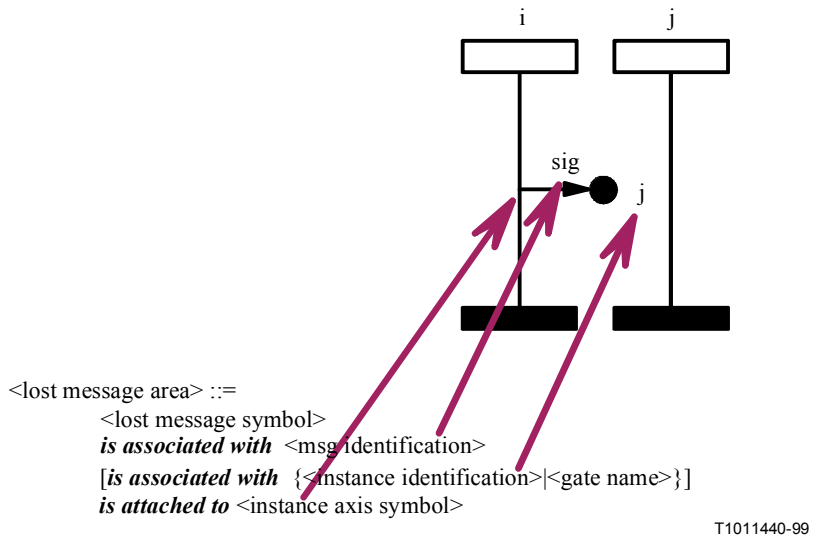


Figura 3/Z.120 – Ejemplo correspondiente a 'is associated with'

is attached to

"<area1> ***is attached to*** <area2>" no es como una regla de producción BNF normal y su utilización está restringida. <area2> debe ser un símbolo o un conjunto de símbolos. El significado de la regla de producción "<P>::=<area1> ***is attached to*** <area2>" es que si un no terminal <P> se expande utilizando esta regla, únicamente se produce el símbolo <area1>. En vez de producir también <area2>, se identifica una aparición de <area2> que es producida por una de las otras producciones. El resultado del constructivo ***is attached to*** es una relación lógica y geométrica entre <area1> y el símbolo <area2> procedente de la aparición implicada. Cuando el lado de la derecha es un conjunto de símbolos, hay una relación entre <area1> y todos los elementos del conjunto.

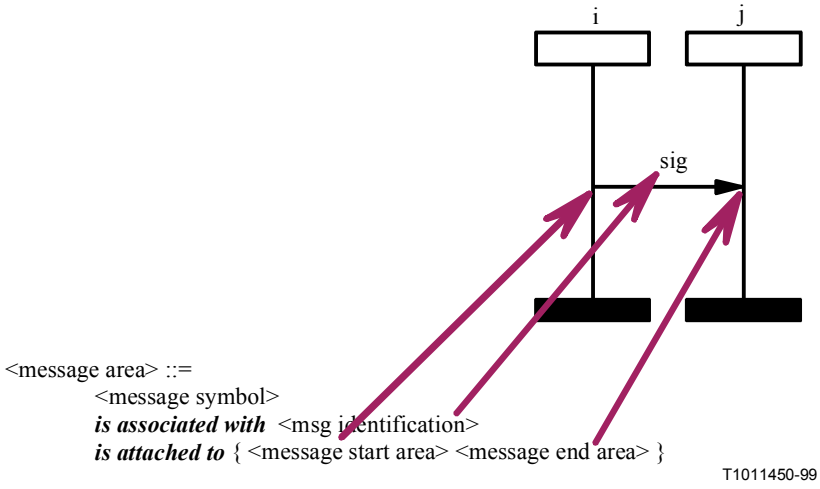


Figura 4/Z.120 – Ejemplo correspondiente a 'is attached to'

Obsérvese que si símbolo A ***is attached to*** símbolo B, símbolo B ***is attached to*** también a símbolo A. En consecuencia, una ligadura entre dos símbolos siempre se define dos veces en la gramática. Para ver esto, vamos seguir a uno de los no terminales del ejemplo anterior:

<message end area> ::=
<message in area> | <gate in area> | <gate out def area> | <inline expr>

<message in area> ::=
<message in symbol>
is attached to <instance axis symbol>
is attached to <message symbol>

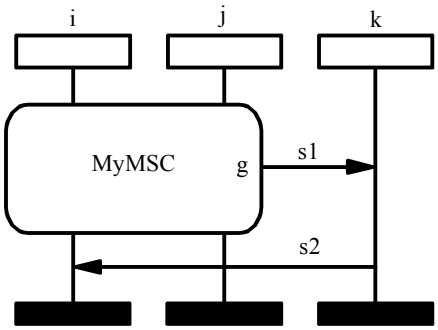
<message in symbol> ::=
<void symbol>

Por tanto, un <message symbol> ***is attached to*** un <message in symbol> y viceversa.

El constructivo "***is attached to***" puede ser especializado agregándole como prefijo o como sufijo indicaciones de los lugares a que se liga el constructivo. Los modificadores pueden ser ***top***, ***bottom***, ***left***, ***right*** o una combinación de éstos. Ello significa que se puede tener constructivos como "<area1> ***bottom is attached to top or right*** <area2>" lo que significa que la parte inferior de <area1> se liga lógica y geométricamente a la parte superior o derecha de <area2>.

above

"<area1> ***above*** <area2>" significa que <area1> y <area2> están ordenadas lógicamente. Esto se expresa geométricamente ordenando verticalmente <area1> y <area2>. Si dos <area> tienen la misma coordenada vertical no se define ninguna ordenación entre ellas, salvo que una salida de un mensaje debe aparecer antes de una entrada.



<event layer> ::=
<event area> | <event area> ***above*** <event layer>

T1011460-99

Figura 5/Z.120 – Ejemplo correspondiente a 'above'

La figura 5 describe una secuencia de eventos. Cuando los ejemplos están en diferentes ejemplares, la coordenada vertical geométrica no tiene significado semántico. La figura 5 describe la siguiente secuencia:

i,j: ***reference*** mr1: MyMSC ***gate*** g ***output*** s1 ***to*** k;

k: ***input*** s1 ***from*** mr1 ***via*** g;

k: ***output*** s2 ***to*** i;

i: ***input*** s2 ***from*** k.

set

El metasímbolo ***set*** es un operador posfijo que actúa sobre los elementos sintácticos inmediatamente precedentes encerrados entre llaves e indica un conjunto (no ordenado) de artículos. Cada artículo puede ser cualquier grupo de elementos sintácticos, en cuyo caso debe expandirse antes de aplicar el metasímbolo ***set***.

<text layer> ::=
{<text area>*} ***set***

es un conjunto de cero o más <text area>.

<msc body area> ::=
{ <instance layer> <text layer> <gate def layer>
<event layer> <connector layer> } ***set***

es un conjunto no ordenado de elementos dentro de las llaves.

2 Reglas generales

2.1 Reglas léxicas

Las reglas léxicas definen unidades léxicas. Las unidades léxicas son los símbolos terminales de la *sintaxis textual concreta*.

<lexical unit> ::=
<word>
| <character string>
| <special>
| <composite special>
| <note>
| <keyword>

<word> ::=
{<alphanumeric> | <full stop>}*
<alphanumeric>
{<alphanumeric> | <full stop>}*

<alphanumeric> ::=
<letter>
| <decimal digit>
| <national>

<letter> ::=
A | B | C | D | E | F | G | H | I | J | K | L | M
| N | O | P | Q | R | S | T | U | V | W | X | Y | Z
| a | b | c | d | e | f | g | h | i | j | k | l | m
| n | o | p | q | r | s | t | u | v | w | x | y | z

<decimal digit> ::=
0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<national> ::=
| ` | ¤ | @ | \
| <left square bracket>
| <right square bracket>
| <left curly bracket>
| <vertical line>
| <right curly bracket>
| <overline>
| <upward arrow head>

<left square bracket> ::=
[

<right square bracket> ::=
]

<left curly bracket> ::=
{

<vertical line> ::=
|

<right curly bracket> ::=
}

```
<left open> ::=
                    (
<left closed> ::=
                    <left square bracket>
<right open> ::=
                    )
<right closed> ::=
                    <right square bracket>
<abs time mark> ::=
                    @
<rel time mark> ::=
                    &
<overline> ::=
                    ~
<upward arrow head> ::=
                    ^
<full stop> ::=
                    .
<underline> ::=
                    _
<left angular bracket> ::=
                    <
<right angular bracket> ::=
                    >
<character string> ::=
                    <apostrophe>{<alphanumeric>
                |   <other character>
                |   <special>
                |   <full stop>
                |   <underline>
                |   <space>
                |   <apostrophe><apostrophe>}* <apostrophe>
<text> ::=
                {   <alphanumeric>
                |   <other character>
                |   <special>
                |   <full stop>
                |   <underline>
                |   <space>
                |   <apostrophe> }*
<apostrophe> ::=
                    '
<other character> ::=
                  ? | &   | %   | +   | -   | !   | /   | >   | *   | "   | <   | =
<special> ::=
                    (   | )   | ,   | ;   | :
<composite special> ::=
                    <qualifier left> | <qualifier right>
<qualifier left> ::=
                    <<
<qualifier right> ::=
                    >>
```

<note> ::=
/* <text> */

<name> ::=
{<letter> | <decimal digit> | <underline> | <full stop>}+

<keyword> ::=
action
| **after**
| **all**
| **alt**
| **as**
| **before**
| **begin**
| **block**
| **by**
| **call**
| **comment**
| **concurrent**
| **condition**
| **connect**
| **create**
| **data**
| **decomposed**
| **def**
| **empty**
| **end**
| **endafter**
| **endbefore**
| **endconcurrent**
| **endexpr**
| **endinstance**
| **endmethod**
| **endmsc**
| **endsuspension**
| **env**
| **equalpar**
| **escape**
| **exc**
| **expr**
| **external**
| **finalized**
| **found**
| **from**
| **gate**
| **in**
| **inf**
| **inherits**
| **inline**
| **inst**
| **instance**
| **int_boundary**
| **label**
| **language**
| **loop**
| **lost**
| **method**
| **msc**
| **mscdocument**
| **msg**
| **nestable**
| **nonnestable**
| **offset**
| **opt**
| **order**

| **otherwise**
| **out**
| **par**
| **parenthesis**
| **process**
| **receive**
| **redefined**
| **reference**
| **related**
| **replyin**
| **replyout**
| **seq**
| **service**
| **shared**
| **startafter**
| **startbefore**
| **starttimer**
| **stop**
| **stoptimer**
| **suspension**
| **system**
| **text**
| **time**
| **timeout**
| **timer**
| **to**
| **undef**
| **using**
| **utilities**
| **variables**
| **via**
| **virtual**
| **when**
| **wildcards**

<space> ::=
Carácter de espacio del Alfabeto N.º 5 del UIT-T

Los caracteres <national> se representan en la lista anterior según la Versión Internacional de Referencia del Alfabeto Internacional N.° 5 del CCITT (Recomendación T.50). La responsabilidad de la definición de representaciones nacionales de esos caracteres compete a los organismos de normalización nacionales.

Los caracteres de control se definen como en la Recomendación T.50. Una secuencia de caracteres de control puede aparecer donde pueda aparecer un <space>, y tiene el mismo significado que un <space>. El <space> representa el carácter de espacio del Alfabeto Internacional N.° 5 del CCITT.

La aparición de un carácter de control no es significativa en <note>. Un carácter de control no puede aparecer en literales de cadena de caracteres si su presencia es significativa.

En todas las <lexical unit>, las letras mayúsculas y la minúsculas son distintas, lo que significa que AB, ab, Ab y aB son cuatro unidades léxicas diferentes.

Una <lexical unit> es terminada por el primer carácter que no puede formar parte de la <lexical unit> según la sintaxis especificada anteriormente. Cuando un carácter <underline> va seguido de uno o más <space>, se pasan por alto todos esos caracteres (incluido el <underline>), esto es A_ B designa el mismo <name> que AB. Esta utilización de <underline> permite la división de varias <lexical unit> en más de una línea.

Cuando el carácter / va seguido inmediatamente de un * fuera de una <note>, inicia una <note>. El carácter * seguido inmediatamente del carácter / en una <note> siempre termina la <note>. Una <note> puede insertarse antes o después de cualquier <lexical unit>.

Una cadena de datos es una cadena de caracteres que será analizada por un analizador externo al MSC. El analizador léxico MSC necesita retornar la cadena de datos como un testigo (*token*), que a su vez se pasará al analizador externo.

Como sería preferible que la cadena de datos pudiera reconocerse a partir del MSC y a partir del lenguaje de datos sin un número de caracteres delimitadores adicionales, se definen conceptos en la gramática de MSC para que concuerden directamente con la mayor parte de los lenguajes de datos.

Se definen tres conceptos de paréntesis: los paréntesis normales, que pueden estar anidados, lo que significa que unos paréntesis pueden tener paréntesis anidados en su interior, los paréntesis no anidables en los que todo lo que se encuentre en su interior se considera no significativo, y los paréntesis simétricos en los que el paréntesis de apertura y el de cierre son iguales. Los delimitadores de paréntesis reales se declaran en el encabezamiento del documento MSC.

Para que el método sea aplicable en todas las circunstancias se define también un carácter de escape que indica cuándo los delimitadores de paréntesis no deben considerarse como tales sino de otra manera diferente.

<string> ::=
<pure data string>

<pure data string> ::=
<non-parenthesis> [<parenthesis>] [<pure data string>]

<parenthesis> ::=
<nestable par> | <equal par> | <non-nestable par>

<nestable par> ::=
<open par> <pure data string> <close par>

El <open par> y el <close par> deben ser delimitadores de paréntesis correspondientes definidos en un par de paréntesis en el encabezamiento del documento MSC definido en 5.2.

<non-parenthesis> ::=
{<non-par-non-escape> |
<escapechar> {<escapechar> | <open par> | <close par>} }*

<non-par-non-escape> ::=
character string containing no escape char and no paranthesis delimiters and
not the possible terminators","")";" ";" ";=" "=:" or **comment**

<equal par> ::=
<delim> <unmatched string> <delim>

<unmatched string> ::=
<non-parenthesis> [<nestable par>] [<unmatched string>]

<non-nestable par> ::=
<open par> <text> <close par>

El <open par> y el <close par> deben ser delimitadores de paréntesis correspondientes definidos en un par de paréntesis en el encabezamiento del documento MSC.

Si el <string> ha de contener los caracteres que de otro modo terminarían un <non-par-non-escape>, deben estar contenidos dentro de un paréntesis o bien se han de escapar.

2.2 Reglas de visibilidad y denominación

Las entidades son identificadas y referenciadas mediante nombres asociados. Las entidades se agrupan en clases de entidad para dar flexibilidad a las reglas de denominación. Existen las siguientes clases de entidad:

a) documento MSC;

b) MSC;

c) ejemplar;

d) condición;

e) temporizador;

f) mensaje;

g) puerta;

h) conector de intervalo de tiempo dividido;

i) variable;

j) nombre general.

Las unidades de ámbito son el sistema completo de documentos MSC, un documento MSC y un MSC.

El sistema completo de documentos MSC es el ámbito para definir nombres de documento MSC (que son equivalentes a nombres de clase de ejemplar).

El documento MSC es el ámbito para definir MSC, ejemplares, condiciones, temporizadores, mensajes, conectores de tiempo divididos, variables y nombres generales.

El MSC es el ámbito para definir puertas y parámetros formales de MSC. Los parámetros formales tienen precedencia sobre las entidades definidas en el documento MSC.

Un ejemplar que no tenga una clase explícita tomará como clase implícita una clase que será designada por el nombre del ejemplar.

Los mensajes que entran y salen de ejemplares descompuestos deben ser redeclarados con el mismo nombre y firma dentro del documento MSC que define la descomposición. Una firma de mensaje está constituida por los tipos y orden de los parámetros del mensaje.

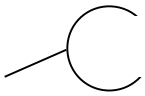
Una puerta es referenciada desde el exterior del MSC en que está definida, y solamente en una referencia a ese MSC.

En la notación textual a veces es necesario introducir identificadores específicos para objetos individuales que no necesitan una denominación explícita en la notación gráfica. Un ejemplo son las referencias de MSC, donde la notación gráfica puede tratar fácilmente dos o más apariciones de la misma referencia de MSC, mientras que la notación textual tiene que distinguirlas por identificadores individuales únicos.

Si bien en una descripción gráfica pueden utilizarse líneas y recurrir a la proximidad espacial para unir constructivos, en una notación textual necesitan identificadores para esta finalidad. En algunos casos, por ejemplo cuando se deba describir una relación que esté extensamente distribuida en el espacio, la notación gráfica también necesitará identificadores de entidades. Tales identificadores pueden encontrarse en un número de constructivo. La referencia de MSC contiene un identificador; las puertas se pueden identificar por nombres. Tales nombres, o bien están contenidos en un símbolo (como la referencia de MSC) espacialmente adyacente, descrito en la gramática gráfica como ***is associated with*** (como puertas de mensaje en un diagrama), o están representados por un símbolo de denominación especial que contiene el nombre, que está ligado al constructivo (como en el caso de áreas de eventos de instancia). Cuando las puertas no tienen un nombre explícito, se construye un nombre implícito.

<general name area> ::=
<general name symbol> ***contains*** <name>
is attached to {<instance event area> | <inline expression area> |
<operand area>}

<general name symbol> ::=



El <general name symbol> puede ser reflejado en espejo. El <nombre> contenido está escrito dentro del segmento de círculo, pero puede extenderse fuera del símbolo a través del extremo abierto. La <general name area> está ligada a otros constructivos mediante la línea recta.

2.3 Comentario

Hay tres clases de comentarios.

En primer lugar está la *note*, que aparece únicamente en la sintaxis textual (véanse las reglas léxicas).

En segundo lugar está *comment*, que es una notación para representar explicaciones informales asociadas con símbolos o texto.

La *sintaxis textual concreta* de los comentarios es:

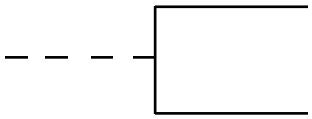
<end> ::=
[<comment>]**;**

<comment> ::=
comment <character string>

En la *gramática gráfica concreta* se utiliza la siguiente sintaxis:

<comment area> ::=
<comment symbol> ***contains*** <text>

<comment symbol> ::=



Un <comment symbol> puede ligarse a los símbolos gráficos.

En tercer lugar está el *text*, que puede utilizarse para comentarios globales.

El texto en la *gramática textual* se define por:

<text definition> ::=
text <character string> <end>

En la *gramática gráfica* se utiliza la siguiente sintaxis:

<text area> ::=
<text symbol> ***contains*** <text>

<text symbol> ::=

2.4 Reglas de dibujo

El usuario puede elegir el tamaño de los símbolos gráficos. Las fronteras de los símbolos no deben superponerse ni cruzarse. Son excepciones a esta regla:

a) cruce de símbolo de mensaje, símbolo de respuesta y símbolo de ordenación general con símbolo de mensaje, símbolo de respuesta, símbolo de ordenación general, líneas en símbolos de temporizador, símbolo de creación de línea, símbolo de eje de ejemplar,

símbolo de método, símbolo de suspensión, símbolo de corregión, línea de trazo discontinuo en un símbolo de comentario y en un símbolo de condición;

b) cruce líneas en símbolos de temporizador con símbolo de mensaje, símbolo de respuesta, símbolo de ordenación general, líneas en símbolos de temporizador, símbolo de creación de línea, línea de trazo discontinuo en un símbolo de comentario y en un símbolo de condición;

c) cruce de símbolo de mensaje, símbolo de respuesta y símbolo de ordenación general con el símbolo de expresión en línea o el símbolo de expresión en línea exc cuando los eventos de entrada (in) y de salida (out) de símbolo de mensaje, símbolo de respuesta y símbolo de ordenación general no están conectados al mismo eje de ejemplar;

d) cruce de símbolo creación de línea con símbolo mensaje, símbolo de respuesta, símbolo de ordenación general, líneas de símbolos de temporizador, símbolo de eje de ejemplar, símbolo de método, símbolo de suspensión, símbolo de corregión y línea de trazo discontinuo en símbolo de comentario;

e) cruce de símbolo de condición con símbolo de eje de ejemplar, símbolo de método, símbolo de corregión, símbolo de mensaje, símbolo de respuesta, símbolo de ordenación general, líneas en símbolos de temporizador;

f) cruce de símbolo de expresión en línea con símbolo de eje de ejemplar;

g) cruce de símbolo de referencia con símbolo de eje de ejemplar;

h) cruce de símbolo de acción con símbolo de eje de ejemplar en forma de línea y la superposición de símbolo de acción con símbolo de eje de ejemplar en forma de columna;

i) superposición de símbolo de método, símbolo de suspensión y símbolo de corregión con símbolo de eje de ejemplar;

j) superposición de símbolo de método con símbolo de método y símbolo de suspensión;

k) cruce de símbolo de línea de hmsc con símbolo de línea de hmsc.

El símbolo de eje de ejemplar y el símbolo de corregión pueden representarse en dos formas: en forma de línea única y en forma de columna. No se permite mezclar ambas formas en un ejemplar, salvo un eje de línea única con corregiones en forma de columna.

Si una condición compartida (véase 4.7) atraviesa un ejemplar que no interviene en esa condición, el eje de ejemplar se dibuja de modo que atraviese la condición.

Si una referencia compartida (véase 7.3) atraviesa un ejemplar que no interviene en esa referencia, el eje de ejemplar se dibuja de modo que atraviese la referencia.

Si una expresión en línea compartida (véase 7.2) atraviesa un ejemplar que no interviene en esa expresión en línea, el eje de ejemplar no puede dibujarse a través de la expresión de línea.

Cuando el símbolo de eje de ejemplar se representa en forma de columna, las fronteras verticales del símbolo de acción tienen que coincidir con las líneas de la columna.

Las líneas de mensaje pueden ser horizontales o inclinadas hacia abajo (con respecto al sentido de la flecha) y pueden estar formadas por una secuencia de segmentos rectos conectados.

Si un evento de entrada y un evento de salida están en el mismo punto de un eje de ejemplar, se considera que el evento de entrada está dibujado por encima del evento de salida. No se puede ligar una ordenación general a este punto. No se permite dibujar dos o más eventos de salida en el mismo punto. No se permite dibujar dos o más eventos de entrada en el mismo punto. Los siguientes eventos son eventos de entrada: entrada de mensaje, mensaje encontrado, recepción, recepción encontrada, respuesta de entrada, respuesta de entrada encontrada y expiración de periodo de temporización. Los siguientes eventos son eventos de salida: salida de mensaje, mensaje perdido, llamada, llamada perdida, respuesta de salida, respuesta de salida perdida, arranque de temporizador, parada de temporizador y creación de ejemplar.

2.5 Paginación de los MSC

Los MSC pueden ser particionados en varias páginas. La partición puede ser horizontal y vertical. Alternativamente, la partición puede contornearse mediante la composición de MSC o la descomposición de ejemplar (véase 7.4).

Si un MSC es particionado verticalmente en varias páginas, se repite el <msc heading> en cada página, pero el símbolo de fin de ejemplar únicamente puede aparecer en una página (en la "última" página del ejemplar en cuestión). Para cada ejemplar, la <instance head area> debe aparecer en la primera página en que comienza el ejemplar en cuestión y deberá repetirse en forma de trazo discontinuo en cada una de las páginas siguientes en las que continúa.

Si los mensajes, temporizadores, enunciados de creación o condiciones continúan de una página a la siguiente, el texto completo asociado al mensaje, temporizador, creación o condición deberá estar presente en la primera página y repetirse la totalidad o parte del texto en la página siguiente.

En las páginas de un MSC debe incluirse la numeración de páginas para indicar la posición correcta de las páginas. Las páginas deben numerarse con parejas de símbolos: "v-h", donde "v" es el número de página vertical y "h" el número de página horizontal. Se utilizarán numerales arábigos para los números de página verticales y letras mayúsculas del idioma inglés ('A' a 'Z') para los números de página horizontales. Si la gama 'A'-'Z' no es suficiente, se ampliará con las parejas 'AA' a 'AZ', 'BA' a 'BZ', etc.

3 Documento de gráficos de secuencias de mensajes

El documento de gráficos de secuencias de mensajes define una clase de ejemplar y la colección asociada de gráficos de secuencias de mensajes, que a su vez define un conjunto de trazos. Los documentos MSC pueden describirse gráficamente o textualmente. Puesto que el documento de gráficos de secuencias de mensajes define los ejemplares utilizados en los gráficos de secuencias de mensajes, los documentos de gráficos de secuencias de mensajes definen las estructuras de descomposición de ejemplar.

El encabezamiento del documento de gráfico de secuencias de mensajes contiene el nombre del documento y, facultativamente, después de la palabra clave **related to**, el identificador (nombre de trayecto) del documento a que se refieren los MSC. Este documento puede estar descrito en lenguaje de especificación y descripción (SDL), lenguaje de modelado unificado (UML, *unified modeling language*), notación combinada arborescente y tabular (TTCN), etc.

Gramática textual concreta

<textual msc document> ::=
<document head>
<textual defining part> <textual utility part>

<document head> ::=
mscdocument <instance kind> [**related to** <sdl reference>]
[<inheritance>] <end>
<using clause>
<containing-clause>
<message decl clause>
<timer decl clause>
[<data definition>]
[<parenthesis declaration>]

<textual defining part> ::=
<defining msc reference>*

<textual utility part> ::=
utilities [<containing-clause>] <defining msc reference>*

<defining msc reference> ::=
reference [<virtuality>] <msc name>

<virtuality> ::=
virtual | **redefined** | **finalized**

<using clause> ::=
{**using** <instance kind> <end> }*

<containing-clause> ::=
{**inst** <instance item>}+

<instance item> ::=
<instance name> [**:** <instance kind>] [<inheritance>]
[<decomposition>]
{ <dynamic decl list> | <end>}

<inheritance> ::=
inherits <instance kind>

<message decl clause> ::=
{**msg** <message decl> <end>}*

<timer decl clause> ::=
{**timer** <timer decl> <end>}*

<sdl reference> ::=
<sdl document identifier>

<identifier> ::=
[<qualifier>] <name>

<qualifier> ::=
<qualifier left> <text> <qualifier right>

El texto de un calificador no debe contener '<<' or '>>'.

La cláusula descompuesta en <instance item> no se aplica a listas utilizadas en <document head>, sino a la descomposición de ejemplares en los MSC de alto nivel.

Gramática gráfica concreta

<msc document area> ::=
<frame symbol> ***contains*** {<document head>
is followed by <defining part area> ***is followed by*** <utility part area>}

<defining part area> ::=
{{<defining msc reference area>*} ***set***} ***is followed by*** <separator area>

<utility part area> ::=
[<containing area> ***is followed by***] {<defining msc reference area>*} ***set***

<containing area> ::=
<containing-clause>

<defining msc reference area> ::=
<msc reference symbol>
contains [<virtuality>] <msc name>

Requisitos estáticos

Los ejemplares contenidos en los MSC referenciados a partir de la parte definidora y de la parte utilidad deberán escogerse únicamente de la lista de ejemplares de la <containing-clause> en el <document head> y el comienzo de la parte utilidad.

La cláusula contenedora facultativa de la parte utilidad incluirá ejemplares que están contenidos en los MSC de la parte utilidad, pero no de la parte definidora.

Los ejemplares contenidos en la <containing-clause> pueden ser **decomposed** en algunos de los MSC. Sea el ejemplar *x* contenido en MSC *m* **decomposed as** *xm*. Entonces, la parte definidora del documento MSC para *x* tiene que contener el MSC *xm*. Todos los ejemplares que son descompuestos en MSC tienen que ser declarados como descompuestos en las cláusulas contenedoras.

Cuando hay una cláusula **inherits** dentro de la cláusula contenedora, y la propia clase del ejemplar heredera está definida por un documento MSC, debe haber una cláusula **inherits** correspondiente de su <document head>.

La cláusula <virtuality> de la referencia de MSC definidor tiene que corresponder con el <msc heading> de la definición del MSC.

Un MSC **redefined** o **finalized** debe tener un MSC correspondiente en la clase de ejemplar heredada que no está **finalized** en la clase de ejemplar heredada.

Semántica

Un documento de gráficos de secuencias de mensajes es una colección de gráficos de secuencias de mensajes que facultativamente hacen referencia a un correspondiente documento en SDL.

El documento MSC define un ejemplar o una clase de ejemplar. Un ejemplar tiene asociado una colección de gráficos de secuencias de mensajes. Los MSC son escenarios de interacción entre los ejemplares contenidos en el ejemplar definido. La parte definidora (MSC definidores) define la colección de trazos MSC mientras que la parte utilidad contiene solamente patrones (MSC de utilidad) reutilizados por la parte definidora. No obstante, se permite que los MSC de la parte utilidad hagan referencia a los MSC de la parte definidora.

El significado de la cláusula **inherits** facultativa es lo mismo que:

1) incluir todas los ejemplares de la clase heredada en las cláusulas contenedoras de la clase heredera;
2) incluir todos los MSC definidores de la clase heredada en la parte definidora de la clase heredera;
3) incluir todos los MSC de utilidad de la clase heredada en la parte utilidad de la clase heredera;
4) reemplazar cada MSC **virtual** o cada MSC **redefined** de la clase heredada por el correspondiente MSC **redefined** o **finalized** de la clase heredera.

La cláusula **using** facultativa define la utilización de documentos MSC como bibliotecas de los MSC. El significado de la cláusula **using** es incluir la parte definidora de la biblioteca de MSC en la parte utilidad del documento MSC circundante.

Los nombres de ejemplares pueden ser calificados por nombres de ejemplares circundantes.

La palabra clave **decomposed** en la cláusula contenedora indica que el ejemplar se descompone en al menos un MSC.

4 MSC básico

4.1 Gráfico de secuencias de mensajes

Un gráfico de secuencias de mensajes, normalmente designado en forma abreviada por un MSC, describe un flujo de mensajes entre ejemplares. Un gráfico de secuencias de mensajes describe un comportamiento parcial de un sistema. Aunque el nombre *gráfico de secuencias de mensajes* se debe evidentemente a su representación gráfica, se utiliza tanto para la representación gráfica como para la textual.

Un MSC se describe mediante ejemplares y eventos o mediante una expresión que relaciona referencias de MSC sin citar explícitamente los ejemplares (véase 7.5).

Gramática textual concreta

<message sequence chart> ::=
[<virtuality>] **msc** <msc head> { <msc> | <hmsc> } **endmsc** <end>

<msc> ::=
<msc body>

<msc head> ::=
<msc name>[<msc parameter decl>] [<time offset>]<end>
[<msc inst interface>] <msc gate interface>

<msc parameter decl> ::=
([<data parameter decl>][<instance parameter decl>]
[<message parameter decl>][<timer parameter decl>])

<instance parameter decl> ::=
inst <instance parm decl list> <end>

<instance parm decl list> ::=
<instance parameter name> [: <instance kind>] [,<instance parm decl list>]

<instance parameter name> ::=
<instance name>

<message parameter decl> ::=
msg <message parm decl list>

<message parm decl list> ::=
<message decl list>

<timer parameter decl> ::=
timer <timer parm decl list>

<timer parm decl list> ::=
<timer decl list>

<msc inst interface> ::=
<containing-clause>

<instance kind> ::=
[<kind denominator>] <identifier>

<kind denominator> ::=
system | **block** | **process** | **service** | <name>

<msc gate interface> ::=
<msc gate def>*

<msc gate def> ::=
gate { <msg gate> | <method call gate> | <reply gate>|
<create gate> | <order gate> } <end>

<msg gate> ::=
<def in gate> | <def out gate>

<method call gate> ::=
<def out call gate> | <def in call gate>

<reply gate> ::=
<def out reply gate> | <def in reply gate>

<create gate> ::=
<def create in gate> | <def create out gate>

<order gate> ::=
<def order in gate> | <def order out gate>

<msc body> ::=
<msc statement>*

<msc statement> ::=
<text definition> | <event definition>

<event definition> ::=
<instance name> **:** <instance event list>
| <instance name list> **:** <multi instance event list>

<instance event list> ::=
<instance event> +

<instance event> ::=
<orderable event> | <non-orderable event>

<orderable event> ::=
[**label** <event name> <end>]
{ <message event> | <incomplete message event> |
<method call event> | <incomplete method call event> | <create> |
<timer statement> | <action> }
[**before** <order dest list>] [**after** <order dest list>] <end>
[**time** <time dest list> <end>]

La **label** <event name> opcional es utilizada cuando el evento sigue una relación de orden general.

<order dest list> ::=
<order dest> [, <order dest list>]

<time dest list> ::=
<time interval> [<time dest>] [, <time dest list>]

<time dest> ::=

<event name> | [**begin** | **end**] <reference identification>

La relación de tiempo para eventos ordenables se utiliza para expresar temporización. En el caso de temporización relativa, la temporización se da con respecto a un evento anterior. En la temporización absoluta se asigna el tiempo absoluto.

El <gate name> en <order dest> hace referencia a <def order out gate>, <actual order in gate>, <inline order out gate> o <def order out gate>. El <event name> en <order dest> hace referencia a un <orderable event>.

<non-orderable event> ::=
<start method> | <end method> | <start suspension> | <end suspension> |
<start coregion> | <end coregion> | <shared condition> |
<shared msc reference> | <shared inline expr> |
<instance head statement> | <instance end statement> | <stop>

<instance name list> ::=
<instance name> { **,** <instance name> }* | **all**

<multi instance event list> ::=
<multi instance event> +

<multi instance event> ::=
<condition> | <msc reference> | <inline expr>

Requisitos estáticos

Para cada <instance head statement> debe existir también un evento <instance end statement> o <stop> correspondiente. Para cada ejemplar no deben existir eventos antes de que se defina su <instance head statement>. Para cada ejemplar no debe haber eventos después de su <instance end statement>. Para cada ejemplar no debe haber más de un <instance head statement> ni tampoco más de un <instance end statement>.

Los ejemplares especificados dentro del <msc> o el <hmsc> deben ser un subconjunto de los ejemplares especificados en el <textual msc document> circundante.

La <containing-clause>, si está presente, debe contener los mismos ejemplares especificados dentro del <msc> o el <hmsc>.

Para <hmsc>, una descomposición de ejemplares debe describirse en la cláusula de descomposición de la <containing-clause> en el encabezamiento de MSC.

Gramática gráfica concreta

<msc diagram> ::=
<simple msc diagram> | <hmsc diagram>

<simple msc diagram> ::=
<msc symbol>
contains <msc heading> <msc body area>

<hmsc diagram> ::=
<msc symbol>
contains {<msc heading> [<containing-clause>]} <mscexpr area>

<msc symbol> ::=
<frame symbol> ***is attached to*** { <def gate area>* } ***set***

<frame symbol> ::=

<msc heading> ::=
msc <msc name>[<msc parameter decl>] [<time offset>]

<msc body area> ::=
{ <instance layer> <text layer> <gate def layer>
<event layer> <connector layer> } ***set***

<instance layer> ::=
{ <instance area>* } ***set***

<text layer> ::=
{ <text area>* } ***set***

<gate def layer> ::=
{ <def gate area>* } ***set***

<event layer> ::=
<event area> | <event area> ***above*** <event layer>

<connector layer> ::=
{ <message area>* | <incomplete message area>* |
<method call area>* | <incomplete method call area>*
<reply area>* | <incomplete reply area>*} ***set***

<event area> ::=
<instance event area>
| <shared event area>
| <create area>

<instance event area> ::=
{<message event area>
| <method call event area>
| <reply event area>
| <timer area>
| <concurrent area>
| <method area>
| <suspension area>
| <action area>}
[***is followed by*** <general name area>]

<shared event area> ::=
| <condition area>
| <msc reference area>
| <inline expression area>

Requisitos estáticos

Todos los mensajes y temporizadores con parámetros deben declararse en el documento MSC circundante. Los ejemplares utilizados en los MSC deben también definirse en el documento MSC circundante.

Las clases de ejemplar de los parámetros de ejemplar deben definirse en el documento MSC circundante. Las clases de ejemplares se definen por el hecho de que ejemplares de la clase especificada están contenidas en el documento MSC. Cuando la clase del ejemplar no se indica explícitamente en la declaración de parámetro, habrá de entenderse que se ha indicado cualquier clase.

Semántica

Un MSC describe la comunicación entre cierto número de componentes de sistema y entre estos componentes y el resto del mundo, denominado entorno. Para cada componente de sistema abarcado por un MSC hay un eje de ejemplar. La comunicación entre componentes de sistema se efectúa mediante mensajes. El envío y el consumo de mensajes son dos eventos asíncronos. Se supone que el entorno de un MSC es capaz de recibir mensajes del gráfico de secuencias de mensajes y enviar mensajes al gráfico de secuencias de mensajes; no se supone ninguna ordenación de los eventos de mensaje dentro del entorno.

Se supone un reloj global para un gráfico de secuencias de mensajes. El transcurso del tiempo se representa de arriba hacia abajo a lo largo de cada eje de ejemplar, pero no se supone ninguna escala de tiempo propia. Si no se introduce ninguna corregión o expresión en línea (véanse 7.1 y 7.2), se supone una ordenación total de eventos en el tiempo a lo largo de cada eje de ejemplar. Los eventos de ejemplares diferentes se ordenan mediante mensajes (para que un mensaje pueda ser consumido tiene que haber sido previamente enviado) o mediante el mecanismo de ordenación generalizada. Utilizando este mecanismo de ordenación generalizada, pueden ordenarse explícitamente los "eventos ordenables" de ejemplares diferentes (incluso en MSC diferentes). No se prescribe ningún otro tipo de ordenación. Por consiguiente, un gráfico de secuencias de mensajes impone una ordenación parcial al conjunto de eventos contenidos. Una relación binaria que es transitiva, antisimétrica e irreflexiva se denomina orden parcial.

Para las entradas de mensaje (etiquetadas por in(m*i*)) y las salidas de mensaje (etiquetadas por out(m*i*)) del gráficos de secuencias de mensajes de la figura 6 a) se obtiene la siguiente relación de ordenación: out(m2) < in(m2), out(m3) < in(m3), out(m4) < in(m4), in(m1) < out(m2) < out(m3) <in(m4), in(m2) < out(m4) junto con el cierre transitivo.

La ordenación parcial se puede describir en forma mínima (sin una representación explícita del cierre transitivo), mediante su gráfico de conectividad [figura 6 b)].

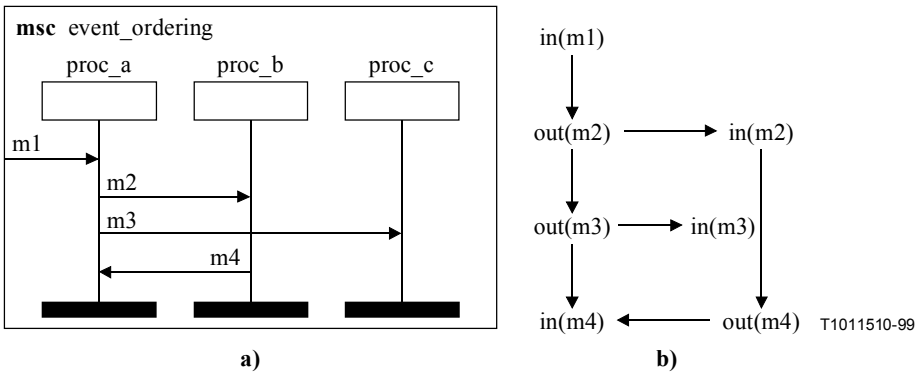


Figura 6/Z.120 – Gráfico de secuencias de mensajes y gráfico de conectividad correspondiente

En el anexo B se proporciona una semántica formal de MSC basada en álgebra de proceso. La semántica de un MSC se puede relacionar con la semántica de SDL mediante la noción de gráfico de alcanzabilidad. Cada secuenciación de un MSC describe un trazo de un nodo a otro nodo (o conjunto de nodos) del gráfico de alcanzabilidad que describe el comportamiento de una especificación de sistema SDL. El gráfico de alcanzabilidad consta de nodos y bordes. Los nodos designan estados del sistema global. Un estado del sistema global está determinado por los valores de las variables y el estado de ejecución de cada proceso, y por el contenido de las colas de mensajes. Los bordes corresponden a los eventos que son ejecutados por el sistema, por ejemplo el envío y el consumo de un mensaje o la ejecución de una tarea. Una secuencialización de un MSC designa una ordenación total de eventos compatible con la ordenación parcial definida por el MSC.

En la representación textual, la <event definition> la proporciona o bien un <instance name> seguido de la <event list> ligada, o <instance name list> seguida de un <multi instance event> ligado empleándose como separador el símbolo dos puntos (:). El no terminal <instance event> designa un evento que está ligado a un ejemplar individual, por ejemplo <action> o un objeto compartido, por ejemplo <shared condition>, empleándose la palabra clave **shared** junto con la lista de ejemplares o la palabra clave **all** para designar el conjunto de ejemplares entre las cuales se comparte la condición. Un objeto compartido puede representarse alternativamente por <multi instance event>. Se han introducido las notaciones diferentes para facilitar, por una parte una descripción *orientada a ejemplares* y por otra una descripción *orientada a eventos*. Ambas notaciones pueden combinarse arbitrariamente.

La descripción *orientada a ejemplares* indica eventos asociados a un ejemplar.

En la representación textual *orientada a eventos*, los eventos pueden indicarse en forma de una trazo de ejecución posible y no ordenarse con respecto a ejemplares.

La <msc interface> facultativa que describe la interfaz del MSC con su entorno consta de la <msc inst interface> y de la <msc gate interface>. La <msc inst interface> proporciona una declaración de los ejemplares, por ejemplo <instance name> y, facultativamente, <instance kind>. Como normalmente un MSC consta únicamente de una sección de una ejecución de sistema, la <msc inst interface> describe los puntos de conexión de ejemplares con el entorno. La <msc gate interface> proporciona una definición de mensaje y puertas de ordenación contenidas en el MSC. Las puertas de mensaje definen los puntos de conexión de mensajes con el entorno. Facultativamente, nombres de puertas pueden asociarse con puertas.

4.2 Ejemplar

Un gráfico de secuencias de mensajes se compone de ejemplares de entidades que interactúan. Un ejemplar de una clase de ejemplar tiene las propiedades de esta clase. Con relación a SDL, un ejemplar puede ser un sistema, bloque, proceso, o servicio representado en SDL. En el encabezamiento de un ejemplar, además del nombre del ejemplar puede especificarse el nombre de la clase, por ejemplo nombre de proceso.

En el cuerpo del ejemplar se especifica la ordenación de los eventos.

Gramática textual concreta

<instance head statement> ::=
instance [<instance kind>] [<decomposition>] <end>

<instance end statement> ::=
endinstance <end>

Gramática gráfica concreta

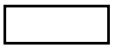
<instance area> ::=
<instance fragment area> [***is followed by*** <instance area>]

<instance fragment area> ::=
<instance head area> ***is followed by*** <instance body area>

<instance head area> ::=
<instance head symbol>
is associated with <instance heading>
[***is attached to*** <createline symbol>
[***is attached to***
{{<int symbol> | <abs time symbol>}***}*set***]]

<instance heading> ::=
<instance name> [[**:**] <instance kind>] [<decomposition>]

<instance head symbol> ::=

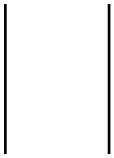


<instance body area> ::=
<instance axis symbol>
is followed by { <instance end symbol> | <stop symbol> }

<instance axis symbol> ::=
{ <instance axis symbol1> | <instance axis symbol2> }
is attached to { <event area>* } ***set***

<instance axis symbol1> ::=

<instance axis symbol2> ::=



<instance end symbol> ::=

El <instance heading> puede colocarse encima o dentro del <instance head symbol> o puede dividirse de modo que el <instance name> esté colocado dentro del <instance head symbol> mientras que la <instance kind> y <decomposition> se colocan por encima. Si el <instance heading> está dividido, el símbolo dos puntos (:) es facultativo.

Todos los fragmentos de un ejemplar con el mismo nombre deben colocarse directamente uno debajo de otro, cuando aparecen en la misma página.

Para indicar que una referencia de MSC contiene sólo creación de ejemplar o sólo parada de ejemplar, los extremos del fragmento deben estar ligados al símbolo de referencia de MSC.

Semántica

Los ejemplares se definen dentro del cuerpo del gráfico de secuencias de mensajes. El símbolo de fin de ejemplar determina el fin de una descripción del ejemplar dentro de un MSC. No describe la terminación del ejemplar (véase 4.11). Consiguientemente, el símbolo de cabecera de ejemplar determina el comienzo de una descripción del ejemplar dentro de un MSC. No describe la creación del ejemplar (véase 4.10). Todos los fragmentos de ejemplar con el mismo nombre constituyen el mismo ejemplar.

En el contexto de SDL, un ejemplar puede hacer referencia a un proceso (palabra clave **process**), a un servicio (palabra clave **service**) o a un bloque (palabra clave **block**), o a un sistema (palabra clave **system**). Fuera de SDL puede hacer referencia a cualquier clase de entidad. La definición de ejemplar proporciona una descripción de evento para entradas de mensaje y salidas de mensaje, llamadas y respuestas de métodos, acciones, condiciones compartidas y locales, eventos de temporizador, creación de ejemplar y parada de ejemplar. Fuera de las corregiones (véase 7.1) y de las expresiones en línea (véase 7.2) se supone una ordenación total de los eventos a lo largo de cada eje de ejemplar. Dentro de las corregiones no se supone ninguna ordenación de eventos si no se han prescrito ulteriores constructivos de sincronización en forma de relaciones de ordenación general.

4.3 Mensaje

Un mensaje dentro de un MSC es una relación entre una salida y una entrada. La salida puede provenir o bien del entorno (a través de una puerta) o de un ejemplar, o bien ser **found**; y una entrada o bien se dirige hacia el entorno (una puerta) o hacia un ejemplar, o es **lost**.

Un mensaje intercambiado entre dos ejemplares se puede dividir en dos eventos: la entrada de mensaje y la salida de mensaje; por ejemplo, el segundo mensaje de la figura 6 a) puede dividirse en out(m2) (salida) e in(m2) (entrada). En un mensaje se pueden asignar parámetros (véase 5.8).

La correspondencia entre salidas de mensaje y entradas de mensaje debe definirse de manera unívoca. En la representación textual, normalmente la correspondencia entre entradas y salidas se desprende de la identificación del nombre del mensaje y de la especificación de la dirección. En la representación gráfica, un mensaje se representa por una flecha.

La pérdida de un mensaje, es decir el caso en que se envía un mensaje pero no se consume, puede indicarse por la palabra clave **lost** en la representación textual y por un cuadrado negro en la representación gráfica.

De modo simétrico, un mensaje encontrado espontáneamente, es decir un mensaje que aparece sin provenir de ninguna parte, se define por la palabra clave **found** en la representación textual y por un cuadrado blanco en la representación gráfica.

Una ordenación de eventos de mensaje en ejemplares diferentes puede definirse mediante las palabras clave **before** y **after** en la representación textual. En la representación gráfica, los constructivos de sincronización en forma de líneas de conexión definen estos conceptos de ordenación generalizada.

El intervalo de tiempo en un mensaje temporizado da el retardo entre la salida del mensaje y la entrada del mensaje. Asimismo, la entrada de mensaje en el caso de una entrada de mensaje incompleta, y la salida de mensaje en el caso de una salida de mensaje incompleta, pueden ser sometidas a constricciones de tiempo. Además, los eventos salida de mensaje y entrada de mensaje pueden ser cometidos a constricciones de tiempo con respecto a otros eventos.

Gramática textual concreta

<message event> ::=
    <message output> | <message input>

<message output> ::=
    out <msg identification> **to** <input address>

<message input> ::=
    in <msg identification> **from** <output address>

<incomplete message event> ::=
    <incomplete message output> | <incomplete message input>

<incomplete message output> ::=
    out <msg identification> **to lost** [<input address>]

<incomplete message input> ::=
in <msg identification> **from found** [<output address>]

<msg identification> ::=
<message name> [**,** <message instance name>] [(<parameter list>)]

El <message instance name> sólo se necesita en la notación textual.

<output address> ::=
<instance name> | { **env** | <reference identification>} [**via** <gate name>]

<reference identification> ::=
reference <msc reference identification>
| **inline** <inline expr identification>

El <gate name> hace referencia a una <def in gate>. Si la palabra clave **env** se utiliza sola significa que <output address> designa una <def in gate> que tiene un nombre implícito dado por la correspondiente <msg identification> y sentido de flujo **in**.

<input address> ::=
<instance name> | { **env** | <reference identification>} [**via** <gate name>]

El <gate name> hace referencia a una <def out gate>. Si la palabra clave **env** se utiliza sola significa que <input address> designa una <def out gate> que tiene un nombre implícito dado por la correspondiente <msg identification> y sentido de flujo **out**.

Requisitos estáticos

Para los mensajes intercambiados entre ejemplares se deben cumplir las siguientes reglas: para cada <message output> hay que especificar un <message input> correspondiente y viceversa. Cuando <message name> y <address> no basten para establecer una correspondencia unívoca hay que utilizar el <message instance name>.

No se permite que la <message output> presente una dependencia causal con respecto a su <message input> a través de otros mensajes o constructivos de ordenación general. Tal dependencia causal se da cuando el gráfico de conectividad (véase la figura 6) contiene bucles. Si se especifica una <parameter list> para un <message input> habrá que especificarla también para el <message output> correspondiente y viceversa. La <parameter list> para <message output> consta solamente de <expression> y la <parameter list> para <message input> consta solamente de <pattern>.

Gramática gráfica concreta

<message event area> ::=
{ <message out area> | <message in area> }
{***is followed by*** <general order area> }*
{***is attached to*** <general order area> }*

Los eventos de mensajes pueden ser sometidos a una ordenación general según diferentes relaciones de orden general. Los eventos de mensaje aparecen a cada lado de la relación de orden.

<message out area> ::=
<message out symbol>
is attached to <instance axis symbol>
is attached to <message symbol>
[***is attached to***
{<int symbol> | <abs time symbol> }*]

<message out symbol> ::=
<void symbol>

<void symbol> ::= .

El <void symbol> es un punto geométrico sin extensión espacial.

El <message out symbol> es realmente un punto situado en el eje de ejemplar. El final de un símbolo de mensaje que carezca de cabeza de flecha está también situado en este punto del eje de ejemplar.

<message in area> ::=
<message in symbol>
is attached to <instance axis symbol>
is attached to <message symbol>
[***is attached to***
{<int symbol> | <abs time symbol> }*]

<message in symbol> ::=
<void symbol>

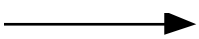
El <message in symbol> es realmente un punto situado en el eje de ejemplar. El final de un símbolo de mensaje que carezca de cabeza de flecha está también situado en este punto del eje de ejemplar.

<message area> ::=
<message symbol>
is associated with <msg identification> [**time** <time interval>]
is attached to {<message start area> | <message end area>}

<message start area> ::=
<message out area> | <actual out gate area>
| <def in gate area> | <inline gate area>

<message end area> ::=
<message in area> | <actual in gate area>
| <def out gate area> | <inline gate area>

<message symbol> ::=



Se permite la imagen especular de <message symbol>. El punto de la cabeza de flecha debe estar situado en el eje de ejemplar.

En la representación gráfica no es necesario el nombre de ejemplar de mensaje para una descripción sintáctica única.

<incomplete message area> ::=
{ <lost message area> | <found message area> }
{ ***is followed by*** <general order area> }*
{ ***is attached to*** <general order area> }*

<lost message area> ::=
<lost message symbol> ***is associated with*** <msg identification>
[***is associated with*** { <instance name> | <gate name> }]
is attached to <message start area>

<lost message symbol> ::=

El <lost message symbol> describe el evento del lado de salida, es decir la línea continua comienza en la <message start area> donde se produce el evento. El objetivo deseado facultativo del mensaje puede indicarse mediante un identificador asociado al símbolo. La identificación del objetivo debe escribirse cerca del círculo negro, mientras que la identificación del mensaje debe escribirse cerca de la flecha.

Puede también utilizarse la imagen especular del símbolo.

<found message area> ::=
<found message symbol> ***is associated with*** <msg identification>
[***is associated with*** { <instance name> | <gate name> }]
is attached to <message end area>

<found message symbol> ::=

El <found message symbol> describe el evento del lado de entrada (cabeza de flecha) que debe estar en una <message end area>. El ejemplar o puerta que, presuntamente, era el origen del mensaje se indica mediante la identificación facultativa proporcionada por el texto asociado al círculo del símbolo. La identificación del mensaje deberá escribirse cerca de la flecha.

Puede también utilizarse la imagen especular del símbolo.

Requisitos estáticos

Una <parameter list> de una <message area> donde ambos extremos están ligados a <instance event area> consta de <binding>. Una <parameter list> de una <message area> donde un extremo está ligado a un <output event area> y el otro está ligado a una puerta o está perdido consta de <expression>. Una <parameter list> de una <message area> donde un extremo está ligado a un <input event area> y el otro está ligado a una puerta o está encontrado, consta de <pattern>.

Semántica

En un MSC, la salida de mensaje designa el envío del mensaje desde la entrada del mensaje hasta el consumo del mensaje. No se proporciona ningún constructivo para la recepción de mensaje (entrada en la memoria tampón).

Un mensaje incompleto es un mensaje que es o bien una salida (cuya entrada se ha perdido o es desconocida), o bien una entrada (cuya salida ha sido encontrada o es desconocida).

Para el caso en que los eventos de mensaje coincidan con otros eventos, véanse las reglas de dibujo en 2.4.

4.4 Flujo de control

El MSC puede describir flujos de control, no sólo mediante mensajes asíncronos, sino también por medio de llamadas y respuestas.

Un método es un una unidad de comportamiento, provista de un nombre, dentro del ejemplar. Un método puede ser invocado a distancia y los resultados de los cálculos del método pueden ser retornados mediante una respuesta al llamante. La respuesta tendrá el mismo nombre que la llamada correspondiente.

Una llamada de método puede ser asíncrona o síncrona. Una llamada asíncrona implica que el llamante puede continuar sin necesidad de esperar la respuesta a la llamada. En cambio, una llamada síncrona implica que el llamante pasará a una región de suspensión donde no se producen eventos hasta que la llamada retorne. Como las llamadas de método pueden tener lugar en el interior de ejemplares descompuestos, los métodos y regiones de suspensión pueden omitirse. Si se utilizan el <method symbol> y el <suspension symbol>, el <method symbol> indica que un ejemplar está activo. El <suspension symbol> indica que un ejemplar está suspendido, por lo general en espera de que se resuelva alguna condición de bloqueo (por ejemplo en espera de la respuesta a una llamada síncrona) o para obtener acceso a algún recurso compartido (por ejemplo una CPU) con el fin de continuar la tarea en curso. El símbolo de eje de ejemplar normal (<instance axis symbol1> o <instance axis symbol2>) en este contexto significa que el ejemplar está inactivo y se encuentra en espera de que un evento de entrada (<message input> o <call in>) para activarse y comenzar las tareas que puede realizar. El símbolo que describe el nivel de activación de un ejemplar no implica ningún efecto dinámico sobre la semántica formal, que sólo considera la ordenación de los eventos, pero que sólo impone requisitos en cuanto al lugar en que los mensajes pueden tener su eventos de entrada y de salida.

Las llamadas de método y las respuestas de método pueden también estar incompletas, de tal modo que sea la llamada, sea la respuesta, se pierde.

Al igual que los mensajes sometidos a constricciones de tiempo, las llamadas de método y las respuestas de método pueden ser sometidas a constricciones de tiempo.

Gramática textual concreta

<method call event> ::=
<call out> | <call in> |<reply out> | <reply in>

<call out> ::=
call <msg identification> **to** <input address>

<call in> ::=
receive <msg identification> **from** <output address>

<reply out> ::=
replyout <msg identification> **to** <input address>

<reply in> ::=
replyin <msg identification> **from** <output address>

<incomplete method call event> ::=
<incomplete call out> | <incomplete call in> |
<incomplete reply out> | <incomplete reply in>

<incomplete call out> ::=
call <msg identification> **to lost** [<input address>]

<incomplete call in> ::=
receive <msg identification> **from found** [<output address>]

<incomplete reply out> ::=
replyout <msg identification> **to lost** [<input address>]

<incomplete reply in> ::=
replyin <msg identification> **from found** [<output address>]

<start method> ::=
method <end>

<end method> ::=
endmethod <end>

<start suspension> ::=
suspension <end>

<end suspension>::=
endsuspension <end>

Gramática gráfica concreta

<method call area> ::=
<message symbol>
is associated with <method identification>[**time** <time interval>]
is attached to { <method call start area> <method call end area> }
is attached to {<instance axis symbol>}
[***is attached to*** {<method area>}]

<method identification> ::=
call <msg identification>

<method call start area> ::=
<call out area> | <actual out gate area> |
<def in gate area> | <inline gate area>
is attached to <instance axis symbol>
is attached to <message symbol>
[***is attached to*** <suspension symbol>]
[***is attached to***
{<int symbol> | <abs time symbol> }*]

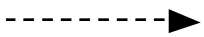
<method call end area> ::=
<call in area> | <actual in gate area> |
<def out gate area> | <inline gate area>
is attached to <instance axis symbol>
is attached to <message symbol>
[***is attached to*** <method symbol>]
[***is attached to***
{<int symbol> | <abs time symbol> }*]

<reply area> ::=
<reply symbol>
is associated with <msg identification>[**time** <time interval>]
is attached to { <reply start area> <reply end area>}

<reply start area> ::=
<reply out area> | <actual out gate area> |
<def in gate area> | <inline gate area>
is attached to <instance axis symbol>
is attached to <reply symbol>
[***is attached to*** <method symbol>]
[***is attached to***
{<int symbol> | <abs time symbol> }*]

<reply end area> ::=
<reply in area> | <actual in gate area> |
<def out gate area> | <inline gate area>
is attached to <instance axis symbol>
is attached to <reply symbol>
[***is attached to*** <suspension symbol>]
[***is attached to***
{<int symbol> | <abs time symbol> }*]

<reply symbol> ::=



<incomplete method call area> ::=
{ <lost method call area> | <found method call area> }
{ ***is followed by*** <general order area> }*
{ ***is attached to*** <general order area> }*

<lost method call area> ::=
<lost message symbol> ***is associated with*** <method identification>
[***is associated with*** { <instance name> | <gate name> }]
is attached to <method call start area>

<found method call area> ::=
<found message symbol> ***is associated with*** <method identification>
[***is associated with*** { <instance name> | <gate name> }]
is attached to <method call end area>

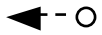
<incomplete reply area> ::=
{ <lost reply area> | <found reply area> }
{ ***is followed by*** <general order area> }*
{ ***is attached to*** <general order area> }*

<lost reply area> ::=
<lost reply symbol> ***is associated with*** <msg identification>
[***is associated with*** { <instance name> | <gate name> }]
is attached to <reply start area>

<lost reply symbol> ::=

<found reply area> ::=
<found reply symbol> ***is associated with*** <msg identification>
[***is associated with*** { <instance name> | <gate name> }]
is attached to <reply end area>

<found reply symbol> ::=



<method call event area> ::=
{<call out area> | <call in area>}
{***is followed by*** <general order area> }*
{***is attached to*** <general order area> }*

<call out area> ::=
<call out symbol>
is attached to <instance axis symbol>
is attached to <message symbol>

<call out symbol> ::=
<void symbol>

<call in area> ::=
<call in symbol>
is attached to <instance axis symbol>
is attached to <message symbol>

<call in symbol> ::=
<void symbol>

<reply event area> ::=
{<reply out area> | <reply in area>}
{***is followed by*** <general order area> }*
{***is attached to*** <general order area> }*

<reply out area> ::=
<reply out symbol>
is attached to <instance axis symbol>
is attached to <message symbol>

<reply out symbol> ::=
<void symbol>

<reply in area> ::=
<reply in symbol>
is attached to <instance axis symbol>
is attached to <message symbol>

<reply in symbol> ::=
<void symbol>

<method area> ::=
<method symbol>
is attached to <instance axis symbol>
[***contains*** <method event layer>]

<method event layer>::=
<method event area> | <method event area> ***above*** <method event layer>

<method event area>::=
<event area>

<method symbol> ::=

<suspension area> ::=
<suspension symbol>
is attached to <instance axis symbol>
[***contains*** <suspension event layer>]

<suspension event layer>::=
<method invokation area>
| <method invokation area> ***above*** <suspension event layer>

<method invokation area> ::=
<method start area>
is followed by <method area>
is followed by <method end area>

<method start area> ::=
<call in area> | <found method call area>

<method end area> ::=
<reply out area> | <lost reply area>

<suspension symbol> ::=

Requisitos estáticos

Las siguientes reglas gramaticales se refieren a eventos en un ejemplar dado:

<start method> <instance event list> <end method>

<start suspension> <instance event list> <end suspension>

<call out> <instance event list> <reply in>, donde la <reply in> corresponde a la <call out>.

<call in> <instance event list> <reply out>, donde la <reply out> corresponde a la <call in>.

Un <start method> sólo puede seguir directamente después de un evento <call in>, evento <incomplete call in>, evento <message input>, evento <incomplete message input>, <end suspension> o un evento <create> donde el ejemplar creado está sometido al <start method>. Un símbolo de método puede terminar en cualquier momento, pero un <end method> tiene siempre que seguir directamente a un evento <reply out> o a un evento <incomplete reply out>.

Desde el punto de vista dinámico, llamada y respuesta (si existe) deben aparecer en pares. Un determinado ejemplar que emite un evento <call out> no puede emitir otro evento llamada saliente antes de que se reciba un evento <reply in> correspondiente o antes de que se reciba un evento <call in> que presenta una dependencia causal con respecto a su evento <call out> (esto es, llamadas y respuestas pueden estar anidadas). Si una región de suspensión comienza directamente después de una llamada saliente, termina con la correspondiente respuesta.

Todos los <instance event> se permiten en símbolos de método. Las regiones de método y las regiones de suspensión pueden estar superpuestas. Por "superpuesto" ha de entenderse que las

regiones pueden dibujarse con un pequeño desplazamiento horizontal con respecto a la región de método o de suspensión en que está contenido.

No se permite que las regiones de suspensión contengan ningún evento. Sin embargo, a una región de suspensión se puede superponer a una región de método cuando la región de suspensión comienza con una <call out> y en consecuencia es llamada recursivamente con una <call in> (directa o indirectamente).

Una región de suspensión puede terminar en cualquier momento si no sigue directamente a un evento <call out>, en cuyo caso termina con el correspondiente evento <reply in>.[1]

4.5 Entorno y puertas

Las puertas representan la interfaz entre el MSC y su entorno. Cualquier mensaje o relación de orden ligado a la trama MSC constituye una puerta. Véase la figura 7 para ejemplos de puertas.

Toda puerta de mensaje siempre tiene un nombre. El nombre puede definirse explícitamente mediante un nombre asociado a la puerta en la trama. En otro caso puede darse el nombre de forma implícita mediante el sentido de flujo del mensaje a través de la puerta y el nombre del mensaje, por ejemplo "in_X" para una puerta que recibe un mensaje X de su entorno.

Se utilizan puertas de mensaje cuando referencias al MSC se sitúan en un contexto más amplio en otro MSC. Las puertas reales de la referencia al MSC se conectan entonces a otras puertas de mensajes o ejemplares. De forma similar a las definiciones de puertas, las puertas reales pueden tener nombres implícitos o explícitos.

Las puertas de orden representan relaciones de orden no completadas en las que un evento interior al MSC se ordenará con relación a un evento del entorno. Las puertas de orden siempre se denominan de forma explícita. Se considera que las puertas de orden tienen un sentido de flujo: desde el evento ordenado en primer lugar al evento que le sigue.

Las puertas de orden se usan también en referencias a MSC en otros MSC. Las puertas de orden reales de la referencia de MSC se conectan a otras puertas de orden o a eventos.

Las puertas de creación representan la posibilidad de separar el evento de creación del ejemplar creado.

Las puertas de las expresiones en línea son similares a las puertas en tramas de MSC y referencias de MSC, pero se diferencian por el hecho de que la trama de la expresión en línea sirve como trama de la definición de puerta (en el interior) y como símbolo de utilización de la puerta real (en el exterior). Cuando la puerta real en línea se dibuja directamente en otra trama (trama de expresión en línea o de diagrama MSC), las cabezas de flecha en los puntos de intersección pueden omitirse para evitar que los diagramas resulten demasiado cargados por una gran cantidad de cabezas de flechas muy próximas unas de otras.

---

[1] Requisitos estáticos adicionales pueden ser *añadidos* a los requisitos estáticos para adaptar a un determinado modelo de ejecución, o surgen como una consecuencia cuando se dibuja ese modelo. Por ejemplo, si se describe el flujo de control de un lenguaje puramente de procedimiento (esto es, sin paralelismo), un evento <call out> siempre tiene que ir seguido de una <start suspension>, o bien, si cada ejemplar puede siempre tratar eventos (es decir, tiene su propia CDU), una región de suspensión sólo puede comenzar tras un evento <call out>.

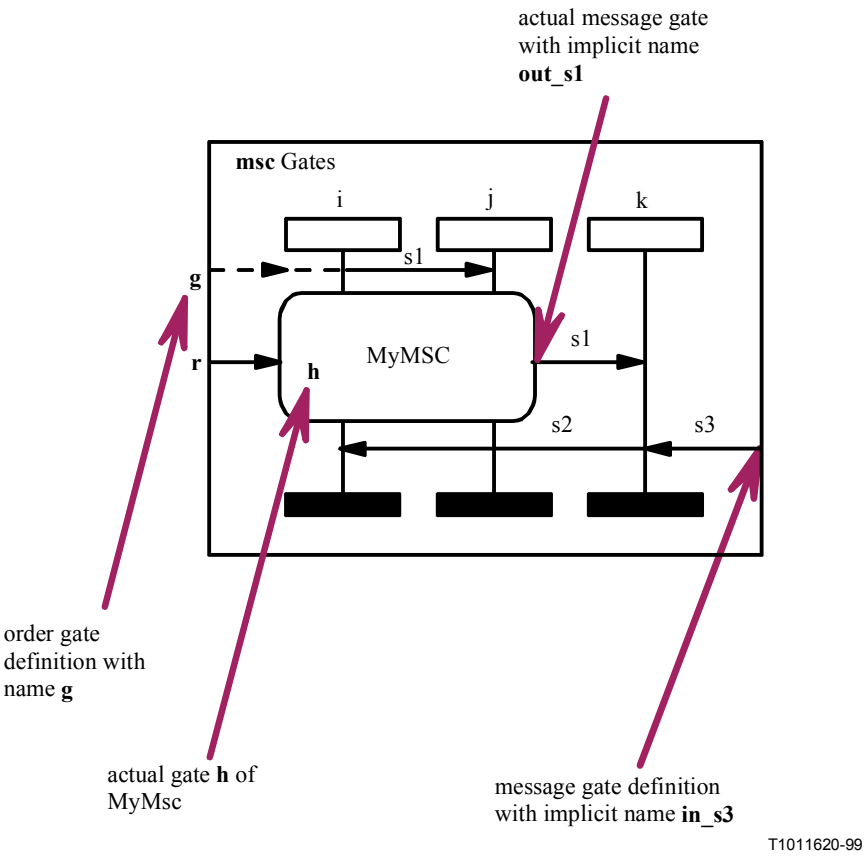


Figura 7/Z.120 – Ejemplo de puertas

Gramática textual concreta

<actual out gate> ::=
[<gate name>] **out** <msg identification> **to** <input dest>

<actual in gate> ::=
[<gate name>] **in** <msg identification> **from** <output dest>

<input dest> ::=
lost [<input address>] | <input address>

<output dest> ::=
found [<output address>] | <output address>

<def in gate> ::=
[<gate name>] **out** <msg identification> **to** <input dest>

<def out gate> ::=
[<gate name>] **in** <msg identification> **from** <output dest>

Si se omite <gate name> se supone un nombre implícito dado por el sentido de flujo y la <msg identification> correspondiente.

<actual order out gate> ::=
<gate name> **before** <order dest>

<order dest> ::=
<event name> | { **env** | <reference identification>} **via** <gate name>

El <event name> hace referencia a un evento ordenable. El <gate name> hace referencia a <def order out gate>, <actual order in gate>, <inline order out gate> o <inline order in gate>.

<actual order in gate> ::=
<gate name>
[**after** <order dest list>]

<def order in gate> ::=
<gate name> **before** <order dest>

El primer <gate name> define el nombre de esta puerta de orden.

<def order out gate> ::=
<gate name>
[**after** <order dest list>]

<actual create out gate> ::=
create out <create gate identification> **create** <create target>

<actual create in gate> ::=
create in <create gate identification>

<create target> ::=
<instance name> | { **env** | <reference identification>} [**via** <gate name>]

<def create in gate> ::=
create out [<create gate identification>] **create** <create target>

<def create out gate> ::=
create in <create gate identification>

Si se omite el <gate name> se supone un nombre implícito dado por el sentido de flujo y la <msg identification> correspondiente. El <gate name> define el nombre de esta puerta de orden.

<inline out gate> ::=
<def out gate>
[**external out** <msg identification> **to** <input dest>]

<inline in gate> ::=
<def in gate>
[**external in** <msg identification> **from** <output dest>]

<inline out call gate> ::=
<def out call gate>
[**external call** <msg identification> **to** <input dest>]

<inline in call gate> ::=
<def in call gate>
[**external receive** <msg identification> **from** <output dest>]

<inline out reply gate> ::=
<def out reply gate>
[**external replyout** <msg identification> **to** <input dest>]

<inline in reply gate> ::=
<def in reply gate>
[**external replyin** <msg identification> **from** <output dest>]

<inline create out gate> ::=
<def create out gate>
[**external** <create>]

<inline create in gate> ::=
<def create in gate>
[**external create from** <create source>]

<create source> ::=
<instance name> |
{**env** | <reference identification>} [**via** <create gate identification>]

<inline order out gate> ::=
<gate name>
[[**after** <order dest list>] **external before** <order dest>]

<inline order in gate> ::=
<gate name> **before** <order dest>
[**external** [**after** <order dest list>]]

<actual out call gate> ::=
[<gate name>] **call** <msg identification> **to** <input dest>

<actual in call gate> ::=
[<gate name>] **receive** <msg identification> **from** <output dest>

<def in call gate> ::=
[<gate name>] **call** <msg identification> **to** <input dest>

<def out call gate> ::=
[<gate name>] **receive** <msg identification> **from** <output dest>

Si se omite el <gate name> se supone un nombre implícito dado por el sentido de flujo y la <msg identification> correspondiente.

<actual out reply gate> ::=
[<gate name>] **replyout** <msg identification> **to** <input dest>

<actual in reply gate> ::=
[<gate name>] **replyin** <msg identification> **from** <output dest>

<def in reply gate> ::=
[<gate name>] **replyout** <msg identification> **to** <input dest>

<def out reply gate> ::=
[<gate name>] **replyin** <msg identification> **from** <output dest>

Si se omite el <gate name> se supone un nombre implícito dado por el sentido de flujo y la <msg identification> correspondiente.

Gramática gráfica concreta

<inline gate area> ::=
{ <inline out gate area> | <inline in gate area> |
<inline create out gate area> | <inline create in gate area> |
<inline out call gate area> | <inline in call gate area> |
<inline out reply gate area> | <inline in reply gate area> }
[***is associated with*** <gate identification>]

<inline out gate area> ::=
<void symbol>
is attached to <inline expression symbol>
[***is attached to*** { <message symbol>| <found message symbol> }]
[***is attached to*** { <message symbol> | <lost message symbol> }]

<inline in gate area> ::=
<void symbol>
is attached to <inline expression symbol>
[***is attached to*** { <message symbol> | <lost message symbol>}]
[***is attached to*** { <message symbol> | <found message symbol>}]

<inline out call gate area> ::=
<void symbol>
is attached to <inline expression symbol>
[***is attached to*** { <message symbol>| <found message symbol> }]
[***is attached to*** { <message symbol> | <lost message symbol> }]

<inline in call gate area> ::=
<void symbol>
is attached to <inline expression symbol>
[***is attached to*** { <message symbol> | <lost message symbol>}]
[***is attached to*** { <message symbol> | <found message symbol>}]

<inline out reply gate area> ::=
<void symbol>
is attached to <inline expression symbol>
[***is attached to*** { <reply symbol>| <found reply symbol> }]
[***is attached to*** { <reply symbol> | <lost reply symbol> }]

<inline in reply gate area> ::=
<void symbol>
is attached to <inline expression symbol>
[***is attached to*** { <reply symbol> | <lost reply symbol>}]
[***is attached to*** { <reply symbol> | <found reply symbol>}]

<inline create out gate area> ::=
<void symbol>
is attached to <inline expression symbol>
[***is attached to*** <createline symbol>]
[***is attached to*** <createline symbol>]

<inline create in gate area> ::=
<void symbol>
is attached to <inline expression symbol>
[***is attached to*** <createline symbol>]
[***is attached to*** <createline symbol>]

Una puerta de expresión en línea está normalmente ligada a un símbolo de mensaje situado dentro de la trama de expresión en línea y a un símbolo de mensaje fuera de la trama de expresión en línea. Cuando no hay ningún símbolo ligado al interior de la puerta, ello significa que hay un mensaje incompleto asociado a la puerta.

<inline order gate area> ::=
<inline order out gate area> | <inline order in gate area>

<inline order out gate area> ::=
<void symbol>
is attached to <inline expression symbol>
is attached to <general order symbol>
[***is attached to*** <general order symbol>]

El primer <general order symbol> es una relación de ordenación general interior a la expresión en línea tal que el evento interior a la expresión es anterior a la puerta. El <general order symbol> facultativo hace referencia a una relación de orden general ligada a la puerta fuera de la expresión en línea.

<inline order in gate area> ::=
<void symbol>
is attached to <inline expression symbol>
is attached to <general order symbol>
[***is attached to*** <general order symbol>]

El primer <general order symbol> es una relación de ordenación general interior a la expresión en línea tal que la puerta es anterior al evento interior a la expresión. El <general order symbol> facultativo se refiere a una relación de orden general ligada a la puerta fuera de la expresión en línea.

<def gate area> ::=
{ <def out gate area> |<def in gate area>|
<def order out gate area> | <def order in gate area>
<def out call gate area> | <def in call gate area>
<def out reply gate area> | <def in reply gate area>
<def create out gate area> | <def create in gate area>}
is attached to <msc symbol>

<def out gate area> ::=
<void symbol> [***is associated with*** <gate identification>]
is attached to { <message symbol> | <found message symbol> }

El <message symbol> o el <found message symbol> deben tener el extremo de cabeza de flecha ligado al <def out gate area>.

<def in gate area> ::=
<void symbol>[***is associated with*** <gate identification>]
is attached to { <message symbol> | <lost message symbol> }

El <message symbol> o el <lost message symbol> deben tener su extremo abierto ligado al <def in gate area>.

def out call gate area> ::=
<void symbol> [***is associated with*** <gate identification>]
is attached to { <message symbol> | <found message symbol> }

El <message symbol> o el <found message symbol> deben tener su extremo de cabeza de flecha ligado al <def out call gate area>.

<def in call gate area> ::=
<void symbol>[***is associated with*** <gate identification>]
is attached to { <message symbol> | <lost message symbol> }

El <message symbol> o el <lost message symbol> deben tener su extremo abierto ligado al <def in call gate area>.

<def out reply gate area> ::=
<void symbol> [***is associated with*** <gate identification>]
is attached to { <reply symbol> | <found reply symbol> }

El <reply symbol> o el <found reply symbol> deben tener su extremo de cabeza de flecha ligado al <def out reply gate area>.

<def in reply gate area> ::=
<void symbol>[***is associated with*** <gate identification>]
is attached to { <reply symbol> | <lost reply symbol> }

El <reply symbol> o el <lost reply symbol> deben tener su extremo abierto ligado al <def in reply gate area>.

<def order out gate area> ::=
<void symbol>[***is associated with*** <gate identification>]
is attached to <general order area>

<def order in gate area> ::=
<void symbol>[***is associated with*** <gate identification>]
is followed by <general order area>

<def create out gate area> ::=
<void symbol> [***is associated with*** <create gate identification>]
is attached to <createline symbol>

El <createline symbol> debe tener su extremo de cabeza de flecha ligado al <def create out gate area>.

<def create in gate area> ::=
<void symbol>[***is associated with*** <create gate identification>]
is attached to <createline symbol>

El <createline symbol> debe tener su extremo abierto ligado al <def create in gate area>.

<gate identification> ::=
<gate name>

<create gate identification> ::=
[<gate identification> :] <instance kind>

La <instance kind> hace referencia a la clase del ejemplar que se va a crear. La clase del ejemplar puede también corresponder a un <instance name> singular.

<actual gate area> ::=
<actual out gate area> | <actual in gate area> |
<actual out call gate area> | <actual in call gate area> |
<actual out reply gate area> | <actual in reply gate area> |
<actual create out gate area> | <actual create in gate area> |
<actual order out gate area> | <actual order in gate area>

<actual out gate area> ::=
<void symbol> [***is associated with*** <gate identification>]
is attached to <msc reference symbol>
[***is attached to*** { <message symbol> | <lost message symbol> }]

El <actual out gate area> está ligado al extremo abierto del <message symbol> o del <lost message symbol>.

<actual in gate area> ::=
<void symbol> [***is associated with*** <gate identification>]
is attached to <msc reference symbol>
[***is attached to*** { <message symbol> | <found message symbol> }]

El <actual in gate area> está ligado al extremo de cabeza de flecha del <message symbol> o del <found message symbol>.

<actual out call gate area> ::=
<void symbol> [***is associated with*** <gate identification>]
is attached to <msc reference symbol>
[***is attached to*** { <message symbol> | <lost message symbol> }]

El <actual out call gate area> está ligado al extremo abierto del <message symbol> o del <lost message symbol>.

<actual in call gate area> ::=
<void symbol> [***is associated with*** <gate identification>]
is attached to <msc reference symbol>
[***is attached to*** { <message symbol> | <found message symbol> }]

El <actual in call gate area> está ligado al extremo de cabeza de flecha del <message symbol> o del <found message symbol>.

<actual out reply gate area> ::=
<void symbol> [***is associated with*** <gate identification>]
is attached to <msc reference symbol>
[***is attached to*** { <reply symbol> | <lost reply symbol> }]

El <actual out reply gate area> está ligado al extremo abierto del <reply symbol> o del <lost reply symbol>.

<actual in reply gate area> ::=
<void symbol> [***is associated with*** <gate identification>]
is attached to <msc reference symbol>
[***is attached to*** { <reply symbol> | <found reply symbol> }]

El <actual in reply gate area> está ligado al extremo de cabeza de flecha del <reply symbol> o del <found reply symbol>.

<actual create out gate area> ::=
<void symbol> [***is associated with*** <create gate identification>]
is attached to <msc reference symbol>
[***is attached to*** <createline symbol>]

El <actual create out gate area> está ligado al extremo abierto del <createline symbol>.

<actual create in gate area> ::=
<void symbol> [***is associated with*** <create gate identification>]
is attached to <msc reference symbol>
[***is attached to*** <createline symbol>]

El <actual create in gate area> está ligado al extremo de cabeza de flecha del <createline symbol>.

<actual order out gate area> ::=
<void symbol> [***is associated*** with <gate identification>]
is attached to <msc reference symbol>
is followed by <general order area>

<actual order in gate area> ::=
<void symbol>[***is associated*** with <gate identification>]
is attached to <msc reference symbol>
is attached to <general order area>

Requisitos estáticos

Se permite que los <gate name> definidos de un MSC sean ambiguos, pero las referencias a <gate name> ambiguas son ilegales.

Para conexiones de puerta, la definición de puerta asociada debe corresponder con la puerta real. Correspondencia significa, a los efectos de mensajes, que el mensaje ligado a la definición de puerta es del mismo tipo que el mensaje ligado a la puerta real. Los sentidos de flujo deben también corresponder.

La clase de la definición de puerta debe ser la misma que la de la puerta real conectada. Las clases de puertas son: puertas de mensaje, puertas de ordenación, y puertas de creación.

4.6 Ordenación general

Se utiliza la ordenación general para imponer ordenaciones adicionales a eventos que no están definidos por la ordenación normal dada por la semántica del MSC. Por ejemplo, para especificar que un evento en un ejemplar debe aparecer antes que otro evento en otro ejemplar, con el que no guarda ninguna relación, o para especificar ordenaciones de eventos contenidos en una corregión.

Gramática textual concreta

La gramática textual se indica en 4.1.

Gramática gráfica concreta

<general order area> ::=
<general order symbol>
is attached to <ordered event area>

<general order symbol> ::=
<general order symbol1> | <general order symbol2>

<general order symbol1> ::=

Se autoriza que el <general order symbol1> tenga una forma escalonada. No puede haber escalones ascendentes y descendentes en la misma ordenación. Ello implica que puede estar constituido por segmentos verticales y horizontales consecutivos. Los segmentos de <general order symbol1> pueden superponerse, pero ello únicamente se autoriza si puede determinarse de forma inequívoca qué <general order symbol1> intervienen. Esto significa que no pueden estar implicados nuevos órdenes.

El <general order symbol1> únicamente puede tener lugar dentro de un <instance axis symbol2> (forma de columna). Las líneas de conexión desde el <general order symbol1> a las <ordered event area> ordenadas por aquél deben ser horizontales.

<general order symbol2> ::=

El <general order symbol2> puede tener cualquier orientación y también estar curvado.

<ordered event area> ::=
<actual order in gate area>
| <actual order out gate area>
| <def order in gate area>
| <def order out gate area>
| <inline order gate area>
| <message event area>
| <incomplete message area>
| <method call event area>
| <incomplete method call area>
| <reply event area>
| <incomplete reply area>
| <timer area>
| <create area>
| <action symbol>

Requisitos estáticos

El orden parcial de los eventos definidos por los constructivos de ordenación general y los mensajes debe ser irreflexivo.

Semántica

Los símbolos de orden general describen órdenes entre eventos que de otra forma no estarían ordenados. Esto es particularmente útil cuando eventos de una corregión deben estar más ordenados que todas las permutaciones posibles.

En la notación gráfica, cuando se usa el <general order symbol1> el evento que está gráficamente más alto debe aparecer antes que el evento más bajo. Cuando se utiliza el <general order symbol2>, la flecha apunta de un evento que aparece primero a otro que aparece después.

La gramática textual define las relaciones de orden parcial por las palabras clave **before** y **after**. Estas palabras indican directamente en qué orden los eventos que intervienen deben aparecer en los trazos válidos. Las relaciones de temporización son definidas por la palabra clave **time**. Indican en qué orden los eventos que intervienen deben aparecer en los trazos válidos.

Ejemplo de ordenación general

En la figura 8 se representan cuatro ejemplos que describen la misma ordenación general. Están implicados los siguientes órdenes:

a **before** b, a **before** d, c **before** b, c **before** d. a y c no están ordenados, y b y d no están ordenados.

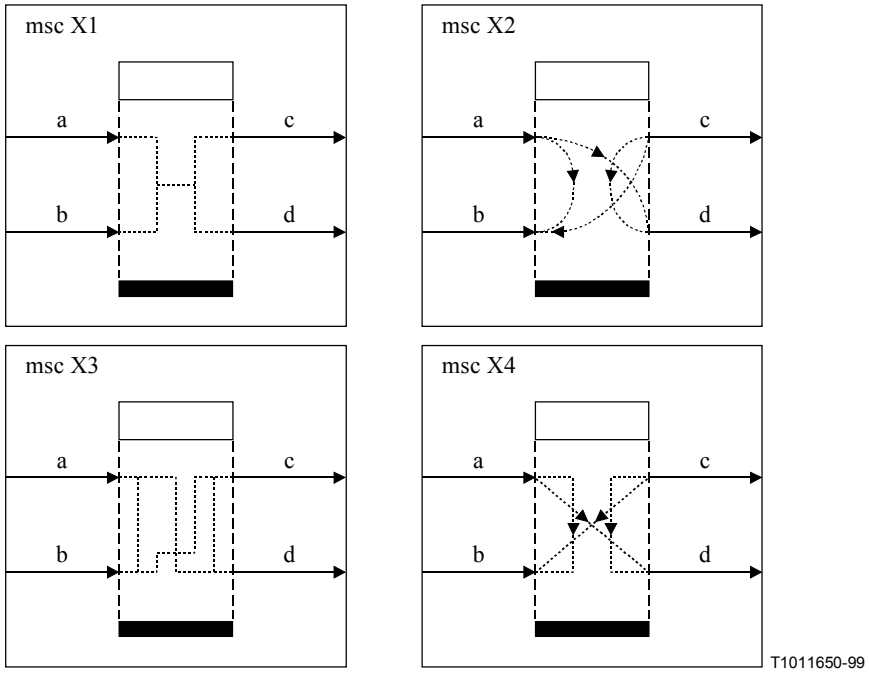


Figura 8/Z.120 – Ejemplo de ordenación general

4.7 Condición

Pueden utilizarse condiciones para restringir los trazos que un MSC puede efectuar, o la forma en que se componen en los MSC de alto nivel. Hay dos tipos de condición: los condiciones de *setting* (condiciones de fijación) y las condiciones de *guarding* (condiciones de guarda). Las condiciones de fijación (*setting*) fijan o describen el estado del sistema global actual (condición global) o algún estado no global (condición no global). En este último caso la condición puede ser local, esto es, está ligada a una y solamente un ejemplar.

Las condiciones de guarda (*guarding*) restringen el comportamiento de un MSC al permitir solamente la ejecución de eventos en una determinada parte del MSC en función de sus valores. Esta parte, que es el alcance de la condición es, o bien el MSC completo, o el operando más interior de una expresión en línea o una rama de un HMSC. La guarda debe colocarse al comienzo de este alcance. La condición es o bien un estado (global o no global), en el que debe estar el sistema (según lo definido por condiciones de fijación precedentes), o una expresión booleana en el lenguaje de datos.

En la representación textual, la condición tiene que estar definida para cada ejemplar a que está ligada mediante la palabra clave **condition** junto con el nombre de la condición. Si la condición hace referencia a más de un ejemplar, hay que añadir la palabra clave **shared** junto con la lista de ejemplares en las que se hace referencia a la condición. Una condición global, esto es, una condición que hace referencia a todas los ejemplares, puede definirse mediante las palabras clave **shared all**. Las condiciones de guarda se muestran con las palabras clave **condition** y **when**; las condiciones de fijación se muestran solamente con la palabra clave **condition**. En la sintaxis gráfica, la palabra clave **when** se incluye en el símbolo de condición para condiciones de guarda; para condiciones de fijación no hay una palabra clave o símbolo adicionales.

Se pueden utilizar condiciones de guarda por ejemplo en expresiones en línea (véase 7.2) para determinar qué operando de una expresión **alt** es aplicable según las opciones que se hayan elegido en anteriores expresiones **alt**, o en HMSC (véase 7.5).

Gramática textual concreta

<shared condition> ::=
[<shared>] <condition identification> <shared> <end>

<condition identification> ::=
condition <condition text>

<condition text> ::=
<condition name list> | **when** { <condition name list> | (<expression>) } |
otherwise

<condition name list> ::=
<condition name>{ , <condition name> }*

<shared> ::=
shared { [<shared instance list>] | **all** }

<shared instance list> ::=
<instance name> [**,** <shared instance list>]

<condition> ::=
[<shared>] <condition identification> <end>

El <shared> facultativo antes de <condition identification> en la gramática textual concreta es análogo al <shared> facultativo en la gramática gráfica concreta (véase más adelante).

Requisitos estáticos

Para cada <instance name> contenido en una <shared instance list> de una <condition> debe especificarse un ejemplar con una <condition> compartida correspondiente. Si el ejemplar *b* está contenido en la <shared instance list> de una <condition> compartida ligada al ejemplar *a*, entonces el ejemplar *a* debe estar contenido en la <shared instance list> de la <condition> compartida correspondiente ligada al ejemplar *b*. Si el ejemplar *a* y el ejemplar *b* comparten la misma <condition>, entonces, para cada mensaje intercambiado entre estos ejemplares, el <message input> y el <message output> se deben colocar o bien ambos antes de la <condition>, o bien ambos después de ésta.

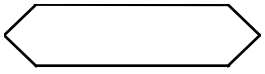
Si dos condiciones están ordenadas directamente (porque tienen un ejemplar en común) o indirectamente mediante condiciones en otros ejemplares, este orden debe respetarse en todos los ejemplares que comparten esas dos condiciones. De acuerdo con las posibilidades de una descripción de sintaxis *orientada a ejemplares* u *orientada a eventos* (véase 4.1), hay dos formas de sintaxis para condiciones: dentro de la descripción *orientada a ejemplares* se utiliza la forma <shared condition> mientras que en la descripción *orientada a eventos* se utiliza una forma de multiejemplar no terminal que emplea una <multi instance event list> seguida por un símbolo de dos puntos (:) y el no terminal <condition> (véase 4.1). En la forma <shared condition>, la palabra clave **shared** se utiliza también, de manera congruente, para condiciones locales; por tanto, dentro del no terminal <shared>, la <shared instance list> es facultativa.

Otherwise sólo puede aparecer como la guarda de exactamente un operando de una expresión alternativa.

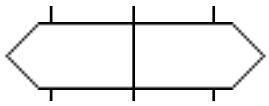
Gramática gráfica concreta

<condition area> ::=
<condition symbol> ***contains*** <condition text> [<shared>]
is attached to { <instance axis symbol>* } ***set***

<condition symbol> ::=



La <condition area> puede hacer referencia a un solo ejemplar o estar ligada a varios ejemplares. Si una <condition> compartida atraviesa un <instance axis symbol> que no interviene en esta condición, el <instance axis symbol> se dibuja de modo que atraviese la <condition>:



Los ejemplares <shared> no tienen <instance area> en el diagrama porque no hay eventos sobre estos ejemplares; no obstante, estos ejemplares <shared> siguen estando abarcados por la condición. Si hay ambigüedades en cuanto a la ordenación de los ejemplares <shared>, deberán representarse en el diagrama explícitamente con una <instance area>.

Requisitos estáticos

Una condición de guarda debe colocarse al comienzo de su alcance, donde un alcance es o bien un MSC completo, o un operando de una expresión en línea, o una rama de un HMSC.

Una condición de guarda debe cubrir siempre todas la instancias de su alcance. Una instancia está lista si contiene un evento que puede ejecutarse antes de cualquier otro evento del alcance.

Si una guarda contiene una expresión de datos, esta expresión debe ser del tipo Boolean. Si esta expresión contiene además variables dinámicas, sólo puede cubrir una única instancia que debe entonces ser la única instancia lista del alcance.

Semántica

Las condiciones de fijación definen el estado de sistema real del ejemplar o ejemplares que comparten la condición. Se pueden utilizar condiciones de guarda para limitar las formas posibles en que se puede continuar un MSC.

Los eventos en el alcance afectado por una condición de guarda sólo pueden ejecutarse si la condición de guarda se cumple en el momento en que se ejecuta el primero de estos eventos. Si la condición es una expresión en el lenguaje de datos, esto significa que se evalúa como "true". Si la guarda es un conjunto de nombres de condiciones, la *última* condición de fijación sobre ese conjunto de ejemplares debe tener una intersección no vacía con la guarda. Sólo se comprueban las condiciones de fijación sobre exactamente los mismos ejemplares; las condiciones que fijan el estado de un subconjunto o superconjunto de estos ejemplares no son comprobados.

En 7.2 se indican interpretaciones específicas de guardas en expresiones en línea. La guarda **otherwise** es *true* si todos los demás operandos de la expresión alternativa son *false.*

4.8 Temporizador

En los MSC se puede especificar la fijación de un temporizador y un subsiguiente transcurso del periodo de temporización debida a la expiración del temporizador, o la fijación de un temporizador y una parada subsiguiente del temporizador. Además, pueden utilizarse separadamente los constructivos de temporizador individuales – fijación, parada/expiración del periodo del temporizador – por ejemplo en el caso en que se reparta entre diferentes MSC la expiración del temporizador o la supervisión de tiempo. En la representación gráfica, el símbolo de comienzo tiene la forma de un reloj de arena conectado al eje de ejemplar por un símbolo de línea. La extinción del periodo de temporización se describe por una flecha de mensaje que apunta al ejemplar que está ligado al símbolo de reloj de arena. El símbolo de parada tiene la forma de una cruz conectada al ejemplar por una línea.

La especificación del nombre de ejemplar del temporizador y de la duración del temporizador es facultativa en las representaciones textual y gráfica.

Un temporizador se puede arrancar con o sin una duración. Cuando se arranca con una duración, existe la posibilidad de expresar un límite inferior y un límite superior para el instante en que habrá

de producirse la extinción del periodo de temporización. El límite superior para la extinción del periodo de temporización puede definirse de modo que sea el infinito, lo que se representa por la palabra clave **inf**. Los eventos arranque, parada y extinción del periodo de temporización de un temporizador pueden ser sometidas a constricciones de tiempo adicionales. Esta temporización se representa entonces en la relación **time** de eventos ordenables.

El periodo de temporización de un temporizador se define como sigue:

[0, **inf**),	if no duration is given
[<duration min>, **inf**),	if only the minimal duration is given
[0, <duration max>],	if only the maximal duration is given
[<duration min>, <duration max>],	if minimal and maximal duration are given

Gramática textual concreta

<timer statement> ::=
<starttimer> | <stoptimer> | <timeout>

<starttimer> ::=
starttimer <timer name> [**,** <timer instance name>]
[<duration>] [(<parameter list>)]

<duration> ::=
<left square bracket>
[<min durationlimit>] [, <max durationlimit>] <right square bracket>

<durationlimit> ::=
<expression string> | **inf**

<stoptimer> ::=
stoptimer <timer name> [**,** <timer instance name>]

<timeout> ::=
timeout <timer name> [**,** <timer instance name>]
[(<parameter list>)]

En el caso en que el <timer name> no baste para una correspondencia unívoca, deberá emplearse el <timer instance name>.

Requisitos estáticos

La <parameter list> para <starttimer> consta solamente de <expression> y la <parameter list> para <timeout> consta solamente de <pattern>.

Gramática gráfica concreta

<timer area> ::=
{ <timer start area> | <timer stop area> | <timeout area> }
{***is followed by*** <general order area> }*
{***is attached to*** <general order area> }*

<timer start area> ::=
<timer start area1> | <timer start area2>

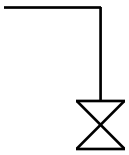
<timer start area1> ::=
<timer start symbol> ***is associated with*** <timer name>
[<duration>] [(<parameter list>)]
is attached to <instance axis symbol>
[***is attached to***
{<int symbol> | <abs time symbol> }*]
[***is attached to*** {<restart symbol> | <timer stop symbol2>
| <timeout symbol3> }]

<timer start area2> ::=
<restart symbol> ***is associated with*** <timer name>
[<duration>] [(<parameter list>)]
is attached to <instance axis symbol>
[***is attached to***

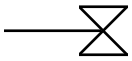
{<int symbol> | <abs time symbol> }*]
is attached to <timer start symbol>
[***is attached to*** { <timer stop symbol2> | <timeout symbol3> }]

<timer start symbol> ::=
<start symbol1> | <start symbol2>

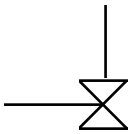
<start symbol1> ::=



<start symbol2> ::=



<restart symbol> ::=



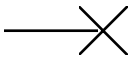
<timer stop area> ::=
<timer stop area1> | <timer stop area2>

<timer stop area1> ::=
<timer stop symbol1> ***is associated with*** <timer name>
is attached to <instance axis symbol>
[***is attached to***
{<int symbol> | <abs time symbol> }*]

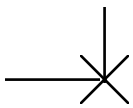
<timer stop area2> ::=
<timer stop symbol2> ***is associated with*** <timer name>
is attached to <instance axis symbol>
[***is attached to***
{<int symbol> | <abs time symbol> }*]
is attached to { <timer start symbol> | <restart symbol> }

<timer stop symbol> ::=
<timer stop symbol1> | <timer stop symbol2>

<timer stop symbol1> ::=



<timer stop symbol2> ::=

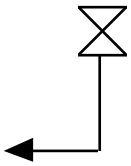


<timeout area> ::=
<timeout area1> | <timeout area2>

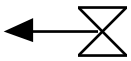
<timeout area1> ::=
<timeout symbol> ***is associated with*** <timer name>
[(<parameter list>)]
is attached to <instance axis symbol>
[***is attached to***
{<int symbol> | <abs time symbol> }*]

<timeout symbol> ::=
<timeout symbol1> | <timeout symbol2>

<timeout symbol1> ::=



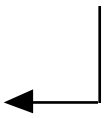
<timeout symbol2> ::=



<timeout area2> ::=

<timeout symbol3> [***is associated with*** <timer name>
(<parameter list>)]
is attached to <instance axis symbol>
[***is attached to***
{<int symbol> | <abs time symbol> }*]
is attached to { <timer start symbol> | <restart symbol> }

<timeout symbol3> ::=



Requisitos estáticos

La <parameter list> de un <timer start area> consta solamente de <expression>. La <parameter list> de un <timeout area > consta de <pattern>.

Semántica

Por arranque ha de entenderse la fijación del temporizador y por parada la anulación del temporizador. Por expiración del periodo de temporización ha de entenderse la extinción o consumo de la señal del temporizador.

Para el caso en que eventos de temporizador coinciden con otros eventos, véanse las reglas de dibujo en 2.4.

4.9 Acción

Además del intercambio de mensajes pueden especificarse acciones en los MSC. Una acción es un evento atómico que puede tener asociado sea texto informal, sea enunciados de datos formales. El texto informal se encierra entre comillas simples para distinguirlo de enunciados de datos. Los enunciados de datos consisten en una lista de enunciados separados por el símbolo coma (,), donde cada enunciado puede ser una vinculación, un enunciado de definir, o un enunciado de indefinir.

Gramática textual concreta

<action> ::=

action <action statement>

<action statement> ::=

<informal action> | <data statement list>

<informal action> ::=

<character string>

Gramática gráfica concreta

<action area> ::=

<action symbol>
is attached to <instance axis symbol>
[***top or bottom is attached to top or bottom***

{ {<int symbol> | <abs time symbol> }* } ***set***]
{ ***is followed by*** <general order area> }*
{ ***is attached to*** <general order area> }*
contains <action statement>

<action symbol> ::=

Cuando el eje de ejemplar tenga la forma de columna, la anchura de <action symbol> debe coincidir con la anchura de la columna.

Semántica

Una acción describe una actividad atómica interna de un ejemplar. Cuando una acción contiene enunciados de datos, el evento modifica el estado evaluando cada estado concurrentemente. Esta concurrencia refleja la atomicidad de una acción.

4.10 Creación de ejemplar

Al igual que en SDL, en los MSC se puede especificar la creación y terminación de ejemplares. Un ejemplar puede ser creado por otro ejemplar. No se puede ligar ningún evento de mensaje a un ejemplar antes de su creación.

Gramática textual concreta

<create> ::=
create <instance name> [(<parameter list>)]

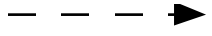
Para cada <create> debe existir un ejemplar correspondiente con el nombre especificado. Un ejemplar sólo puede crearse una vez, es decir dentro de un MSC *simple* no deben aparecer dos o más <create> con el mismo nombre.

Gramática gráfica concreta

<create area> ::=
<createline symbol> [***is associated with*** <parameter list>]
is attached to
{ {<instance axis symbol> | <def create in gate area> |
<actual create out gate area> }
{ <instance head area> | <def create out gate area> |
<actual create in gate area> }} ***set***
{ ***is followed by*** <general order area> }*
{ ***is attached to*** <general order area> }*
[***is attached to***
{{<int symbol> | <abs time symbol>}*}***set***]

El evento creación se representa por el final del <createline symbol> que no tiene cabeza de flecha. El evento creación está ligado a un eje de ejemplar. Si el <create area> está ordenado generalmente, esta ordenación se aplica al evento creación. Si se asigna una constricción de tiempo, se aplica al evento de creación. La cabeza de flecha apunta al <instance head symbol> del ejemplar creado.

<createline symbol> ::=



Se permite la imagen especular del <createline symbol>.

Semántica

Crear define la creación dinámica de un ejemplar por otro. Desde el punto de vista dinámico, en la vida de un ejemplar sólo puede haber una creación, y ninguno de los eventos relativos al ejemplar puede tener lugar antes de su creación.

Para el caso en que la creación del ejemplar coincide con otros eventos, véanse las reglas de dibujo en 2.4.

4.11 Parada de ejemplar

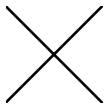
La parada de ejemplar es la contrapartida de la creación de ejemplar, con la salvedad de que un ejemplar sólo puede detenerse por sí mismo mientras que un ejemplar se crea por otro ejemplar.

Gramática textual concreta

<stop> ::=
stop <end>

Gramática gráfica concreta

<stop symbol> ::=



Semántica

La parada al final de un ejemplar representa la terminación de este ejemplar.

Desde el punto de vista dinámico, en la vida de un ejemplar sólo puede haber una parada, y ninguno de los eventos relativos al ejemplar puede tener lugar después de la parada del ejemplar.

5 Conceptos de datos

5.1 Introducción

Los datos se incorporan en MSC en varios lugares, tales como pasos de mensaje, casillas de acción, y referencias de MSC. Los datos se usan de dos maneras distintas: estáticamente, como en la parametrización de un diagrama MSC, o dinámicamente, como en la adquisición de un valor mediante la recepción de un mensaje. Para permitir la utilización de MSC con lenguajes elegidos por el usuario no se define ningún lenguaje específico de MSC. En cambio, en la definición del MSC hay puntos en los se requiere una cadena que representa algún aspecto de la utilización de datos, como expresiones o declaraciones de tipo. Para definir la semántica de MSC con datos se requieren funciones que extraen de esas cadenas la información necesaria para interactuar con los conceptos de datos MSC.

5.2 Interfaz de sintaxis con lenguajes de datos externos

Cuando se utiliza una cadena terminal para el lenguaje de datos, el nombre de producción va precedido de palabras subrayadas que sugieren su significado de la interfaz real. Por ejemplo <variable string> es una cadena terminal que habrá de ser analizada gramaticalmente en el lenguaje de datos como una variable. Las cadenas de datos terminales son las siguientes:

- <variable string>;
- <type ref string>;
- <data definition string>;
- <expression string>.

Para poder distinguir los datos en MSC se definen delimitadores de paréntesis y caracteres de escape. La sintaxis real para cadenas de datos se indica en 2.1. La idea es que las cadenas de datos en el lenguaje de datos sean reconocibles sin un uso excesivo de secuencias de escape y delimitadores especiales.

Gramática textual concreta

```
<parenthesis declaration> ::=
                        parenthesis <par decl list> <end>

<par decl list> ::=
                        { <nestable par pair> | <equal par delim> | <non-nestable par pair>
                        | <escape decl>}
                        [, <par decl list>]

<nestable par pair> ::=
                        nestable <left delim>, <right delim> <end>

<equal par delim> ::=
                        equalpar <left delim> <end>

<non-nestable par pair> ::=
                        nonnestable <left delim>, <right delim> <end>

<escape decl> ::=
                        escape <escapechar>

<left delim> ::=
                        <delim> <open par> <delim>

<right delim> ::=
                        <delim> <close par> <delim>

<delim> ::=
                        <apostrophe>
                    |   <alphanumeric>
                    |   <other character>
                    |   <special>
                    |   <full stop>
                    |   <underline>

<open par> ::=
                        <character string>

<close par> ::=
                        <character string>

<escapechar> ::=
                        <character string>
```

Los <delim> son iguales a ambos lados de los delimitadores izquierdo y derecho declarados. La regla de interpretación es ésta: el carácter elegido como delimitador no debe aparecer en el texto que define los parámetros de apertura y de cierre.

Semántica

La interfaz sintáctica da instrucciones sobre la manera en que el analizador léxico debe extraer cadenas de datos de la notación MSC en su conjunto. El analizador léxico recibe el principio de una cadena de datos y aplica las declaraciones de paréntesis para determinar cuándo habrá de terminar la cadena y pasarla al analizador del lenguaje de datos que es externo a la interpretación del MSC como tal.

Hay tres tipos de paréntesis y un mecanismo de escape. Para una explicación más detallada de los cuatro conceptos, véase 2.1. Un paréntesis anidable es un paréntesis en el que la cadena entre los delimitadores izquierdo y derecho puede contener también paréntesis que deben ser reconocidos. La parentización debe finalmente estar equilibrada.

Un paréntesis igual utiliza el mismo delimitador en el lado izquierdo y en el derecho. Por tanto, este paréntesis no pueden ser anidado ya que los delimitadores izquierdo y derecho no pueden distinguirse.

Por último, existe el paréntesis no anidable en los que "los interiores" deben, simplemente, no ser tenidos en cuenta. El analizador léxico buscará el delimitador derecho correspondiente.

La regla de interpretación para el mecanismo de escape es la siguiente: se considera que la cadena que sigue al carácter de escape constituye los interiores de un paréntesis y no un delimitador o un carácter de escape.

5.3 Interfaz semántica con lenguajes de datos externos

Para definir la semántica de un MSC con datos se han suministrado varias funciones relativas a las cadenas de datos, que van desde funciones para asegurar que las cadenas de datos están sintácticamente bien formadas, a funciones semánticas que pueden evaluar expresiones de datos. Es decir, la semántica de un MSC esta parametrizada por estas funciones de datos y la herramienta de análisis de MSC sólo tendría que estar provista de interfaces con estas funciones para calcular la semántica de MSC. Las funciones pueden dividirse en dos grupos: las funciones de interfaz para requisitos estáticos y las funciones de interfaz para semántica dinámica.

Funciones de interfaz para requisitos estáticos

Para comprobar que cada una de las cuatro clases <string> de datos es conforme con su gramática se requieren cuatro predicados de *buena formación* que deberán ser satisfechos solamente cuando las cadenas que constituyen sus argumentos sean analizadas gramaticalmente de forma correcta. Por tanto, se requieren cuatro predicados Wf1, Wf2, Wf3, y Wf4 que tienen las siguientes firmas:

Wf1: <variable string> → *Bool*

Wf2: <data definition string> → *Bool*

Wf3: <type ref string> → *Bool*

Wf4 <expression string> → *Bool*

Para asegurarse de que las cadenas de datos no sólo pasan correctamente el análisis gramatical, sino que también cumplen los requisitos estáticos del lenguaje, se deben definir tres funciones de *comprobación de tipo*. La primera de estas funciones, Tc1, para <data definition string> no requiere argumentos adicionales.

Tc1: <data definition string> → *Bool*

La segunda función, Tc2, tiene por objeto comprobar <type ref string>, pero como estas cadenas pueden contener información definida en la <data definition string>, como la definición de constantes utilizadas para suministrar límites de arrays, o referencias a definiciones de tipo, la función tiene que tomarlas como parámetros.

Tc2: <data definition string> → <type ref string> → *Bool*

La función de comprobación de tipo para <expression string>, Tc3, tiene que ser parametrizada por la <data definition string> junto con información definida por las declaraciones de variables y de comodines. Esta última información puede ser codificada como un conjunto de parejas, siendo el primer elemento una <variable string> y el segundo su correspondiente <type ref string>, independientemente de que la <variable string> se refiera a una declaración de variables o a una declaración de comodines. Por tanto, la firma para Tc3 es:

Tc3: <data definition string> × (<variable string> × <type ref string>)-*set* →

<expression string> → *Bool*

En varios lugares los requisitos estáticos de MSC exigen que los nombre de las variables sean únicos, por lo que en la interfaz se requiere también una función EqVar para comparar los nombres de variables:

EqVar: <variable string> × <variable string>) → *Bool*

Esta última función en la comprobación de la conformidad de tipos entre diferentes cadenas de datos, por ejemplo para comprobar que el tipo de una variable es el mismo que el de una expresión a la que se asigna. Basta con tener una función que pueda comprobar que una expresión es del tipo requerido; también en este caso se requiere información contextual de las cadenas de definición de datos, y las declaraciones de variables y de comodines.

Tc4: <data definition string> × (<variable string> × <type ref string>)-*set* →

(<type ref string> × <expression string>) → *Bool*

Funciones de interfaz de semántica dinámica

Se requieren cuatro funciones para definir la semántica dinámica de MSC con datos. Las primeras tres son funciones relativas a la sintaxis del lenguaje de datos, y sólo la cuarta, Eval, es una función de evaluación semántica. Las funciones sintácticas se requieren para tratar las expresiones de datos que contienen comodines pues éstas tienen semánticas diferentes de las expresiones sencillas que se encuentran en muchos lenguajes de datos. La semántica dinámica transforma expresiones que contienen comodines en expresiones que no contienen comodines; para ello emplea las primeras tres funciones.

La primera función, Vars, se utiliza para extraer las <variable string> de una <expression string>, y se requiere para realizar comprobaciones semánticas dinámicas, como por ejemplo verificar que las variables han sido escritas antes de ser leídas, y también como un medio para identificar comodines. Dado que la semántica trata múltiples comodines que aparecen en una expresión como magnitudes independientes, Vars tiene que determinar cuántas veces una <variable string> aparece en una expresión. Al igual que la función Tc4 es parametrizada por <data definition string> e información de declaración de variables y de comodines, de la misma manera lo es Vars. El resultado de aplicar Vars a una expresión de datos es un conjunto de parejas, siendo el primer elemento de cada una de ellas una <variable string>, y el segundo el número de apariciones de la variable en la expresión. Sólo las variables que aparecen en la expresión aparecerán en el resultado de las funciones. La firma de Vars viene dada por:

Vars: <data definition string> × (<variable string> × <type ref string>)-*set* →

<expression string> → (<variable string> × *Nat*)-*set*

Por ejemplo, la aplicación de Vars a la expresión "f(x, c, x + y)", donde x e y están declaradas como variables o comodines y c es una constante, produciría el conjunto:

{(x, 2), (y, 1)},

pues x aparece dos veces en la expresión e y aparece una vez.

Para que una expresión que contiene múltiples comodines idénticos sea evaluada correctamente, las apariciones individuales deben ser reemplazadas por nombres de variable únicos. Para ello se necesitan las dos funciones siguientes, Replace, que puede sustituir la aparición de una cadena de variable o una cadena de comodín en una expresión por otra cadena de variable, y NewVar, que puede generar una nueva cadena de variable. La función sustituta toma como argumentos una <data definition string>, la cadena de variable que va a ser sustituida, un número que define qué aparición va a ser sustituida, la cadena de variable sustituta, y finalmente la expresión que habrá de ser transformada. La <data definition string> tiene que suministrarse como un argumento porque – según el lenguaje de datos que se utilice – identificadores pueden ser sobrecargados como

constantes/variables, etc., y para distinguir entre un identificador de variable y otros identificadores se necesita información contextual. La firma es:

Replace: <data definition string> →

<variable string> × *Nat* × <variable string> → <expression string> →

<expression string>

El resultado de Replace(data_defs)(x, 2, z)(f(x, c, x + y)) podría ser la expresión f(x, c, z + y) o f(z, c, x + y), lo que dependerá de la forma de efectuar la indexación. Al suministrar una función Replace para un lenguaje de datos no se especifica cuántas apariciones de variables habrán de ser indexadas, ya que la función se utiliza en la semántica dinámica para reemplazar todos los comodines, sin que importe el orden en que se reemplazan.

Para la sustitución de comodines es necesario crear nuevas cadenas de variables, y esto se consigue suministrando una función NewVar a la interfaz. Dado un conjunto de <variable string>, la función deberá generar una nueva <variable string> que sea diferente de cada una de las comprendidas en el conjunto suministrado. En cuanto a Replace, los nombres de otros identificadores definidos deberán tomarse en consideración a partir de las cadenas de definición de datos. La firma de NewVar es:

NewVar: <data definition string> → <variable string>-*set* → <variable string>

La función Eval se utiliza para evaluar una expresión de datos y proporciona la interfaz semántica con MSC. La función Eval retorna el valor de una expresión de datos en algún dominio de datos. Por ejemplo, la evaluación de una expresión entera dará por resultado un valor entero. En general, las expresiones retornarán valores más complejos que representan tipos estructurados, tales como arrays, etc. Implícitamente, los dominios de datos forman parte de la interfaz semántica, y serán representados por U, un universo de valores de datos que incluye valores estructurados. Eval toma una <data definition string> como su primer argumento, para proporcionar información contextual sobre el segundo argumento, la cadena de expresión a evaluar. El último argumento representa la información de estado y consiste en vinculaciones entre cadenas de variables y sus valores de dominio. El resultado de la función es un valor de dominio.

Eval: <data definition string> → <expression string> → (<variable string> × U)-*set* → U

La definición de Eval es parcial y sólo está definida si todas las variables que aparecen en la expresión tienen valores definidos en el argumento estado.

5.4 Declaración de datos

Por lo general, la declaración de datos se hace en un documento MSC, siendo la única excepción las variables estáticas, que se declaran en un encabezamiento de MSC. Las declaraciones en documentos MSC incluyen:

- mensajes y temporizadores que tienen parámetros de datos;
- variables dinámicas;
- símbolos de comodín;
- definiciones de datos.

Además, los paréntesis de cadena de datos y el carácter de escape también se declaran en un documento MSC. Los mensajes que tienen parámetros se declaran de tal modo que el tipo y el número de parámetros estén definidos. No es necesario declarar los mensajes que no tienen parámetros. Las variables dinámicas se declaran en una lista de ejemplares de un documento MSC, ya que las variables dinámicas son poseídas por ejemplares. Una declaración indica los nombres de las variables y define su tipo. Las variables que deban representar comodines se declaran, junto con sus tipos, en el documento MSC, al igual que se declaran las definiciones de datos. Las definiciones de datos constan de texto en el lenguaje de datos que, por ejemplo, define tipos estructurados,

constantes y firmas de funciones. Deben proporcionar toda la información requerida para la comprobación de tipo y la evaluación de expresiones de datos utilizadas en MSCs dentro del alcance del documento MSC circundante.

Gramática textual concreta

<message decl list> ::=
<message decl> <end> [<message decl list>]

<message decl> ::=
<message name list> [: (<type ref list>)]

<message name list> ::=
<message name> [, <message name list>]

<timer decl list> ::=
<timer decl> <end> [<timer decl list>]

<timer decl> ::=
<timer name list> [<duration>] [: (<type ref list>)]

<timer name list> ::=
<timer name> [, <timer name list>]

<type ref list> ::=
<type ref string> [, <type ref list>]

<dynamic decl list> ::=
variables <variable decl list>

<variable decl list> ::=
<variable decl item> <end> [<variable decl list>]

<variable decl item> ::=
<variable list> : <type ref string>

<variable list> ::=
<variable string> [, <variable list>]

<data definition> ::=
[**language** <data language name> <end>]
[<wildcard decl>]
[**data** <data definition string> <end>]

<wildcard decl> ::=
wildcards <variable decl list>

Gramática gráfica concreta

Los conceptos de datos están contenidos en partes textuales por lo que no se necesita gramática gráfica.

Requisitos estáticos

Todos los mensajes que tienen parámetros deben declararse en el <document head>. El número y tipo de los parámetros de un mensaje que contiene datos deben también darse en su declaración. Un tipo se da en el lenguaje de datos de parámetros definido por <type ref string>. Todas las referencias utilizadas en una declaración de variables, comodines y mensajes tienen que estar legalmente definidas en el contexto de <data definition string> de acuerdo con las reglas del lenguaje de datos; esto significa que las referencias de tipo tienen que ser sintácticamente correctas de acuerdo con la función Wf3 y del tipo correcto de acuerdo con la función Tc2. Todas las declaraciones de variables y comodines tienen que ser legales de acuerdo con la función Wf1, y todos los nombres tienen que ser únicos cuando son evaluados por la función EqVar. La <data definition string> debe contener toda la información que se utilizará para efectuar las comprobaciones estática y dinámica requeridas por esta Recomendación. Aunque esta cadena puede que no forme fragmentos legales del lenguaje de datos deseado, dichos fragmentos tienen que representar una abstracción formalizada del lenguaje y tener una sintaxis y una semántica (estática) legales, esto es, deben ser sintácticamente correctos de

acuerdo con la función Wf2 y del tipo correcto de acuerdo con la función Tc1. Por ejemplo, sería razonable dar solamente encabezamiento de procedimiento que definen una firma de función y no su encabezamiento/cuerpo completo, pero tal abstracción tendría que ser formalizada y poderse comprobar mediante funciones de interfaz.

Semántica

Un comodín se utiliza en expresiones de datos como un valor "intrascendente". Al definir el significado de un valor MSC con datos, un comodín generará un conjunto de trazos concretos que corresponden a cada trazo no interpretado, donde cada trazo concreto se deriva del trazo no interpretado sustituyendo el comodín por un valor concreto diferente. Si una expresión contiene múltiples apariciones de un comodín, cada una representa una referencia diferente, por lo que cada aparición será sustituida, generalmente, por valores concretos diferentes.

5.5 Datos estáticos

Facultativamente, un MSC puede definir una lista de parámetros formales. Una correspondiente referencia de MSC debe definir una lista de parámetros reales. Una lista de parámetros MSC declara una lista de variables de parámetros formales cuyo alcance es el cuerpo de MSC. Cuando un parámetro aparece en el cuerpo sólo deberá utilizarse en expresiones que hacen referencia a su valor, pues no pueden modificarse dinámicamente.

Gramática textual concreta

<data parameter decl> ::=
<variable decl list>

<actual data parameters> ::=
<actual data parameter list>

<actual data parameter list> ::=
<expression string> [, <actual data parameter list>]

Requisitos estáticos

La <data parameter decl> declara los parámetros formales de un MSC. Cada parámetro formal de la declaración tiene que ser único dentro de la declaración y también diferente de todas las variables dinámicas y comodines declarados en el documento MSC que los posee. Cada variable tiene que estar bien formada de acuerdo con Wf1, y su referencia de tipo debe estar bien formada y su tipo ser correcto en el contexto de la cadena de datos del documento MSC circundante de acuerdo con Wf3 y Tc2, respectivamente. El número y tipo de parámetros tienen que estar aceptados en una referencia de MSC mediante su <actual data parameter list>; la conformidad de tipo se asegura por la función Tc4, dado el contexto de la cadena de definición de datos MSC y la declaración de parámetros formales. Cada parámetro real es una expresión que puede contener variables y comodines. Todos los parámetros que aparecen en la <actual data parameter list> tienen que ser variables estáticas declaradas en el MSC circundante; las variables dinámicas están prohibidas.

Semántica

El significado de una referencia de MSC con parámetros reales es "call by value", donde los parámetros formales son sustituidos por los parámetros reales cada vez que aparecen en su cuerpo. Sin embargo si hay múltiples comodines dados en la <actual data parameter list>, cada uno de ellos representa una referencia separada y única que será distinguida semánticamente en el cuerpo del MSC referenciado sustituyéndolos por nombres de variable nuevos distintos, a cada uno de los cuales se le permite tomar cualquier valor en su dominio. La evaluación de los parámetros reales la efectúa la función Eval una vez que los comodines hayan sido identificados y reemplazados por nuevas variables únicas. Para efectuar el reemplazo se extraen primero todas las variables y comodines de la expresión mediante la función Vars, después de lo cual se utilizan las declaraciones de comodines del documento MSC para identificar cuáles son los comodines de la expresión. Por

último, se reemplazan los comodines utilizando la función Subst con nuevas variables generadas por la función NewVar. Estas nuevas variables pueden tomar cualquier valor en sus dominios de datos.

5.6 Datos dinámicos

Los datos dinámicos hacen referencia a variables MSC a las que se pueden asignar y reasignar valores mediante casillas de acción, parámetros de mensajes y de temporizador y creación de ejemplares. El valor que una variable dinámica puede poseer en cualquier punto en un trazo dependerá, en general, de los eventos precedentes en el trazo. El inicio de una variable dinámica es un ejemplar, y su declaración se expresa en el documento MSC. Sólo los eventos en el ejemplar propietario de una variable dinámica están autorizados a modificar su valor, aunque otros ejemplares pueden hacer referencia a este valor bajo ciertas restricciones.

El mecanismo para asignar o modificar el valor de una variable dinámica es el de una vinculación, que consta de un patrón y una expresión. Las vinculaciones se producen como resultado de pasos de mensajes fijación de temporizadores, expiración de temporizaciones y creación de ejemplares mediante sus listas de parámetros, o en casillas de acción. La semántica dinámica define un conjunto de vinculaciones actuales para cada evento en un trazo de ejecución; se denomina estado del evento. La utilización de comodines en expresiones tiene por consecuencia una subespecificación en el MSC. Se permite que cada comodín tome cualquiera de los valores de su tipo de dominio.

En un MSC definidor no puede haber ningún trazo a través de un MSC en el que se haga referencia a una variable que no haya sido definida. Esto es, cada variable que aparece en una expresión tiene que estar vinculada en el estado utilizado para calcular el valor de la expresión. En un MSC utilidad se permiten referencias a variables no definidas.

Un estado asociado con un evento actual se calcula a partir de estados anteriores junto con el contenido de datos de ese evento. Los estados anteriores utilizados para calcular el nuevo estado dependen del tipo de evento, y todos ellos se derivan de, por lo menos, el último evento no creador ejecutado en el mismo ejemplar que el evento actual. Además, en el caso de eventos que reciben mensajes, y en el caso del primer evento en un ejemplar creado, el estado de los correspondientes eventos de envío o de creación se utiliza también en el cálculo. Efectivamente, esto significa que un estado es mantenido por cada ejemplar, y un nuevo estado se deriva del estado precedente del ejemplar junto con la información de estado pasada al ejemplar por medio de mensajes, o desde el ejemplar progenitor en el caso de creación de ejemplar. Dado que se permite el flujo de información entre ejemplares mediante el paso de mensajes y la creación de ejemplares, el estado asociado con cada evento puede contener vinculaciones a variables no poseídas por el ejemplar sobre la que aparece el evento. Las reglas que rigen el acceso al valor de variables poseídas por ejemplares extranjeros se definen como sigue.

Si un evento de envío o un evento de creación tienen una vinculación a una variable x en su estado asociado, y x aparece en una de sus expresiones de parámetros, y x no está poseída por el ejemplar receptor ni por el ejemplar creado, la vinculación deberá añadirse al estado resultante del evento de recepción, o deberá reemplazar a una vinculación si existe una vinculación en el estado antiguo. Es decir, si el valor de x está registrado en el estado del ejemplar de envío/creación, esta vinculación es heredada implícitamente por el estado del ejemplar receptor/creado en tanto que x no esté poseída por el ejemplar receptor/creado.

Si x es poseída por el ejemplar receptor, la vinculación definida en el evento de envío sólo puede ser heredada si x no está vinculada en el estado del ejemplar receptor y x no está vinculada en la lista de parámetros. Esto es, la vinculación sólo puede ser heredada si x no está definida por el ejemplar receptor y tampoco lo está por parámetros de mensajes.

Si x es poseída por el ejemplar receptor/creado y x está vinculada en los parámetros de mensajes, el nuevo estado toma esta vinculación. Esto es, x es vinculada al valor definido por la parte expresión de la vinculación de parámetro. Este el modo normal de vinculación de parámetro.

Si x es poseída por el ejemplar receptor y x está vinculada en el estado antiguo de este ejemplar, pero no está vinculada en la lista de parámetros, esta vinculación antigua se retiene en el nuevo estado. Es decir, la vinculación del evento de envío no se hereda. Esto refleja la expectativa de que una vez que una variable ha sido definida en su ejemplar, su valor sólo puede ser modificado explícitamente. Toda referencia a este valor tomará el último valor definido explícitamente.

En resumen, si x no está vinculada en el estado antiguo ni en la lista de parámetros, la vinculación del evento de envío puede ser heredada. Intuitivamente, la vinculación de una variable puede ser heredada de otro ejemplar únicamente si la variable se envía al ejemplar haciéndolo aparecer en la expresión del parámetro. Sin embargo, una vinculación no puede heredarse si la variable es poseída, y está dentro de su alcance, por el ejemplar receptor, pues la vinculación local tiene precedencia. Por tanto, el valor de una variable puede transmitirse a otros ejemplares por una cadena de mensajes en tanto que cada mensaje haga referencia explícita a la variable en su lista de parámetros.

5.7 Vinculaciones

Las vinculaciones (*bindings*) son más generales que las simples asignaciones existentes en los lenguajes de programación, debido a la utilización de símbolos de comodines, que permiten una subespecificación. Una vinculación consta de una parte expresión (*expression part*) y de una parte patrón (*pattern part*) que están conectadas por un símbolo de vincular. El símbolo de vincular tiene una forma izquierda y una forma derecha, siendo ambas equivalentes, pero que permiten una lectura más natural de una vinculación asociada con un mensaje o un temporizador. Los símbolos de comodín pueden utilizarse en una vinculación en todo lugar en que podría utilizarse una variable, pero tienen una interpretación diferente de la de las variables pues pueden tomar todos los valores permitidos, a diferencia de las variables a las que se puede asignar un solo valor.

La parte patrón de una vinculación consta de un comodín, o de una variable dinámica; la parte expresión es una expresión de datos, que puede contener comodines, variables estáticas y variables dinámicas. En MSC, los símbolos de comodín son definidos por el usuario, pero en los ejemplos que siguen se utilizará "_" para representar el comodín. En el ejemplo siguiente se presenta una vinculación izquierda y una vinculación derecha que son equivalentes y no tienen comodines:

x := y + 3, y + 3 =: x

Estas vinculaciones pueden leerse como una asignación, o vinculación, del valor de la expresión 'y + 3' a la variable x. Si se utiliza un comodín en lugar de x, como en el ejemplo siguiente, no se hace ninguna asignación por lo que la gramática para parámetros de mensajes permite una escritura abreviada equivalente a la simple utilización de la expresión, que también se muestra. Esta escritura abreviada se utiliza cuando se especifica el valor de envío de un parámetro pero el ejemplar receptor no vincula este valor a ninguna de sus variables dinámicas.

_ := y + 3, y + 3

Se utilizan comodines en expresiones para representar valores "intrascendentes" ("don't care values"). Si la semántica dinámica prescribe un trazo individual no interpretado a través de un MSC que contiene un comodín de expresión, el comodín puede ser sustituido por cualquier valor para obtener un trazo concreto legal. Además, si existen múltiples apariciones de comodines (incluso cuando utilicen el mismo símbolo de comodín), cada uno de ellos es independiente y puede ser sustituido por valores diferentes. Supóngase en la expresión _+ 3 que el comodín '_' tiene que ser de tipo Natural (es decir, entero no negativo), entonces, semánticamente, puede ser sustituido por cualquiera de los valores 0, 1, 2, ...Por tanto, los valores que puede tomar la expresión son 3, 4, 5, ... En la expresión _ + _, las dos apariciones de los símbolos de comodín son independientes, por lo que tanto la primera como la segunda aparición están en libertad de tomar cualquier valor de la gama de los números naturales. Por ejemplo, el primer comodín puede tomar el valor 1 y el segundo el valor 3 con lo que se obtiene el valor 4. Si se requieren los mismos valores, se puede vincular una variable a un comodín en una casilla de acción y, después, utilizar esta

variable en la expresión, en vez de los comodines. Dada la vinculación inicial de x a un comodín, la expresión x + x sólo puede ser evaluada a números naturales pares.

Gramática textual concreta

<binding> ::=
<left binding> | <right binding>

<left binding> ::=
<pattern> <left bind symbol> <expression>

<left bind symbol> ::=
:=

<right binding> ::=
<expression> <right bind symbol> <pattern>

<right bind symbol> ::=
=:

<expression> ::=
<expression string>

<pattern> ::=
<variable string> | <wildcard>

<wildcard> ::=
<wildcard string>

Requisitos estáticos

La parte expresión de una vinculación debe cumplir las reglas que establecen los requisitos estáticos del lenguaje de datos, dado el contexto de la <data definition string> definido en la <document head> de acuerdo con las funciones Wf3 y Tc3. Todas las variables que aparecen en la expresión deben ser declaradas como variables estáticas, variables dinámicas, o comodines; todos los demás identificadores tienen que ser definidos mediante la <data definition string> o estar predefinidos en el lenguaje de datos. Todas las variables de patrones deben ser declaradas como variables dinámicas poseídas por el ejemplar correcto – siendo el ejemplar correcto determinado por el evento en que aparece la vinculación.

En una vinculación, el tipo del patrón debe concordar con el tipo de la expresión. La equivalencia de tipos puede comprobarse aplicando la función Tc4 al patrón y a la expresión.

Semántica

Una vinculación, o una lista de vinculaciones, se evalúa en el contexto de un estado y provoca la aparición de un nuevo estado. Un estado consiste en un conjunto de vinculaciones entre variables y sus valores. Para evaluar una vinculación, primeramente se evalúa su expresión utilizando el estado actual, donde los valores de variables en la expresión se toman de su vinculación en el estado. Si en el estado actual no hay ninguna vinculación de una variable de la expresión, se está en presencia de una referencia ilegal a una variable no definida en la expresión. El valor resultante calculado a partir de la expresión se vincula a la variable del patrón y la nueva vinculación resultante se añade al estado actual para formar el nuevo estado. Si la variable de patrón ya está vinculada en el estado actual, dicha variable será reemplazada en el nuevo estado por la nueva vinculación.

Si la parte patrón de una vinculación es un comodín, la vinculación no modifica el estado. Si la parte expresión de una vinculación contiene comodines, todos y cada uno de ellos se reemplazan por nuevas variables únicas, cada una de las cuales puede tomar cualquier valor en su respectivo dominio. La evaluación de una expresión la efectúa la función Eval una vez que los comodines han sido identificados y reemplazados por nuevas variables únicas. El reemplazo se efectúa extrayendo primero todas las variables y comodines de la expresión mediante la función Vars, después de lo cual se especifican los comodines de la expresión utilizando para ello las declaraciones de comodín en el documento MSC. Por último, se reemplazan los comodines utilizando la función Replace con las

nuevas variables generadas por la función NewVar. Estas variables pueden tomar cualquier valor en la semántica de trazo de un MSC.

5.8 Datos en parámetros de mensajes y temporizadores

Los datos dinámicos entran a formar parte de mensajes y temporizadores a través de sus listas de parámetros. Una lista de parámetros puede estar constituida por vinculaciones, expresiones, o patrones. Los requisitos estáticos determinan cuál de estas tres opciones puede o debe utilizarse. En el caso de mensajes completados en un documento MSC puede utilizarse una vinculación o una expresión, pero no un patrón. En este contexto, una expresión es una escritura abreviada para vincular a un comodín. Para los mensajes incompletos se aplican reglas más complejas.

Se produce un mensaje incompleto cuando un mensaje se origina o termina en una puerta, o es un mensaje perdido, o encontrado. Cuando un mensaje termina o se origina en una puerta, las vinculaciones no pueden estar explícitamente definidas en su lista de parámetros porque las vinculaciones sólo tienen sentido cuando tanto el ejemplar de origen como el de destino son conocidos. Las vinculaciones serán inferidas dinámicamente en la semántica emparejando patrones y expresiones dados por los dos mensajes cuya concordancia se determina a través de la puerta. En el caso de un mensaje que se envía desde un ejemplar y termina en una puerta, sólo puede darse la expresión. En el caso de un mensaje que es tomado de una puerta y enviado a un ejemplar, sólo puede darse el patrón. Restricciones similares se imponen a los mensajes perdidos y a los encontrados. En el ejemplo de la figura 9, la semántica dinámica determinará la concordancia de la *request* (petición) de mensaje a través de la puerta *g* e inferirá la vinculación *y + 3 =: x*. Si un mensaje se origina y termina en una puerta, no se permite información de parámetros.

El conjunto de vinculaciones definido o formado por parámetros de un mensaje provoca un cambio de estado en el evento receptor. Las vinculaciones se evalúan concurrentemente y se utilizan para actualizar el estado antiguo. Como las vinculaciones son concurrentes, el conjunto de variables que aparecen como variables de patrón en las vinculaciones tienen que ser distintas, para impedir que dos vinculaciones traten de vincular dos valores a una misma variable. Además, todas la variables de patrón tienen que ser poseídas por el ejemplar receptor, pues a este ejemplar sólo se le permite modificar sus propias variables, y no otras.

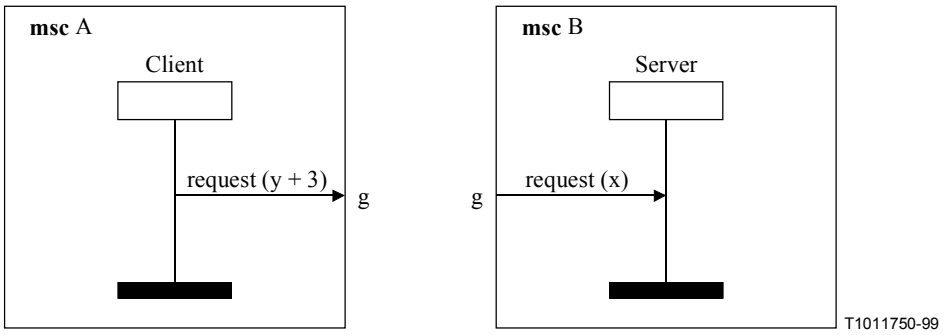


Figura 9/Z.120 – Mensajes, expresiones, patrones y puertas

Gramática textual concreta

<parameter list> ::=
<parameter defn> [, <parameter list>]

<parameter defn> ::=
<binding> | <expression> | <pattern>

Requisitos estáticos

El número y tipo de parámetros de mensaje y temporizador deben ajustarse a los tipos definidos en la declaración del mensaje y el temporizador. El tipo de una vinculación viene determinado por su patrón o por su expresión, ya que éstos deben concordar también. La conformidad de los tipos se comprueba mediante la función Tc4. En una lista de parámetros, el conjunto de variables de patrón debe ser único y todas las variables deben ser poseídas por el ejemplar receptor del mensaje. Las vinculaciones de una lista de parámetros de mensaje se evalúan concurrentemente, con lo que se evita la ambigüedad.

Sólo una vinculación o una expresión se pueden dar como parámetros de mensaje en mensajes completados; los patrones no se pueden dar como parámetros. Sólo una expresión puede darse para eventos de envío incompletos; no pueden darse vinculaciones ni patrones. Sólo patrones pueden darse para eventos de recepción incompletos; no pueden darse ni vinculaciones ni expresiones.

Semántica

Los parámetros de mensaje tienen por cometido actualizar el estado en eventos de recepción de mensaje. Los eventos de salida no cambian de estado, y su estado se hereda simplemente del último evento del ejemplar que no es un evento de creación.

En el caso de mensajes completados, todas las vinculaciones en la lista de parámetros de mensaje del evento de recepción se valúan utilizando el estado antiguo, y las vinculaciones resultantes se utilizan para actualizar el estado antiguo y formar el nuevo estado. Esto es, una vinculación se añade al estado antiguo o, si a una variable ya vinculada en el estado antiguo es vinculada de nuevo en la lista de parámetros, la vinculación más reciente reemplaza a la antigua en el nuevo estado. Las variables que aparecen en las partes expresión de una lista de parámetros contribuyen también a la actualización del estado antiguo. Las vinculaciones de estas variables referenciadas se toman del estado del evento de envío y también se utilizan para actualizar el estado antiguo, salvo las variables que estén poseídas por el ejemplar receptor. En este último caso, la vinculación sólo se añade si la variable no está vinculada en el estado antiguo y no está vinculada por la lista de parámetros.

En el caso de eventos receptores de mensaje incompletos, primeramente se crean dinámicamente vinculaciones emparejando las expresiones de parámetro concordantes procedentes del correspondiente evento de envío con los parámetros del evento de recepción. Las vinculaciones resultantes se evalúan entonces como en el caso de mensajes completados, y el estado se modifica del mismo modo.

5.9 Datos en parámetros de creación de ejemplar

Los datos dinámicos en eventos de creación de ejemplar se tratan de una manera similar a aquella en que se tratan los parámetros de mensajes. Sin embargo, como no hay un evento creado correspondiente al evento creador, el estado es modificado por el evento de creación de acuerdo con cualesquiera vinculaciones en su lista de parámetros. Sin embargo, este estado modificado sólo se emplea para evaluar el evento siguiente en el ejemplar creado, y no el evento siguiente en el ejemplar creador. El estado que existía antes del evento de creación se utiliza para evaluar el evento subsiguiente en el ejemplar creador. Todas las variables utilizadas como variables de patrón en una lista de parámetros de creación tienen que ser poseídas por el ejemplar creado.

5.10 Datos en casillas de acción

Los datos en casillas de acción pueden aparecer como una lista de enunciados (*statements*) separados por el carácter coma (","). Un enunciado es o bien un enunciado de definir (*define statement*), o un enunciado de indefinir (*undefine statement*), o una vinculación. Un enunciado de definir se utiliza para indicar que se ha asignado a una variable algún valor no especificado; es equivalente a la vinculación de una variable a un comodín. Esto es, "**def** x" es equivalente a "x := _", donde "_" es un comodín. Un enunciado de indefinir se utiliza para indicar que una variable ya no está vinculada, es

decir, que la variable no puede ser legalmente referenciada, o que ha tomado un valor que está fuera del alcance. Dentro de una casilla de acción individual, los enunciados se evalúan concurrentemente, no secuencialmente, por lo que se aplican las reglas de los requisitos estáticos que prohíben la existencia de toda ambigüedad entre diferentes enunciados que traten de vincular valores diferentes a la misma variable. La secuenciación puede obtenerse por la secuenciación de casillas de acción.

Gramática textual concreta

<data statement list> ::=
<data statement> [, <data statement list>]

<data statement> ::=
<define statement> | <undefine statement> | <binding>

<define statement> ::=
def <variable identifier>

<undefine statement> ::=
undef <variable identifier>

Requisitos estáticos

Todas las variables que aparecen en un enunciado de definir, un enunciado de indefinir, o en la parte patrón de una vinculación deben ser variables dinámicas poseídas por el ejemplar en que aparece su casilla de acción circundante y, además de esto, deben ser distintas. Esto último es necesario porque los enunciados de una casilla de acción se evalúan concurrentemente.

Semántica

Cada enunciado en una casilla de acción se evalúa utilizando del evento anterior que no es un evento de creación, en el mismo ejemplar. El estado resultante se deriva del estado antiguo actualizando las vinculaciones hechas o deshechas por cada uno de los enunciados. En virtud de los requisitos estáticos no puede haber ambigüedad en la formación del estado resultante. En el caso de vinculaciones, las variables de patrón se vinculan, en el nuevo estado, a los valores de sus expresiones. Un enunciado de indefinir suprime la vinculación de la variable, del estado antiguo, si existe, al formar el nuevo estado. Un enunciado de definir añade una vinculación al estado antiguo, o reemplaza una vinculación del estado antiguo por una vinculación en la que la variable del enunciado está vinculada a un comodín.

5.11 Tipos de datos supuestos

En tres lugares de esta Recomendación el lenguaje MSC supone la existencia de tipos de datos. Estos tipos de datos son:

- expresiones de valor booleano utilizadas en condiciones de guarda (véase 4.7);
- expresiones de número natural utilizadas para definir fronteras de bucles (véase 7.2);
- expresiones de tiempo utilizadas en la especificación de constricciones de tiempo (véase la cláusula 6).

En consonancia con el método seguido en esta Recomendación para el tratamiento de los datos, estos tipos tienen que ser definidos como parte del lenguaje de datos elegido por el usuario. Sin embargo, se requiere que tengan interpretaciones específicas, ya que las semánticas del MSC se definen en términos de estas interpretaciones. Por tanto, el tipo de expresiones utilizadas en condiciones de guarda debe tener el dominio estándar de las variables booleanas, constituido por dos elementos solamente: un elemento *true* y un elemento *false*.

En cuanto a las fronteras de bucle, el dominio del tipo tiene que ser el conjunto de los números naturales, dotado de la usual operación de sustracción (requerida para calcular la diferencia entre el límite inferior y el límite superior de un bucle). Esta condición puede mitigarse permitiéndose que el dominio sea un subconjunto de los números naturales, pues la mayor parte de los lenguajes de

programación sólo tienen representaciones finitas. Esto no afecta a la semántica de los bucles, ya que, en tal situación, los valores límite tendrán que pertenecer al subconjunto permitido. En particular, los bucles infinitos pueden aun expresarse mediante la palabra clave **inf**.

Los requisitos del tipo time son más abstractos y se definen como sigue:

- el dominio debe ser un dominio de ordenación total con un elemento mínimo, u origen, de tiempo cero;
- el dominio debe cerrase tras una operación de adición, utilizada para calcular desplazamientos de tiempo.

Estos son los requisitos mínimos que permiten definir la semántica de MSC temporizados. En la práctica, un usuario tendrá un tipo más rico que soporte operaciones de tiempo adicionales declaradas en la parte definición de datos del documento MSC. Al igual que en el caso de los números naturales, se puede permitir que el dominio real sea un subconjunto del dominio ideal requerido, para tener en cuenta las limitaciones de los lenguajes de datos reales. El requisito relativo a la ordenación total refleja el modelo de tiempo lineal utilizado en MSC, por oposición a lo que podría denominarse un modelo de bifurcación de tiempo. El cierre tras una operación de adición significa que el tiempo nunca puede agotarse.

6 Conceptos de tiempo

Se introducen conceptos de tiempo en MSC con el fin de soportar la noción de tiempo cuantificado para la descripción de los sistemas en tiempo real con un significado preciso de la secuencia de eventos en el tiempo. Los eventos MSC son instantáneos. Se pueden especificar constricciones de tiempo para definir el tiempo en que pueden aparecer eventos.

Cada MSC contiene ejemplares con eventos asociados. Los MSC clásicos que no tienen en cuenta el tiempo pueden interpretarse como un conjunto de trazos de eventos. En la interpretación no temporizada se considera que todos las características relacionadas con el tiempo son comentarios/anotaciones solamente. En la interpretación temporizada, el transcurso de tiempo se representa explícitamente de una manera cuantificada, esto es, los trazos de eventos son realzados con un evento especial que representa el transcurso de tiempo. La interpretación no temporizada de un MSC es un superconjunto de la interpretación temporizada de un MSC, reducido a las características no temporizadas.

La temporización en MSC realza los trazos de un MSC con valores de tiempo cuantitativos, que representan la distancia de tiempo entre parejas de eventos. El avance del tiempo (esto es, la indicación de reloj) es igual para todos los ejemplares en un MSC. Asimismo, todos los valores de reloj son iguales, lo que equivale a suponer un reloj global. Todos los eventos son instantáneos, es decir, atómicos, y no consumen tiempo.

6.1 Semántica temporizada

La semántica temporizada de un MSC puede representarse por trazos con eventos de tiempo especiales tales como:

{e1, e2, t3, e4, t5, e6, e7, e8, … }

La tripleta (e4, t5, e6) significa por ejemplo que después de la aparición del evento e4 transcurre el tiempo t4 hasta que aparece el evento e6. Los eventos entre los cuales no hay evento de tiempo (por ejemplo e1, e2) aparecen simultáneamente, es decir, sin retardo. Se supone que el tiempo avanza y no se queda estancado. Por avance del tiempo ha de entenderse que después de cada evento en un trazo aparece, finalmente, un evento de tiempo. Por no estancamiento se entiende que existe un límite superior en cuanto al número de 'eventos normales' que aparecen entre un evento temporizado y el evento temporizado que le sigue.

El trazo citado más arriba es igual trazo que sigue (en el que se ha indicado explícitamente el retardo cero):

{e1, 0, e2, t3, e4, t5, e6, 0, e7, 0, e8, … }

La semántica no temporizada de un MSC que contiene los trazos:

{e1, e2, e3, … }

corresponde a un conjunto de trazos con un retardo arbitrario entre los eventos, esto es:

{e1, cualquier tiempo, e2, cualquier tiempo, e3, cualquier tiempo, ...}

Por otro lado, la presentación reducida no temporizado de un trazo temporizado:

{e1, t1, e2, t2, e3, t3, e4, t4, e5, t5, … }

es:

{e1, e2, e3, … }

Se ha hecho a propósito que en la semántica temporizada un trazo de un MSC comience con un evento 'normal', es decir, para cada evento, excepto el primero hay un evento precedente. La temporización puede definirse con respecto al evento precedente.

En general, no hay un evento de comienzo único para un ejemplar de un MSC, ni tampoco para un MSC en su totalidad. Un MSC define un conjunto de trazos con eventos de comienzo potencialmente diferentes.

6.2 Temporización relativa

La temporización relativa utiliza parejas de eventos – el evento precedente y el evento subsiguiente – donde el evento precedente habilita (directa o indirectamente, esto es, a través de algunos eventos intermedios) el evento subsiguiente. Para la utilización de la temporización relativa, véase 6.10 sobre intervalos de tiempo.

La temporización relativa puede especificarse mediante el empleo de expresiones arbitrarias de tipo Time, es decir, haciendo referencia a parámetros, comodines y variables dinámicas. El valor concreto de una expresión de tiempo relativa se evalúa una vez que el nuevo estado del evento relacionado con esta temporización relativa ha sido evaluado. Para la noción de un nuevo estado, véase 5.6.

6.3 Temporización absoluta

La temporización absoluta se utiliza para definir la aparición de eventos en puntos de tiempo relacionados con el valor del reloj global.

La temporización absoluta puede especificarse mediante el empleo de expresiones arbitrarias de tipo Time, es decir, haciendo referencia a parámetros, comodines y variables dinámicas. Los valores concretos de una constricción de tiempo se evalúan al comienzo de un intervalo de tiempo una vez que el nuevo estado del evento relacionado con el comienzo del intervalo de tiempo ha sido evaluado. Para la noción de nuevo estado, véase 5.6.

6.4 Dominio de tiempo

El dominio de tiempo puede ser denso o discreto. Debe estar caracterizado por una ordenación total con un elemento mínimo u origen de tiempo cero. Se ha de cerrar en aplicación de una operación adicional, utilizada para calcular desplazamientos de tiempo.

Son ejemplos de dominio de tiempo los números racionales no negativos realzados con una unidad de tiempo que es o bien h (indicativa de hora), min (indicativa de minutos), s (indicativa de

segundos), ms (indicativa de milisegundos), μs (indicativa de microsegundos) y ns (indicativa de nanosegundos).

Requisitos estáticos

El dominio de tiempo debe estar caracterizado por una ordenación total con un elemento mínimo u origen de tiempo cero. Se ha de cerrar en aplicación de una operación adicional, utilizada para calcular desplazamientos de tiempo.

6.5 Variables de tiempo estáticas y dinámicas

Un MSC puede utilizar variables de tiempo estáticas o dinámicas. Estas variables son como cualquiera otra variable, con la salvedad de que son del tipo Time del MSC.

Requisitos estáticos

Las variables de tiempo estáticas y dinámicas cumplen los requisitos de otras variables estáticas y dinámicas como se describe en la cláusula 5 sobre conceptos de datos.

6.6 Desplazamiento de tiempo

Se puede asignar a un MSC un desplazamiento de tiempo, que se utiliza como un desplazamiento de todos los valores de tiempo absolutos dentro de ese MSC. El desplazamiento de tiempo se define por una expresión del dominio de tiempo del MSC. En un MSC sin un desplazamiento de tiempo explícito, se supone por defecto un desplazamiento de tiempo cero.

Gramática textual concreta

<time offset> ::=
offset <time expression>

Gramática gráfica concreta

No se necesita.

Requisitos estáticos

En la declaración de desplazamiento hay que dar una sola expresión de tipo Time. La expresión debe referirse solamente a parámetros MSC declarados.

6.7 Puntos de tiempo, mediciones, e intervalos

Las constricciones de tiempo pueden definirse como puntos de tiempo, es decir, valores de tiempo concretos, o como intervalos de tiempo, es decir, gamas de valores de tiempo entre límites dados. Las observaciones de tiempo se describen por mediciones.

6.8 Puntos de tiempo

Gramática textual concreta

<time point> ::=
[<abs time mark>] <time expression>

Los puntos de tiempo se definen por expresiones de tipo Time. La marca de tiempo absoluto facultativa indica una temporización absoluta.

Gramática gráfica concreta

No se necesita

Requisitos estáticos

La evaluación de un punto de tiempo da un tiempo cuantificado concreto. Un evento sin constricciones de tiempo puede aparecer en cualquier momento.

6.9 Mediciones

Se utilizan mediciones para observar el retardo entre la habilitación de un evento y la aparición de dicho evento (para temporización relativa) y para medir el tiempo absoluto de la aparición de un evento (para temporización absoluta). Con el fin de distinguir una medición relativa de una medición absoluta se utilizan dos marcas diferentes ('&' para medición absoluta y '?' para medición relativa).

Se pueden unir mediciones a intervalos de tiempo. Para cada medición hay que declarar una variable de tiempo para el ejemplar respectivo.

Gramática textual concreta

```
<measurement> ::=
                                <rel measurement>
                              | <abs measurement>

<rel measurement>::=
                                <rel time mark> <time pattern>

<abs measurement>::=
                                <abs time mark> <time pattern>
```

Gramática gráfica concreta

No se necesita.

Requisitos estáticos

Las mediciones deben utilizar solamente patrones de tipo Time, los cuales hacen referencia a variables dinámicas que son declaradas en el documento MSC circundante para el ejemplar respectivo.

Semántica

La evaluación del retardo (en el caso de medición relativa) y del valor del reloj global (en el caso de medición absoluta) y la vinculación a la variable de tiempo forman parte de la evaluación del nuevo estado del evento con el que se relaciona la medición.

6.10 Intervalo de tiempo

Se utilizan intervalos de tiempo para definir constricciones sobre la temporización para la aparición de eventos: el retardo entre una pareja de eventos puede ser constreñida definiendo un límite mínimo o un límite máximo para el retardo entre los dos eventos.

Un intervalo de tiempo no implica que los eventos deban producirse. El cumplimiento de una constricción de tiempo es validado solamente si el evento relacionado con el final de ese intervalo de tiempo aparece en el trazo. Un trazo MSC tiene que cumplir todas sus constricciones de tiempo, esto es, si un trazo infringe una constricción de tiempo es ilegal.

Se pueden utilizar intervalos de tiempo para temporización relativa y para temporización absoluta. Se pueden especificar intervalos de tiempo mediante el empleo de expresiones arbitrarias de tipo Time, es decir, haciendo referencia a parámetros, comodines y variables dinámicas. Los valores concretos de una constricción de tiempo impuesta por un intervalo de tiempo se evalúan al comienzo de un intervalo de tiempo una vez que ha sido evaluado el nuevo estado del evento relacionado con el comienzo del intervalo de tiempo. Para la noción de nuevo estado, véase 5.6.

El orden de una pareja de eventos relacionados con una constricción de tiempo lo determina la semántica dinámica y no la constricción de tiempo.

Mediante fronteras de intervalo, cada evento puede imponer constricciones a intervalos de tiempo en los que interviene. Sin embargo, éstas sólo se tienen en cuenta si el evento hace referencia al comienzo de un intervalo de tiempo que es determinado dinámicamente. Cuando un evento hace referencia al final de un intervalo de tiempo, sus constricciones de tiempo no ofrecen interés. Las representaciones textual y gráfica de un MSC sindican solamente las parejas potenciales de eventos. Una pareja potencial de eventos se indica sea conectando las fronteras de intervalo de dos eventos, sea conectando fronteras de intervalos divididos mediante una etiqueta de intervalo, sea mediante puertas de mensajes.

Gramática textual concreta

```
<time interval> ::=
                    [<interval label>] <singular time>
                |   [<interval label>] <bounded time>
                    [ <measurement> ]

<interval label> ::=
                    int_boundary < interval name>


<singular time> ::=
                    <left closed> <time point> <right closed>
                |   <measurement>

<bounded time> ::=
                    [<abs time mark>] { <left open> | <left closed> }
                    [<time point>] , [<time point>]
                    { <right open> | <right closed> }
```

Requisitos estáticos

Dentro de un intervalo de tiempo hay que utilizar o bien solamente expresiones de tiempo relativas, o solamente expresiones de tiempo absolutas. Se da el límite mínimo, o el límite máximo, o ambos. Un intervalo debe definir al menos uno de los dos límites.

Un intervalo de tiempo absoluto debe ser de la forma [@1,@3) o @[1,3).

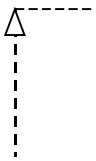
Pueden definirse intervalos de tiempo para cualesquiera dos eventos dentro de un documento MSC.

Gramática gráfica concreta

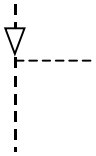
```
<time interval area> ::=
                    <interval area>
                |   <abs time area>

<interval area> ::=
                    <int symbol>
                    is associated with <time interval>
                    is followed by { <cont interval> | <interval area 2>}

<interval area 2> ::=
                    <int symbol>
                    [is associated with <time interval>]

<cont interval> ::=
                    <cont int symbol>
                    is associated with <interval name>

<int symbol> ::=
                    {<int symbol 1> | <int symbol 2> }
                    is attached to
                    { <message out symbol> |<message in symbol> |
                      <reply symbol> | <action symbol> |
                      <timer start symbol> | <timer stop symbol> |<timeout symbol> |
                      <restart symbol> | <createline symbol> |
                      <inline expression symbol> | <separator symbol>
```

<msc reference symbol> | <par frame symbol>
<instance head symbol> }

<int symbol 1> ::=



<int symbol 2> ::=



Los símbolos de intervalo de tiempo <int symbol 1> e <int symbol 2> pueden ser replicados horizontal y verticalmente. La réplica de <int symbol 1> no debe estar ligada a <int_symbol 1>.

<cont int symbol> ::=

El símbolo de continuación para intervalos de tiempo <cont int symbol> puede ser replicado verticalmente. La réplica de <cont int symbol> no debe estar ligada a <int symbol 1> e <int symbol 2> así como <cont int symbol> no debe estar ligado a la réplica de <int symbol 1> o de <int symbol 2>.

<abs time area>::=
<abs time symbol>
is associated with { <abs time expr> | <abs measurement>}
is attached to
{ <message out symbol> |<message in symbol> | <action symbol> |
<timer start symbol> | <timer stop symbol> |<timeout symbol> |
<inline expression symbol> | <separator symbol> |
<msc reference symbol> | <par frame symbol> |
<call in symbol> | <call out symbol> | <reply symbol>}

<abs time symbol>::=

<abs time expr>::=
<abs time mark> <time expression>

En la representación gráfica de un intervalo de temporización relativa se emplean líneas de trazo discontinuo con dos cabezas de flecha (como cabezas de flecha vacías) para indicar el evento de comienzo y el evento de finalización de este intervalo.

Se permiten pequeñas variaciones del aspecto real de las cabezas de flecha como el sombreado o la forma de la flecha. Sin embargo, se recomienda que las cabezas de flecha de intervalos de tiempo puedan distinguirse de las cabezas de flecha de eventos de mensaje o de creación.

En la representación gráfica de un intervalo de temporización absoluta se utiliza una línea de trazo discontinuo para apuntar al evento temporizado de manera absoluta.

Los intervalos pueden dividirse en varias partes, que están conectadas lógicamente por medio de una etiqueta. En el curso de la ejecución, las partes de intervalos divididos se unen para constituir el comienzo y el final de un intervalo.

Las líneas horizontales de los símbolos de temporización absoluta y de temporización relativa pueden representarse por polilíneas arbitrarias, y pueden estirarse y encogerse. Sin embargo, las líneas verticales de los símbolos de temporización relativa tienen que permanecer verticales. La parte final de una polilínea de tiempo absoluto debe ser horizontal y estar asociada con la expresión de tiempo absoluto o con la medición absoluta, respectivamente.

Semántica

En cada trazo de un MSC tiene que haber una correspondencia única entre el comienzo y el final de un intervalo de tiempo dividido (esto es, la unión de intervalos de tiempo divididos se efectúa en la ejecución solamente). Por ejemplo, en un trazo no debe haber dos finalizaciones para un intervalo.

7 Conceptos estructurales

En 7.1 se introducen conceptos estructurales de alto nivel. Las corregiones hacen posible la descripción de áreas en las que los eventos pueden aparecer en cualquier orden. Las expresiones en línea ayudan a estructurar nociones de alternativas, composición paralela y bucles. Se utilizan referencias de MSC para hacer referencia a otros MSC desde dentro de un MSC. Los HMSC hacen abstracciones a partir de ejemplares y dan una visión de conjunto de comportamientos más complicados.

Las referencias de MSC y expresiones en línea pueden tener constricciones de tiempo. La temporización se interpreta con respecto a los eventos de comienzo y terminación, determinados dinámicamente, de la referencia de MSC y la expresión en línea, respectivamente.

7.1 Corregión

En general, la ordenación total de eventos en cada ejemplar (véase 4.2) puede no ser apropiada para las entidades que hacen referencia a un nivel más alto que el de un proceso simple.

En consecuencia, se introduce la corregión para la especificación de eventos no ordenados en un ejemplar. La corregión trata, en particular, el ejemplar importante en la práctica de dos o más mensajes entrantes donde el orden en que se produce el consumo se puede intercambiar. A la inversa, cuando los mensajes son difundidos, el envío de dos o más mensajes salientes se puede intercambiar. Puede definirse una ordenación general mediante relaciones de ordenación general.

Gramática textual concreta

<start coregion>::=
concurrent <end>

<end coregion> ::=
endconcurrent <end>

Gramática gráfica concreta

<concurrent area> ::=
<coregion symbol>
is attached to <instance axis symbol>
contains <coevent layer>

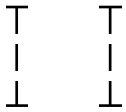
<coregion symbol> ::=
<coregion symbol1> | <coregion symbol2>

<coevent layer> ::=
<coevent area> | <coevent area> ***above*** <coevent layer>

<coevent area> ::=
{ <message event area> | <incomplete message area> | <action area> |
<timer area> | <create area> }*

<coregion symbol1> ::=

<coregion symbol2> ::=



Regla de dibujo: El enunciado de que <coregion symbol> está ligado al <instance axis symbol> significa que el <coregion symbol> tiene que superponerse al <instance axis symbol> como se muestra en la figura 10.

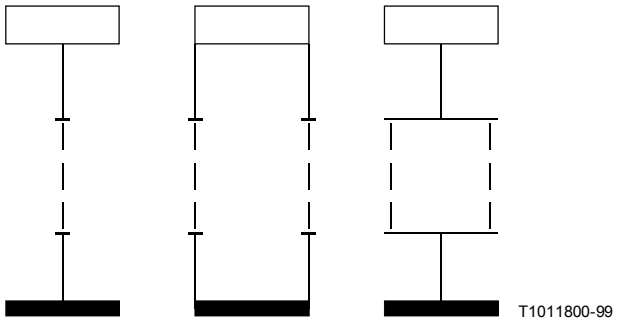


Figura 10/Z.120 – Diferentes formas de corregión

El <coregion symbol1> no debe estar ligado al <instance axis symbol2>.

Requisitos estáticos

En un ejemplar dado, entre <start coregion> y <end coregion> sólo puede haber <orderable event>.

Semántica

Para los MSC, se supone una ordenación total de eventos dentro de cada ejemplar. La corregión permite una excepción: los eventos contenidos en la corregión no están ordenados si no se han prescrito ulteriores constructivos de sincronización ulteriores en forma de relaciones de orden general.

Si un arranque de temporizador y la correspondiente expiración del periodo de temporización o parada están contenidas en un corregión, se supone una relación de ordenación general implícita entre el arranque del temporizador y la expiración del periodo de temporización/parada.

7.2 Expresión en línea

Por medio de expresiones de operador en línea pueden definirse estructuras de composición de evento en un MSC. Los operadores hacen referencia a alternativa, composición paralela, iteración, excepción y regiones facultativas.

Las expresiones en línea temporizadas permiten constreñir o medir el tiempo de ejecución de una alternativa, composición paralela, iteración, excepción y regiones facultativas. Pueden utilizarse intervalos de tiempo relativos e intervalos de tiempo absolutos (con mediciones o sin mediciones).

Los intervalos de tiempo pueden hacer referencia al comienzo y/o final de una expresión en línea. El comienzo es el instante en que se produce dinámicamente el primer evento, y el final es el instante

en que se produce el último evento. En consecuencia, los intervalos de tiempo son aplicables a todos los ejemplares que intervienen en una expresión en línea.

Gramática textual concreta

<shared inline expr> ::=
[<extra-global>]{ <shared loop expr> | <shared opt expr> |
<shared alt expr> | <shared par expr> | <shared exc expr> }
[**startbefore** <time dest list> <end>]
[**startafter** <time dest list> <end>]
[**endbefore** <time dest list> <end>]
[**endafter** <time dest list> <end>]

<extra-global> ::=
external

<shared loop expr> ::=
loop [<loop boundary>] **begin** [<inline expr identification>]<shared> <end>
[<inline gate interface>] [<instance event list>]
loop end [<time interval>] <end>

<shared opt expr> ::=
opt begin [<inline expr identification>] <shared> <end>
[<inline gate interface>] [<instance event list>]
opt end [<time interval>] <end>

<shared exc expr> ::=
exc begin [<inline expr identification>] <shared> <end>
[<inline gate interface>] [<instance event list>]
exc end [<time interval>] <end>

<shared alt expr> ::=
alt begin [<inline expr identification>] <shared> <end>
[<inline gate interface>] [<instance event list>]
{ **alt** <end> [<inline gate interface>] [<instance event list>] }*
alt end [<time interval>] <end>

<shared par expr> ::=
par begin [<inline expr identification>] <shared> <end>
[<inline gate interface>] [<instance event list>]
{ **par** <end> [<inline gate interface>] [<instance event list>] }*
par end [<time interval>] <end>

<inline expr> ::=
[<extra-global>] {<loop expr> | <opt expr> | <alt expr> |
<par expr> | <exc expr>}

<loop expr> ::=
loop [<loop boundary>] **begin** [<inline expr identification>] <end>
[<inline gate interface>] <msc body>
loop end [<time interval>] <end>

<opt expr> ::=
opt begin [<inline expr identification>] <end>
[<inline gate interface>] <msc body>
opt end [<time interval>] <end>

<exc expr> ::=
exc begin [<inline expr identification>] <end>
[<inline gate interface>] <msc body>
exc end [<time interval>] <end>

<alt expr> ::=
alt begin [<inline expr identification>] <end>
[<inline gate interface>] <msc body>
{ **alt** <end> [<inline gate interface>] <msc body> }*
alt end [<time interval>] <end>

<par expr> ::=
par begin [<inline expr identification>] <end>
[<inline gate interface>] <msc body>
{ **par** <end> [<inline gate interface>] <msc body> }*
par end [<time interval>] <end>

<loop boundary> ::=
<left angular bracket> <inf natural> [, <inf natural>]
<right angular bracket>

<inf natural> ::=
inf | <expression>

<inline expr identification> ::=
<inline expr name>

<inline gate interface> ::=
{ **gate** <inline gate> <end>}+

<inline gate> ::=
<inline out gate> | <inline in gate> |
<inline create out gate> | <inline create in gate> |
<inline out call gate> | <inline in call gate> |
<inline out reply gate> | <inline in reply gate> |
<inline order out gate> | <inline order in gate>

Gramática gráfica concreta

<inline expression area> ::=
{ <loop area> | <opt area> | <par area> | <alt area> | <exc area> }
[***top or bottom is attached to top or bottom***
{ {<int symbol> | <abs time symbol> }* } ***set***]
[***is attached to*** <msc symbol>]
[***is followed by*** <general name area>]

<loop area> ::=
<inline expression symbol> [***is attached to*** <time interval area>] ***contains***
{ **loop** [<loop boundary>] <operand area> }
is attached to { <instance axis symbol>* } ***set***
is attached to { <inline gate area>* | <inline order gate area>* } ***set***

<opt area> ::=
<inline expression symbol> [***is attached to*** <time interval area>] ***contains***
{ **opt** <operand area> }
is attached to { <instance axis symbol>* } ***set***
is attached to { <inline gate area>* | <inline order gate area>* } ***set***

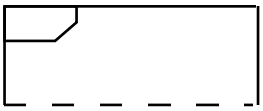
<exc area> ::=
<exc inline expression symbol>
[***is attached to*** <time interval area>] ***contains***
{ **exc** <operand area> }
is attached to { <instance axis symbol>* } ***set***
is attached to { <inline gate area>* | <inline order gate area>* } ***set***

<par area> ::=
<inline expression symbol> [***is attached to*** <time interval area>] ***contains***
{ **par** <operand area>
{ ***is followed by*** <separator area> ***is followed by*** <operand area> }* }
is attached to { <instance axis symbol>* } ***set***
is attached to { <inline gate area>* | <inline order gate area>* } ***set***

<alt area> ::=
<inline expression symbol> [***is attached to*** <time interval area>] ***contains***
{ **alt** <operand area>
{ ***is followed by*** <separator area> ***is followed by*** <operand area> }* }
is attached to { <instance axis symbol> }* ***set***
is attached to {<inline gate area>* | <inline order gate area>* } ***set***

<inline expression symbol> ::=

<exc inline expression symbol> ::=



<operand area> ::=

{ <event layer> | <inline gate area> | <inline order gate area> }* ***set***
[***is followed by*** <general name area>]

<separator area> ::=

<separator symbol>

<separator symbol> ::=

Requisitos estáticos

La <inline expression area> puede hacer referencia a exactamente un ejemplar, o estar ligada a varios ejemplares. Si una expresión en línea atraviesa un ejemplar, que interviene en esta expresión en línea, es facultativo dibujar el eje del ejemplar a través de la expresión en línea.

Todas las expresiones de excepción tienen que ser compartidas por todos los ejemplares en el MSC.

Expresiones en línea extraglobales son aquellas que tienen la palabra clave **external** en la notación textual o que cruzan la trama de MSC en la notación gráfica. Esta última situación gráfica se describe en la gramática como que está "***attached to*** <msc symbol>". Las expresiones extraglobales tienen también que abarcar todos los ejemplares en el MSC.

Las expresiones de datos que definen fronteras de bucle pueden contener variables estáticas y comodines, pero no pueden contener variables dinámicas.

Semántica

Las palabras clave de operador **seq**, **alt**, **par**, **loop**, **opt** y **exc**, que en la representación gráfica se colocan en la esquina superior izquierda, denotan respectivamente composición alternativa, composición paralela, iteración, región facultativa y excepción. En la forma gráfica, una trama encierra los operandos; las líneas de trazo discontinuo representan separadores de operandos.

El operador **seq** representa la operación de recuenciación débil. Un operando de **seq** previsto de una guarda *false* es dinámicamente no válido.

El operador **alt** define ejecuciones alternativas de secciones de MSC. Esto significa que si varias secciones de MSC están destinadas a ser alternativas sólo una de ellas será ejecutada. En el caso de que secciones de MSC alternativas tengan un preámbulo común, la elección de la sección de MSC que será ejecutada se efectúa después de la ejecución del preámbulo común. Los operandos alternativos con una guarda que evalúa a *false* no pueden ser elegidos. Si todos los operandos tienen guardas falsas, ningún trazo legal puede atravesar esta expresión **alt**, es decir, es dinámicamente no válido.

Un operando de una expresión **alt** puede tener como guarda **otherwise**. **Otherwise** se interpreta como la conjunción de las negaciones de las guardas de todos los demás operandos. Por tanto, la guarda **otherwise** es *true* únicamente si las guardas de todos los demás operandos de la expresión alternativa son *false*. Se considera que un operando sin guarda tiene una guarda *true* y es imposible alcanzar la rama **otherwise**.

El operador **par** define la ejecución paralela de secciones de MSC. Esto significa que todos los eventos dentro de las secciones MSC paralelas serán ejecutados, con la única restricción de que se mantiene el orden de los eventos dentro de cada sección. Los operandos paralelos con una guarda que evalúa a *false* se ejecutan a partir de la composición paralela. Si todos los operandos tienen guardas *false*, la expresión **par** evalúa a **empty**.

El constructivo **loop** puede tener varias formas. La forma más sencilla es "**loop** <n,m>", donde n y m son expresiones de tipo números naturales. Esto significa que el operando puede ser ejecutado cuando menos n veces y cuando más m veces. Las expresiones pueden ser reemplazadas por la palabra clave **inf**, como por ejemplo en "**loop** <n,**inf**>". Esto significa que el bucle se ejecutará al menos n veces. Si se omite el segundo operando como por ejemplo en "**loop** <n>", esto se interpreta como "**loop** <n,n>". Por tanto, "**loop** <**inf**>" significa un bucle infinito. Si se omiten los límites del bucle como por ejemplo en "**loop**", esto se interpreta como "**loop** <1,**inf**>". Si el primer operando es mayor que el segundo, el bucle se ejecuta 0 veces. Las pasadas de un bucle están conectadas por medio de la composición secuencial débil.

Cuando el operando de bucle está provisto de una guarda, el bucle se termina cuando la guarda es *false* y se continúa cuando la guarda es *true* en tanto en cuanto no se alcance el límite superior. Por tanto, un bucle será igual a un MSC vacío si la guarda es *false* la primera vez que se entra en el bucle. Si la frontera inferior del bucle no se alcanza debido a la guarda, se interpreta que la totalidad del bucle es dinámicamente ilegal. La frontera superior representa un límite superior al número de iteraciones del bucle.

El operador **opt** equivale a una alternativa en la que el segundo operando es el MSC vacío. Una expresión **opt** provista de guarda pasará siempre a través del operando de opción si la guarda es *true*.

El operador **exc** es una forma compacta de describir casos excepcionales en un MSC. Este operador significa que, o bien los eventos en el interior de <exc inline expression symbol> se ejecutan y después se finaliza el MSC, o que se ejecutan los eventos que siguen al <exc inline expression symbol>. El operador **exc** puede por tanto percibirse como una alternativa en la que el segundo operando es la totalidad del resto del MSC. Todas las expresiones de excepción deben ser compartidas por todos los ejemplares en el MSC. La expresión de excepción es una escritura abreviada para una expresión alternativa donde el resto de la trama circundante es el segundo operando.

En la representación textual, la <inline gate interface> facultativa define los mensajes que entran en la expresión en línea, o salen de ella, mediante puertas. Mediante la identificación de nombre de mensaje y la identificación de nombre de puerta facultativa, la <inline gate interface> define también la conexión de mensaje directa entre dos expresiones en línea.

Las expresiones en línea extraglobales están asociadas con expresiones en línea correspondientes en el ejemplar circundante. Esto significa que cuando se interpreta como descomposición, la expresión en línea combinará sus operandos, uno por uno, con los operandos de otras expresiones en línea. Para una información más detallada, véase 7.4. En el contexto del documento MSC circundante más interior, una expresión en línea extraglobal se interpreta exactamente como cualquier otra expresión en línea.

Las constricciones de tiempo referentes al evento de comienzo y al evento de terminación de una expresión en línea están gráficamente ligadas al <inline expression symbol> y se dan en la sintaxis textual después de las palabras clave **loop end**, **opt end**, **exc end**, **alt end**, y **par end**. Las constricciones de tiempo referentes al comienzo o a la terminación de una expresión en línea son **attached to top or bottom** del <inline expression symbol>. En la sintaxis textual, las palabras clave **startbefore**, **startafter**, **endbefore** y **endafter** se utilizan para indicar que el evento de comienzo de la expresión en línea aparece antes o después de ciertos eventos, o que el evento de terminación de la expresión en línea aparece antes o después de ciertos eventos, respectivamente. Para una explicación más detallada de los constructivos de tiempo, véase 6.7.

7.3 Referencia de MSC

Se utilizan referencias de MSC para hacer referencia a otros MSC del documento MSC. Las referencias de MSC son objetos del tipo dado por el MSC referenciado.

Las referencias de MSC no sólo pueden hacer referencia a un MSC individual, sino también a expresiones de referencia de MSC. Las expresiones de referencia de MSC son expresiones de MSC textuales construidas a partir de los operadores **alt**, **par**, **seq**, **loop**, **opt** y **exc**, y referencias de MSC con parámetros reales posibles.

Los operadores **alt**, **par**, **loop**, **opt** y **exc** se describen en 7.2. El operador **seq** denota la operación de secuenciación débil, donde sólo son ordenados los eventos del mismo ejemplar.

Las referencias de MSC sencillas pueden tener parámetros reales que deben concordar con las correspondientes declaraciones de parámetros de la definición de MSC.

Las puertas reales de la referencia de MSC pueden conectar a constructivos correspondientes en el MSC circundante. Por constructivos correspondientes ha de entenderse que una puerta de mensaje real puede conectar a otra puerta de mensaje real o a un ejemplar o a una definición de puerta de mensaje del MSC circundante. Además, una puerta de ordenación real puede conectar a otra puerta de ordenación real o aun evento ordenable o a una definición de puerta de ordenación.

La utilización sin restricciones de puertas en una referencia de MSC puede llevar fácilmente a la construcción de un MSC que contengan bloqueos inesperados. Por ello, se definen algunas reglas que restringen la utilización gráfica de puertas. La regla básica consiste en que la ordenación vertical gráfica de mensajes con puertas, etc., que aparecen como definiciones de puertas en un MSC referenciado deben ser reproducidos por los mensajes con puertas reales, etc., que aparecen en el MSC referidor. Esta regla impide que el equivalente de la puerta dibuje flechas de mensajes inclinadas en sentido ascendente. El conjunto completo de reglas figura en la sección sobre requisitos estáticos de gráficos.

Se señala además que la aparición de un mensaje con puertas en una referencia de MSC no garantiza el que ese mensaje aparezca en todas las trazas (en realidad, en cualquier traza). Por ejemplo, el mensaje con puertas se puede originar y terminar en expresiones en línea opcionales dentro de sus MSC definidores.

Se utilizan referencias de MSC temporizadas para constreñir o medir el tiempo de ejecución de un MSC referenciado y de expresiones de referencia de MSC.

Los intervalos de tiempo pueden hacer referencia al comienzo y/o la terminación de un referencia de MSC. El valor del comienzo corresponde al instante en que aparece dinámicamente el primer evento, y el valor de la terminación corresponde al instante en que aparece el último evento.

Gramática textual concreta

<shared msc reference> ::=
reference [<msc reference identification> **:**]
<msc ref expr> [**time** <time interval>] <shared> <end>
[**startbefore** <time dest list> <end>]
[**startafter** <time dest list> <end>]
[**endbefore** <time dest list> <end>]
[**endafter** <time dest list> <end>]
<reference gate interface>

<msc reference> ::=
reference [<msc reference identification>**:**]
<msc ref expr> [**time** <time interval>] <end>
<reference gate interface>

<msc reference identification> ::=
<msc reference name>

```
<msc ref expr> ::=
                        <msc ref par expr> { alt <msc ref par expr> }*

<msc ref par expr> ::=
                        <msc ref seq expr> { par <msc ref seq expr> }*

<msc ref seq expr> ::=
                        <msc ref ident expr> { seq <msc ref ident expr> }*

<msc ref ident expr> ::=
                        loop [ <loop boundary> ] <msc ref ident expr> |
                        exc <msc ref ident expr> |
                        opt <msc ref ident expr> |
                        empty |
                        <parent>* <msc name> [<actual parameters>]   |
                        ( <msc ref expr> )

<actual parameters> ::=
                        ([<actual data parameters>]
                        [<actual instance parameters>][<actual message parameters>]
                        [<actual timer parameters>] )

<actual instance parameters> ::=
                        inst <actual instance parm list> <end>

<actual instance parm list> ::=
                        <actual instance parameter> [,<actual instance parm list>]

<actual instance parameter> ::=
                        <instance name>

<actual message parameters> ::=
                        msg <actual message list> <end>

<actual message list> ::=
                        <message name> [, <actual message list>]

<actual timer parameters> ::=
                        timer <actual timer list> <end>

<actual timer list> ::=
                        <timer name> [, <actual timer list>]

<parent> ::=
                        #

<reference gate interface> ::=
                        { <end> gate <ref gate> }*

<ref gate> ::=
                        <actual out gate> | <actual in gate> |
                        <actual order out gate> | <actual order in gate> |
                        <actual create out gate> | <actual create in gate> |
                        <actual out call gate> | <actual in call gate> |
                        <actual out reply gate> | <actual in reply gate>
```

Requisitos estáticos

Una referencia de MSC debe ligar a todo ejemplar presente en el diagrama circundante que está contenido en el MSC a que se refiere la referencia de MSC. Si dos diagramas referenciados por dos referencias de MSC en un diagrama circundante comparten los mismos ejemplares, estos ejemplares deben aparecer también en el diagrama circundante.

La interfaz de la referencia de MSC debe concordar con la interfaz de los MSC referenciados en la expresión, es decir, toda puerta ligada a la referencia debe tener una correspondiente definición de puerta en los MSC referenciados. La correspondencia viene dada por el sentido de flujo y el nombre del mensaje asociado con la puerta y, si está presente, por el nombre de puerta, que es único dentro de la expresión referenciada.

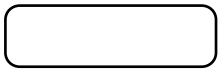
Cuando <msc ref expr> consta de una expresión de operador textual en vez de un <msc name> simple, y cuando más de una referencia de MSC se refiere al mismo MSC, el <msc reference name> facultativo en la <msc reference identification> tiene que ser empleado para direccionar una referencia de MSC en la definición de mensaje (véase 4.3).

La <reference gate interface> debe indicar todas las puertas del diagrama.

Gramática gráfica concreta

<msc reference area> ::=
<msc reference symbol>
[***top or bottom is attached to top or bottom***
{ {<int symbol> | <abs time symbol> }* } ***set***]
contains { <msc ref expr> [**time** <time interval>]
[<actual gate area>*] } ***set***
is attached to { <instance axis symbol>* } ***set***
is attached to { <actual gate area>* } ***set***

<msc reference symbol> ::=



La <msc reference area> puede ser ligada a uno o más ejemplares. Sin un <msc reference area> compartida atraviesa un <instance axis symbol> que no interviene en la referencia de MSC, el <instance axis symbol> se dibuja en forma pasante.

Requisitos estáticos

El orden gráfico de las puertas formales de la definición MSC a que se refiere la referencia de MSC debe ser conservado por las puertas reales de la referencia de MSC. En la notación textual, esto significa que el orden de la <msc gate interface> es el mismo orden de las puertas reales en la <reference gate interface>.

Cuando una referencia de MSC contiene expresiones de referencia la situación es más compleja. Por lo general se ha de construir un conjunto de ordenaciones verticales posibles a partir de las partes constituyentes de la expresión de MSC. La ordenación vertical real utilizada en la referencia de MSC debe proceder entonces de ese conjunto.

Cuando una expresión consta de un nombre de MSC simplemente, se supone que el conjunto de ordenaciones verticales consta de un elemento simplemente, a saber, la ordenación vertical definida por el MSC referenciado.

Para una expresión alternativa, cada operando tiene que definir el mismo conjunto de ordenaciones verticales, que se supone entonces que es el conjunto que representa la expresión. Esta definición abarca también expresiones opcionales y excepcionales, que se toman como formas abreviadas de alternativas.

En el caso de una expresión paralela el conjunto de ordenaciones verticales se construye intercalando las ordenaciones definidas por cada operando. Si se tiene, por ejemplo, *e* **par** *f*, donde *e* define el conjunto de puertas ordenadas verticalmente {<a, b>, <b, a>} y *f* define el conjunto {<u, v>}, hay que intercalar <u, v> con <a, b> para obtener seis ordenaciones derivadas, e intercalar además <u, v> con <b, a> para obtener otras seis ordenaciones. Así pues, *e* **par** *f* define un conjunto de doce ordenaciones verticales posibles. La ordenación de puertas real debe ser por tanto una de esas doce ordenaciones.

El conjunto de ordenaciones definidas por *e* **seq** *f* se obtiene añadiendo cada una de las ordenaciones definidas por *f* al final de cada una de las ordenaciones definidas por *e*. Por ejemplo, si *e* define el conjunto de puertas ordenadas verticalmente {<a, b>, <b, a>} y *f* define el conjunto {<u, v>}, el conjunto resultante es {<a, b, u, v>, <b, a, u, v>}.

Las referencias de MSC no deben hacer referencia, directa ni indirectamente, a su MSC circundante (o sea, no deben ser recursivas).

Los parámetros reales de MSC deben concordar con las correspondientes declaraciones de parámetros de la definición de MSC. Los elementos separados por <end> en los <actual parameters> tienen que mantener una correspondencia biunívoca con los elementos separados por <end> en la <parameter declaration> de la definición de MSC.

Los parámetros de ejemplar reales deben ser de la misma clase que la declaración de parámetros del ejemplar. La clase de los parámetros de ejemplar reales puede ser una clase heredada de la clase de la declaración de parámetros de ejemplar.

Los mensajes reales deben tener la misma firma (de mensaje) que el mensaje de la declaración de parámetro de mensaje. El hecho de que un mensaje tenga la misma firma significa que la lista de parámetros del mensaje es la misma para el mensaje real y para la declaración de parámetros del mensaje.

Los temporizadores reales deben tener la misma firma (de temporizador) que la declaración de parámetros de temporizador. El hecho de que un temporizador tenga la misma firma significa que la lista de parámetros del temporizador es la misma para el temporizador real y para la declaración de parámetros de temporizador.

Un MSC que contiene referencias y vinculaciones de parámetros reales es ilegal si, tras repetidas expansiones de las referencias y vinculaciones de parámetros reales, se obtiene como resultado un MSC ilegal.

Semántica

Cada MSC puede percibirse como una definición de un tipo de MSC. Se pueden utilizar tipos de MSC en otros tipos de MSC por medio de referencias de MSC.

Un tipo de MSC puede ser ligado a su entorno por medio de puertas. Se utilizan puertas para definir puntos de conexión cuando un tipo de MSC se utiliza en otro tipo. Los identificadores de puerta pueden asociarse a los puntos de conexión en forma de nombres.

Los ejemplares que están ligados a la referencia de MSC son parámetros de ejemplar reales por defecto para la referencia de MSC. Esto significa que si la clase de ejemplar está identificando unívocamente el parámetro de ejemplar formal, no es necesario que los ejemplares ligados aparezcan en la lista de parámetros reales.

Una referencia de MSC se puede ligar a ejemplares que no están contenidos en el diagrama referenciado.

En general, la referencia de MSC puede aludir a una expresión de MSC textual. En el caso simple, cuando la expresión de MSC consiste de un nombre de MSC solamente, la referencia de MSC apunta a una definición de tipo de MSC correspondiente. La correspondencia viene dada por el nombre de MSC, que es único en el documento MSC. Cuando el MSC tiene parámetros reales, el resultado es que los parámetros formales del diagrama referenciado son reemplazados por los parámetros reales de la referencia de MSC.

En la representación textual, la <reference gate interface> define los mensajes que entran en la referencia de MSC o salen de ella, a través de puertas. Mediante la identificación de nombre de mensaje y la facultativa identificación de nombre de puerta, la <reference gate interface> define también la conexión de mensaje directa entre dos referencias de MSC.

Una referencia de MSC con la palabra clave **empty** alude a un MSC que no contiene ni eventos ni ejemplares.

Un prefijo <parent> de una referencia de MSC indica que la referencia de MSC alude al MSC que ella redefine por herencia y no al MSC con ese nombre dentro de la clase del ejemplar heredero. El prefijo <parent> puede repetirse para tener acceso a abuelos (*grandparents*), etc.

Las constricciones de tiempo referentes al evento de comienzo y al evento de terminación de una expresión de MSC están gráficamente ligados al <msc reference symbol> y se dan en la sintaxis textual tras la palabra clave **time**. Las constricciones de tiempo referentes al comienzo o a la terminación de una expresión de MSC son **attached to top or bottom** del <msc reference symbol>. En la sintaxis textual, las palabras clave **startbefore**, **startafter**, **endbefore**, y **endafter** se utilizan para indicar que el evento de comienzo de la expresión de MSC aparece antes o después de ciertos eventos, o que el evento de terminación de la expresión de MSC aparece antes o después de ciertos eventos, respectivamente.

7.4 Descomposición de ejemplares

Un documento MSC contiene un conjunto de ejemplares. Cada ejemplar es de una clase de ejemplar (*instance kind*). Los ejemplares que no tienen referencia a la clase de ejemplar tienen implícitamente la clase de ejemplar que tiene el mismo nombre que el ejemplar. Para describir la interacción en diferentes niveles de detalle, MSC introduce la descomposición.

La estructura y comportamiento interiores de una clase de ejemplar se definen mediante una documentación MSC que tiene el mismo nombre de la clase de ejemplar. Por tanto, habrá una jerarquía de documentos MSC que definen la jerarquía del ejemplar. Para definir cómo se relacionan los diferentes niveles se puede especificar que el comportamiento de un ejemplar dentro de un diagrama MSC se redefina en un MSC del documento MSC que define el ejemplar que se descompone.

Así, un documento MSC puede interpretarse únicamente con relación a sus propios ejemplares, sin tener en cuenta una descomposición, o puede interpretarse con relación a niveles inferiores de ejemplares siguiendo las relaciones de descomposición.

El MSC requiere que haya una similitud estructural entre el ejemplar descompuesto y la descomposición correspondiente, pero no es necesario que haya un afinamiento del comportamiento.

Gramática textual concreta

<decomposition> ::=
decomposed [<substructure reference>]

<substructure reference> ::=
as <message sequence chart name>

Gramática gráfica concreta

La gramática gráfica concreta se presenta en 4.2.

Requisitos estáticos

Para cada ejemplar que contiene la palabra clave **decomposed** hay que especificar un correspondiente MSC de afinamiento con el mismo nombre que el ejemplar descompuesto. Si la <substructure reference> se añade después de **decomposed** el MSC de afinamiento debe tener el mismo nombre especificado en la <substructure reference>.

El ejemplar descompuesto puede entenderse como una secuencia de constructivos de lenguaje algunos de los cuales deben interpretarse como puertas en relación con el diagrama de descomposición. En la figura 11 se presenta una visión de conjunto de los conceptos procesables por puerta (*gateable concepts*). Existen *peer gates* (puertas pares) y *decomposition gates* (puertas de descomposición). Las puertas pares se conocen también a partir de la conexión de referencias de MSC, mientras que las puertas de descomposición sólo ofrecen interés para los ejemplares descompuestos.

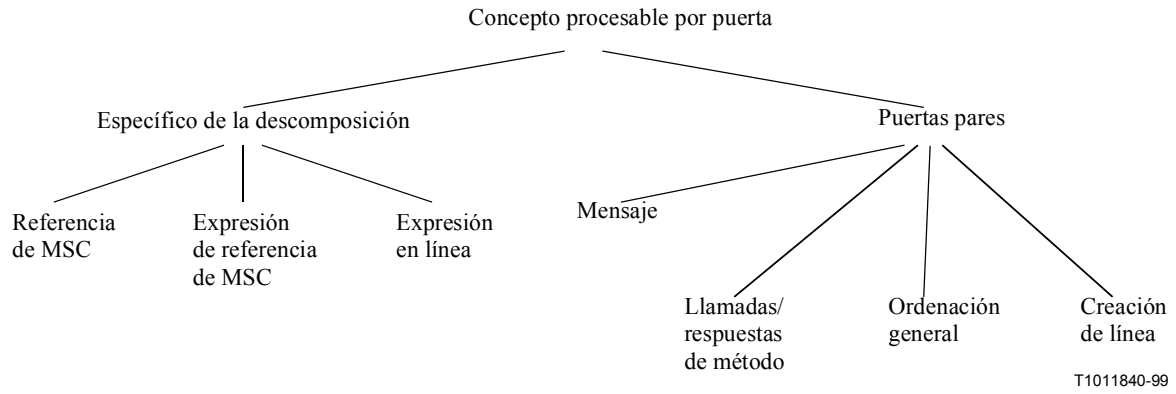


Figura 11/Z.120 – Conceptos procesables por puertas

La interpretación de los conceptos de puerta específicos de la descomposición se da en la sección de semántica más adelante. La interpretación de las puertas pares se da en la cláusula sobre puertas y llamadas de método.

Desde el punto de vista estático, el orden gráfico de las puertas debe conservarse desde el ejemplar descompuesto hasta el diagrama de descomposición, lo que significa que si los dos bordes verticales del diagrama de descomposición están superpuestos, la secuencia de las definiciones de puertas del diagrama debe ser la misma que la del ejemplar descompuesto. En la notación textual esto significa que el orden de la <msc gate interface> es igual al orden de los eventos correspondientes en el ejemplar descompuesto.

Sea el ejemplar *i* en el MSC *M* del documento MSC *D* de la figura 12. A lo largo del ejemplar hay, en secuencia, una expresión alternativa, una referencia de MSC simple (a *A*), una salida de mensaje *s* y una expresión de referencia de MSC (*B* **alt** *C*). La descomposición se encuentra en MSC *iM* de la figura 16 y contiene en secuencia una expresión en línea extraglobal, una referencia de MSC simple (a *iA*), una puerta par de salida de mensaje *s* y una expresión de referencia de MSC (*iB* **alt** *iC*).

Con las puertas pares, esta secuenciación es simple pues hay eventos en el ejemplar descompuesto para concordar con puertas pares en la descomposición. Con conceptos procesables por puerta específicos de la descomposición la situación es algo más compleja. Los constructivos en un ejemplar que no concuerdan con un concepto procesable por puerta no se tienen en cuenta como requisitos estáticos en la descomposición. Tales constructivos son acciones, temporizadores y corregiones. Las corregiones se consideran a este respecto como un área de ejemplar sencillo.

Cuando el ejemplar descompuesto es detenido, la descomposición debe incluir paradas en todos los ejemplares que ella contiene.

Requisitos estáticos

Los siguientes requisitos estáticos deben cumplirse para referencias de MSC que abarcan un ejemplar dado. Sea *i* un ejemplar descompuesto dentro del MSC *M* (figura 12). Sea *i* en *M* descompuesto como *iM* (figura 16). Considérese que existe una referencia de MSC *A* que abarca *i* dentro de *M*. Esta *A* es entonces una referencia de MSC procesable por puerta. Con ella debe concordar una referencia global[2] correspondiente en *iM* a (por ejemplo) *iA* (figura 17) definida en el documento MSC que define *i*. *i* dentro de *A* debe descomponerse entonces como *iA* (figura 13).

Los siguientes requisitos estáticos se cumplen para expresiones en línea que abarcan un ejemplar descompuesto dado (véase la figura 12). La descomposición debe contener una expresión en línea correspondiente de la misma estructura de operación y operandos (véase la figura 16). La expresión

[2] Por "global" ha de entenderse que "abarca todos los ejemplares en el documento MSC".

en línea en la descomposición debe ser extraglobal (véase expresiones en línea) lo que indica que los operandos están conectados a otros operandos de expresiones en línea similares cuando se interpretan mediante descomposición. Cuando se interpreta solamente en el contexto del documento MSC circundante más próximo, una expresión en línea extraglobal se trata como una expresión en línea global simple.

Recursivamente, cada operando de la expresión en línea tiene los mismos requisitos estáticos que el ejemplar descompuesto completo.

El siguiente requisito estático se cumple para expresiones de referencia de MSC que abarcan un ejemplar descompuesto dado. La descomposición debe contener una expresión de referencia de MSC correspondiente con la misma estructura de expresión. Cada uno de los operandos de la expresión de referencia de MSC debe respetar los requisitos estáticos para expresiones de referencia de MSC o referencias de MSC.

Un ejemplar que se descompone en un MSC debe descomponerse en todos los MSC que dependen de este MSC mediante referenciación.

Semántica

La semántica de descomposición se define mediante un modelo de transformación. A continuación se presenta un algoritmo que transformará un MSC, incluida la descomposición de un documento MSC, en otro MSC de otro MSC (construido), en el que el ejemplar descompuesto ha sido dividido en sus componentes. Las transformaciones serán acompañadas de un ejemplo.

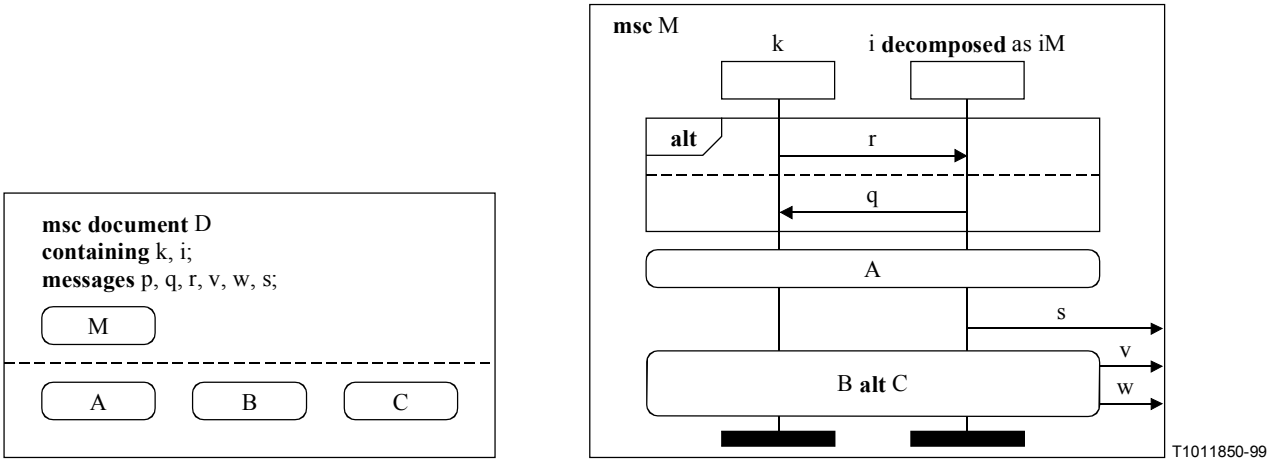


Figura 12/Z.120 – Documento MSC del nivel más alto y MSC definidor

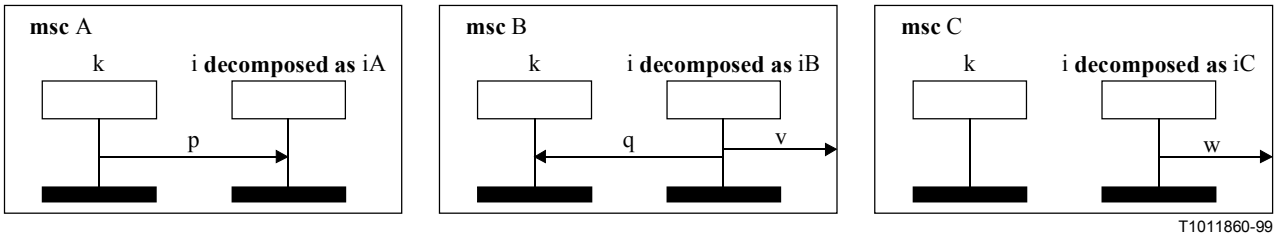


Figura 13/Z.120 – Utilidades de nivel superior

Regla 1: Transformar expresiones de referencia de MSC en expresiones en línea. Véase la figura 14.

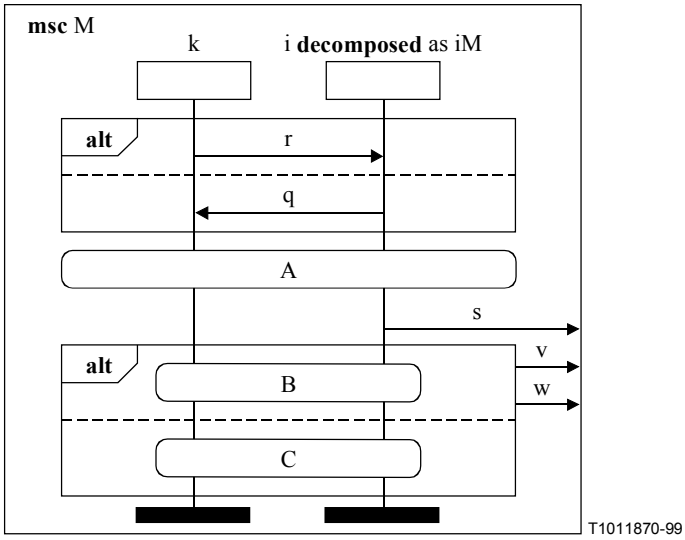


Figura 14/Z.120 – Situación después de aplicar la regla 1

Regla 2: Resolver las referencias de MSC sustituyendo las referencias, dondequiera que aparezcan, por el contenido de los diagramas MSC. Obtener la concordancia de las puertas. Véase la figura 15.

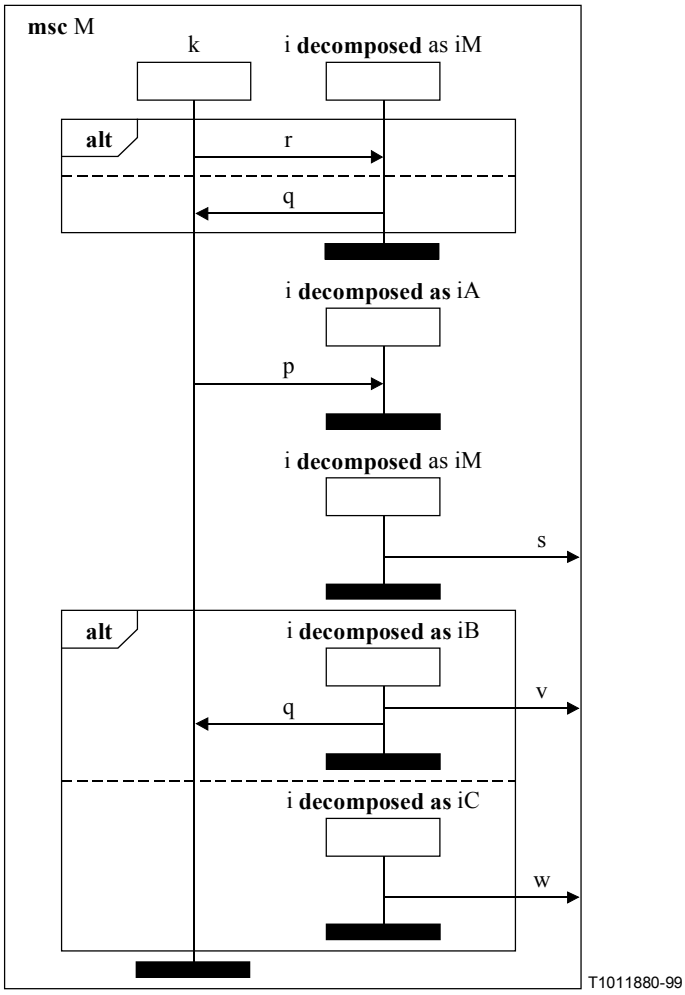


Figura 15/Z.120 – Situación después de aplicar la regla 2

El algoritmo ha llegado ahora a un diagrama en el que la descomposición se produce pieza por pieza y de acuerdo con los requisitos estáticos. Los ejemplares que no están descompuestos se conectan mediante puertas en la forma más simple.

Ahora el algoritmo pasa a la descomposición y es necesario mostrar la definición del documento MSC *i*.

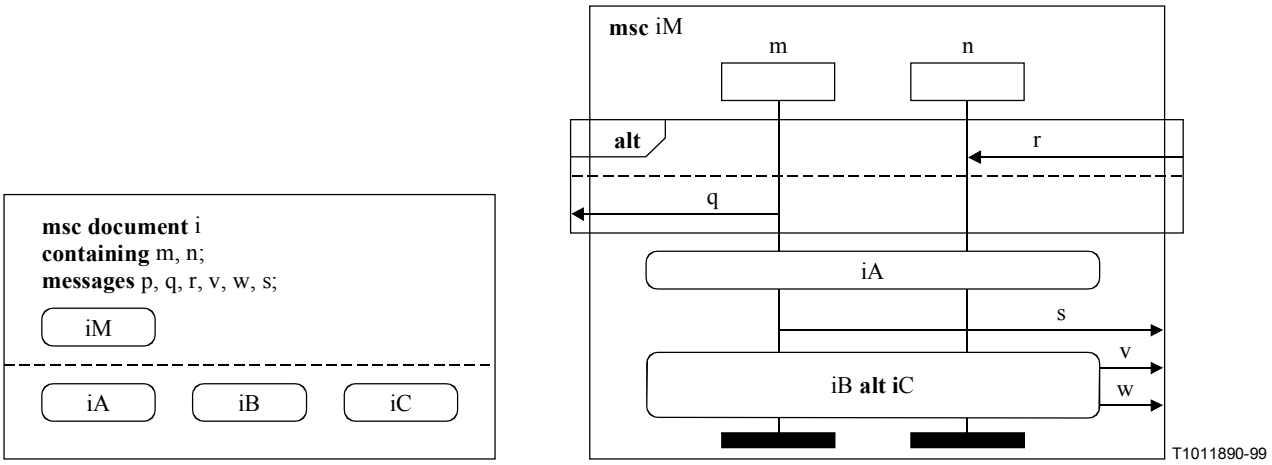


Figura 16/Z.120 – Documento MSC de nivel inferior

Obsérvese la expresión en línea extraglobal y que los requisitos estáticos se cumplen en *iM* en relación con *i en M del documento MSC D* en la figura 12.

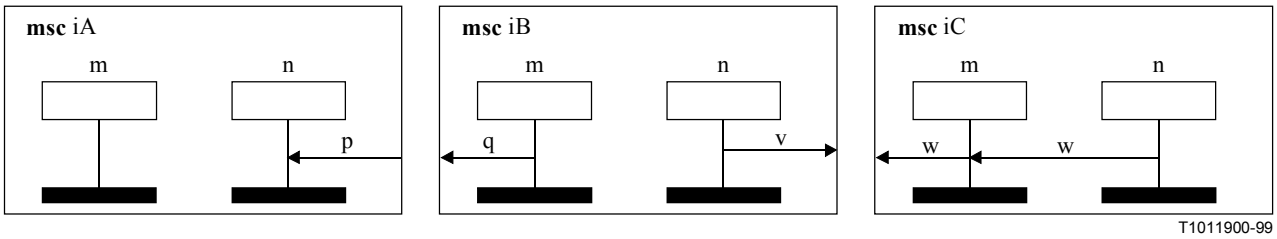


Figura 17/Z.120 – Utilidades en el nivel más bajo

Regla 3: Transformar los ejemplares descompuestos con simples puertas pares mediante una pura sustitución por el contenido del diagrama y concordancia de las puertas pares. Véase la figura 18.

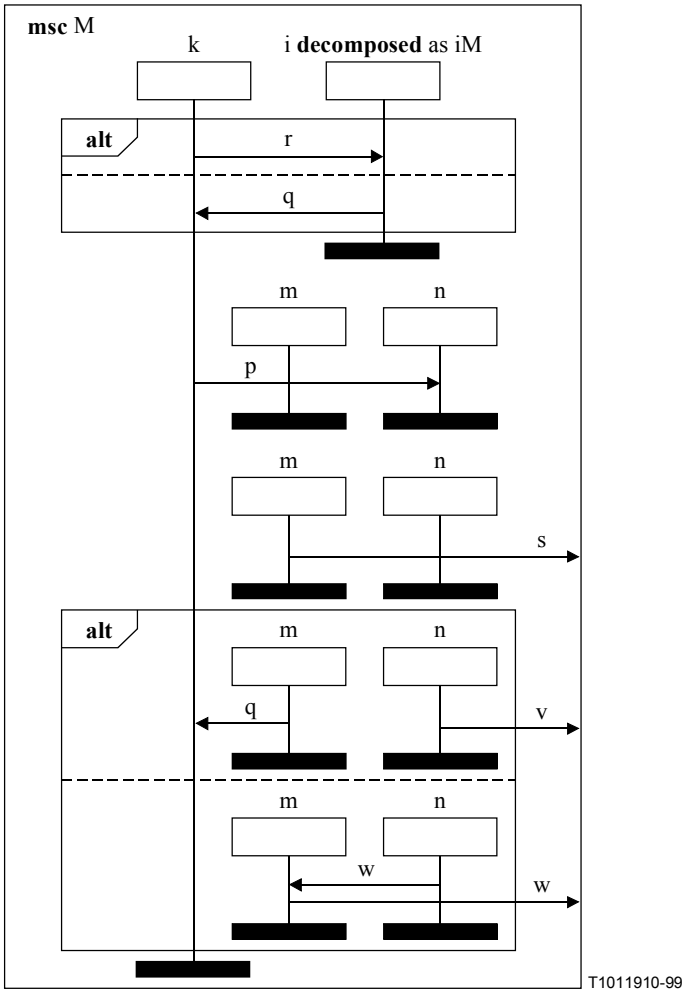


Figura 18/Z.120 – Situación después de aplicar la regla 3

La regla 3 da instrucciones para la sustitución por MSC simples y la concordancia de sus puertas. En el ejemplo pueden verse las sustituciones y los sustitutos. Desde el punto de partida en la figura 15 considérese la descomposición de *i* en *M* dada por *iM*, que se encuentra en la figura 16. De arriba a abajo se producen las siguientes sustituciones. Primero, se deja la expresión en línea a la siguiente regla más abajo. Segundo, en *iM* está la referencia a *iA* que corresponde bien con el fragmento indicado. Se usa *iA* en la figura 17 para la sustitución. Tercero, está la salida de *s* descompuesta directamente en *iM*. Se utiliza la porción de *iM* entre la referencia a *iA* y la expresión de referencia de MSC como sustituto. Cuarto, se llega a la expresión de referencia de MSC (*iB* **alt** *iC*) y se encuentra que ésta corresponde bien con la expresión en línea. Se efectúa la sustitución aplicando los mismos principios en de cada operando.

Regla 4: Transformar expresiones en línea sustituyendo con expresiones en línea extraglobales correspondientes tales que los operandos se conectan en una relación uno-uno. Cuando hay expresiones anidadas, esta regla se aplica recursivamente en cada operando. Véase la figura 19.

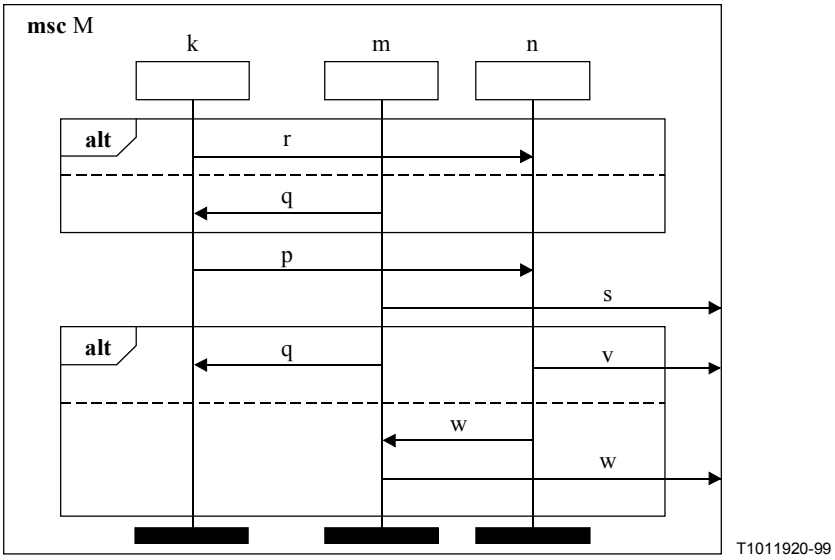


Figura 19/Z.120 – Situación después de aplicar la regla 4

En la figura 19, la regla 4 se ha aplicado y la expresión extraglobal ha sido ha sido insertada por sustitución operando por operando. Además, el diagrama ha sido simplificado combinando cada fragmento a un ejemplar.

El MSC M resultante es un documento MSC D' que contendría los ejemplares k, m, n.

El esquema de transformación podría haber sido igualmente bien expresado efectuando primero la descomposición, antes de resolver las referencias. Se habría necesitado una notación más extensa para la intersección entre una referencia de MSC (expresión) y el contenido de un diagrama de descomposición.

7.5 MSC de alto nivel (HMSC)

Los MSC de alto nivel proporcionan medios para definir gráficamente cómo puede combinarse un conjunto de los MSC. Un HMSC es un gráfico dirigido en el que cada nodo puede ser:

– un símbolo de comienzo (hay un solo símbolo de comienzo en cada HMSC);
– un símbolo de terminación;
– una referencia de MSC;
– una condición;
– un punto de conexión; o
– una trama paralela.

Las líneas de flujo conectan los nodos del HMSC e indican la secuenciación que es posible entre los nodos del HMSC. Las líneas de flujo entrantes se conectan siempre al borde superior de los símbolos del nodo en tanto que las líneas de flujo salientes se conectan al borde inferior. La existencia de más de una línea de flujo saliente desde un nodo indica una alternativa.

Las referencias de MSC pueden utilizarse para referenciar un solo MSC o varios MSC empleando una expresión de MSC textual.

Las condiciones en los HMSC pueden utilizarse para indicar estados de sistema o guardas globales e imponer restricciones a los MSC referenciados en el HMSC.

Las tramas paralelas contienen uno o más HMSC pequeños e indican que los HMSC pequeños son los operandos de un operador paralelo, es decir que los eventos de diferentes HMSC pequeños pueden estar entrelazados.

Los puntos de conexión se introducen para simplificar la representación de los HMSC y no tienen significado semántico.

Los MSC de alto nivel pueden ser constreñidos y medidos con intervalos de tiempo para expresiones de MSC. Además, el tiempo de ejecución de una trama paralela puede ser constreñido o medido. Su interpretación es similar a la interpretación de expresiones MSC temporizadas.

Gramática textual concreta

<hmsc> ::=
expr <msc expression>

<msc expression> ::=
<start> {<node expression> | <text definition>} *

<start> ::=
<label name list> <end>

<node expression> ::=
<label name> **:** { <node> **seq** (<label name list>) | **end** }<end>

<label name list> ::=
<label name> { **alt** <label name> }*

<node> ::=
(<msc ref expr>) [<time interval>]
| <par expression> [<time interval>]
| <condition identification>
| **connect**

<par expression> ::=
expr <msc expression> **endexpr**
{ **par expr** <msc expression> **endexpr** }*

Requisitos estáticos

Todos los <label name> referenciados en <start> o <node expression> deben estar definidos en el HMSC, es decir deben aparecer como "<label name>**:**" en una <node expression>. Desde <start> se debe poder llegar a cada nodo del gráfico HMSC, esto es, el gráfico debe ser un gráfico conectado.

Gramática gráfica concreta

<mscexpr area> ::=
{<text layer> <start area> <node expression area>*
<hmsc end area>* } ***set***

<start area> ::=
<hmsc start symbol> ***is followed by*** { <alt op area>$^{+}$ } ***set***

<hmsc start symbol> ::=

<hmsc end area> ::=
<hmsc end symbol> ***is attached to*** { <hmsc line symbol>$^{+}$} ***set***

<hmsc end symbol> ::=

<hmsc line symbol> ::=
<hmsc line symbol1> | <hmsc line symbol2>

<hmsc line symbol1> ::=

<hmsc line symbol2> ::=

<alt op area> ::=
<hmsc line symbol> ***is attached to*** { <node area> | <hmsc end symbol> }

<node expression area> ::=
<node area> ***is followed by*** { <alt op area>$^{+}$} ***set***
is attached to { <hmsc line symbol>$^{+}$} ***set***

<node area> ::=
<hmsc reference area>
| <connection point symbol>
| <hmsc condition area>
| <par expr area>

<hmsc reference area> ::= <msc reference symbol>
[***top or bottom is attached to top or bottom***
{ {<int symbol> | <abs time symbol> }* } ***set***]
contains <msc ref expr> [**time** <time interval>]

<connection point symbol> ::=

<hmsc condition area> ::=
<condition symbol> ***contains*** <condition text>

<par expr area> ::=
<par frame symbol>
is attached to <hmsc line symbol>
[***top or bottom is attached to top or bottom***
{ {<int symbol> | <abs time symbol> }* } ***set***]
contains { <mscexpr area>+} ***set***

<par frame symbol> ::=
<frame symbol>

Reglas de dibujo

El <hmsc line symbol> puede ser una línea quebrada y tener cualquier sentido.

El que <hmsc start symbol> vaya seguido del <alt op area> significa que los <hmsc line symbol> deben estar unidos a la esquina inferior del <hmsc start symbol> como se indica en la figura 20 en el que <hmsc line symbol>s de dos líneas siguen a un <hmsc start symbol>:

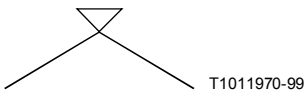


Figura 20/Z.120 – Reglas de dibujo de HMSC

El que el <hmsc line symbol> esté unido a otro símbolo en la regla de producción <alt op area> significa que los <hmsc line symbol> deben estar unidos al borde superior del símbolo en cuestión. En los casos en que el símbolo sea un <msc reference symbol> y un <hmsc end symbol>, ello se ilustra en la figura 21:

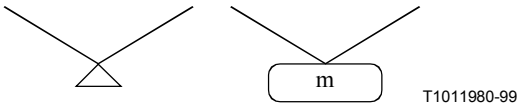


Figura 21/Z.120 – Reglas de dibujo de HMSC

El que la <node area> vaya seguida de una <alt op area> en la regla de producción <node expression area> significa que los <hmsc line symbol> deben estar unidos al borde inferior del símbolo de la <node area>, como se indica en la figura 22 que muestra cómo un <msc reference symbol> va seguido de una <alt op area>:

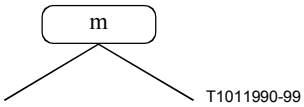


Figura 22/Z.120 – Reglas de dibujo de HMSC

Semántica

El gráfico que describe la composición de los MSC dentro de un HMSC se interpreta de una manera operacional como sigue. La ejecución comienza en el <hmsc start symbol>. Después continúa con un nodo que sigue a uno de los bordes de salida de este símbolo. Se considera que estos nodos son operandos de un operador **alt** (véase 7.2). Tras la ejecución del nodo seleccionado, el proceso de selección y ejecución se repite para los bordes de salida del nodo seleccionado. La ejecución de un nodo significa que la ejecución de un determinado HMSC termina. La ejecución de una referencia de MSC se efectúa de acuerdo con la descripción en 7.3. La ejecución de un punto de conexión es una operación vacía. La ejecución de una trama paralela consiste en ejecutar en paralelo los operandos de una trama paralela, como se describe en 7.2 para el operador **par**. Una ejecución secuencial de dos nodos que están relacionados por un borde se describe por el operador **seq** (véase 7.3).

Una condición de guarda significa que la ejecución no puede continuar más allá de la condición si evalúa a *false*. Si todas las ramas disponibles están bloqueadas por condiciones de guarda *false*, y no se ha llegado a una terminación de HMSC, el HMSC completo no tiene trazos legales.

EJEMPLOS DE GRÁFICOS DE SECUENCIAS DE MENSAJES

Esta parte es informativa, no normativa.

8 Documento de gráficos de secuencias de mensajes

8.1 Documentos MSC

El diagrama de documento MSC de la figura 23 muestra que el ejemplar *ACContext* contiene los ejemplares *ACSystem*, *User*, *Supervisor* y *NewUser*. Los diagramas MSC definidores (o públicos) son *UserAccess*, *PIN_Change* y *NewUser*. Obsérvese que *NewUser* es el nombre de un ejemplar contenedor y el nombre de un diagrama MSC contenido. Esto es legal porque los ejemplares y diagramas son de clases de entidad diferentes.

Se presenta asimismo la interfaz del lenguaje de datos. El lenguaje de datos "C" y el carácter de subrayado se utiliza como comodín de variables. Las definiciones de datos están registradas en el fichero cdefs.h.

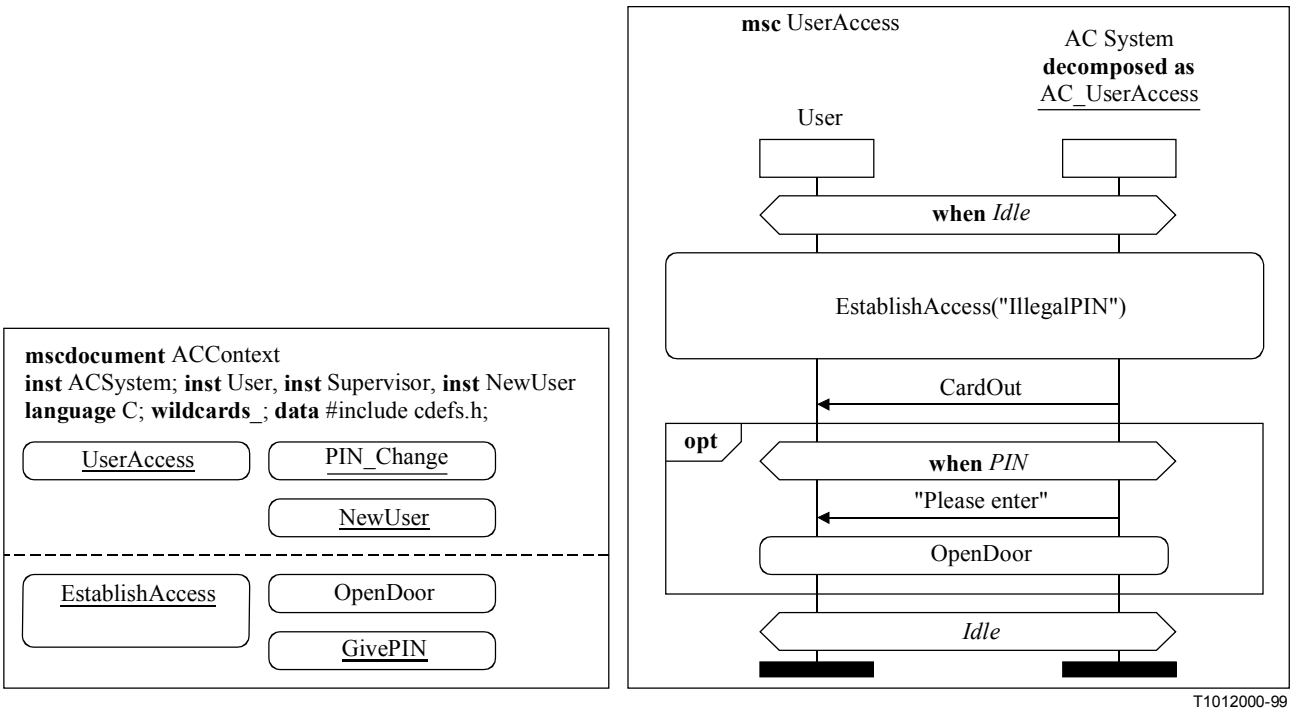


Figura 23/Z.120 – Documento MSC ACContext

Para ilustrar los requisitos estáticos de consistencia en la descomposición se ha presentado también el diagrama MSC *UserAccess* referenciado desde el documento MSC *ACContext*.

8.2 Descomposición de ejemplar

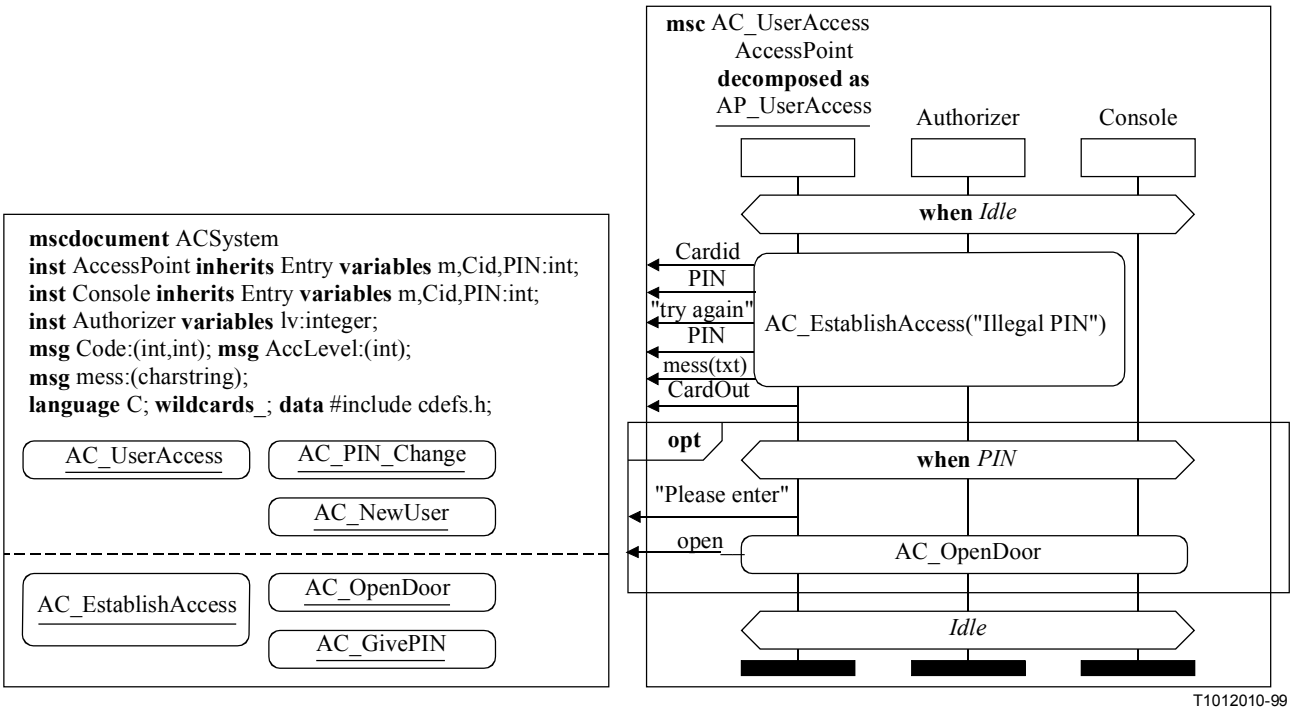


Figura 24/Z.120 – Documento MSC en el nivel inferior inmediato

El documento MSC *ACSystem* presentado en la figura 24 es entonces la descripción del ejemplar *ACSystem* presentada en la figura 23. Muestra también la herencia descrita para los ejemplares contenidos. Más adelante, en la figura 27, se muestra la definición de la clase de ejemplar *Entry*.

Se han presentado también las declaraciones de variables dinámicas de los ejemplares y los mensajes que tienen parámetros.

Los requisitos estáticos para descomposición y las referencias de MSC pueden comprobarse en el ejemplo. En *ACContext* está el MSC *UserAccess* donde el ejemplar contenido *ACSystem* es descompuesto por *AC_UserAccess* que a su vez hace referencia a *AC_EstablishAccess*.

Por otro lado, el *UserAccess* hace referencia a *EstablishAccess* mostrado en la figura 25.

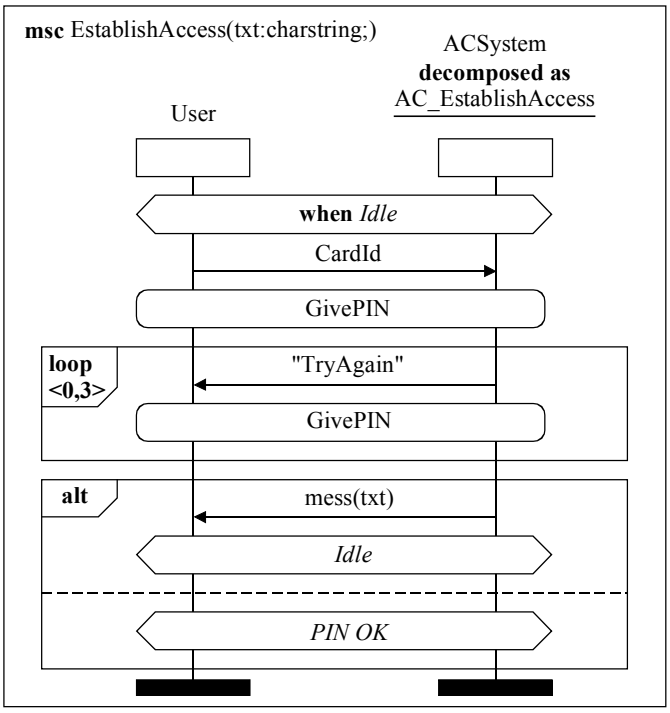


Figura 25/Z.120 – MSC EstablishAccess

Dentro de *EstablishAccess*, que es una utilidad de *ACContext*, el ejemplar *ACSystem* se descompone en *AC_EstablishAccess*. *AC_EstablishAccess* representa por tanto los puntos de confluencia de la referenciación y la descomposición a partir de UserAccess de *ACContext*. En otras palabras, independientemente de que la descomposición o la referenciación se produzca primero, no debe haber diferencias al final.

En *EstablishAccess* hay una definición de una variable estática *txt*. Los parámetros reales pueden verse en la figura 24.

Para ofrecer una exposición más completa, en la figura 26 se muestra *AC_EstablishAccess*.

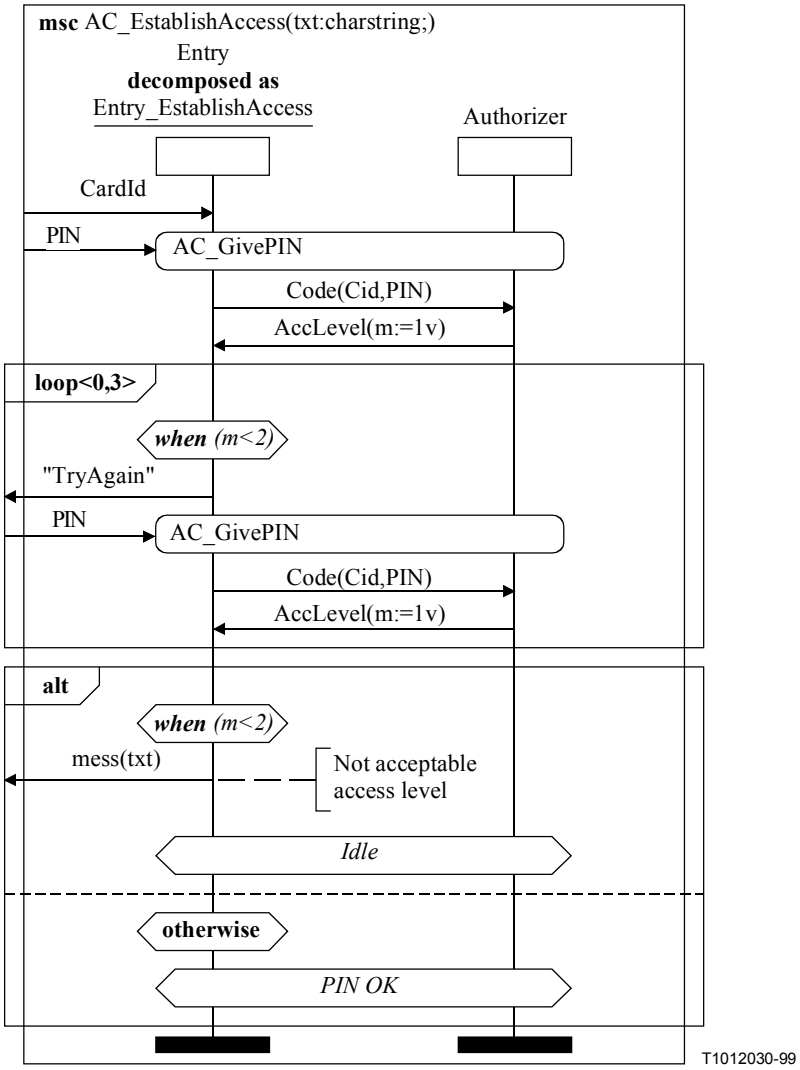


Figura 26/Z.120 – MSC AC_EstablishAccess

En *AC_EstablishAccess* hay condiciones similares a estados utilizadas como guardas y fijación de valores. Hay también expresiones booleanas utilizadas como guardas. En algunos de los mensajes se muestran vinculaciones.

8.3 Herencia de ejemplar

Véanse las figuras 27 y 28.

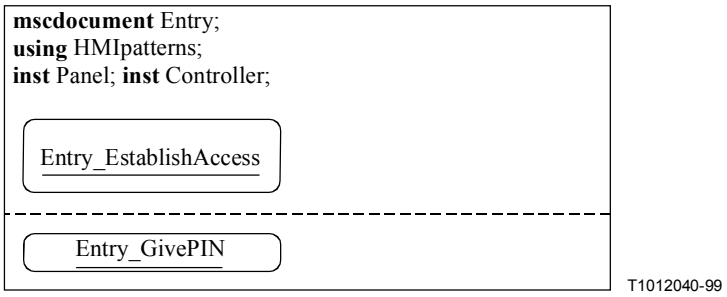


Figura 27/Z.120 – Documento MSC Entry

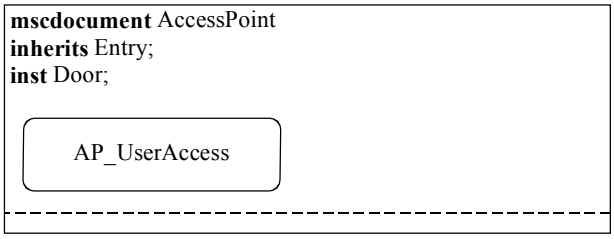


Figura 28/Z.120 – Punto de acceso de MSC que hereda de Entry

En el documento MSC *AccessPoint* se ve que no es necesario definir de nuevo los ejemplares *Panel* y *Controller*, pues ya están definidas en la *Entry* heredada.

9 Gráficos de secuencias de mensajes simples

9.1 MSC básico

En este ejemplo se muestra un establecimiento de conexión simplificado en un sistema de conmutación. Se presentan los constructivos de MSC más básicos: ejemplares (procesos), entorno, mensajes y condiciones globales.

La condición Idle (reposo) es una condición inicial, la condición Seizure (toma) es una condición intermedia, y la condición Talking (conversación) es una condición final.

Véase la figura 29.

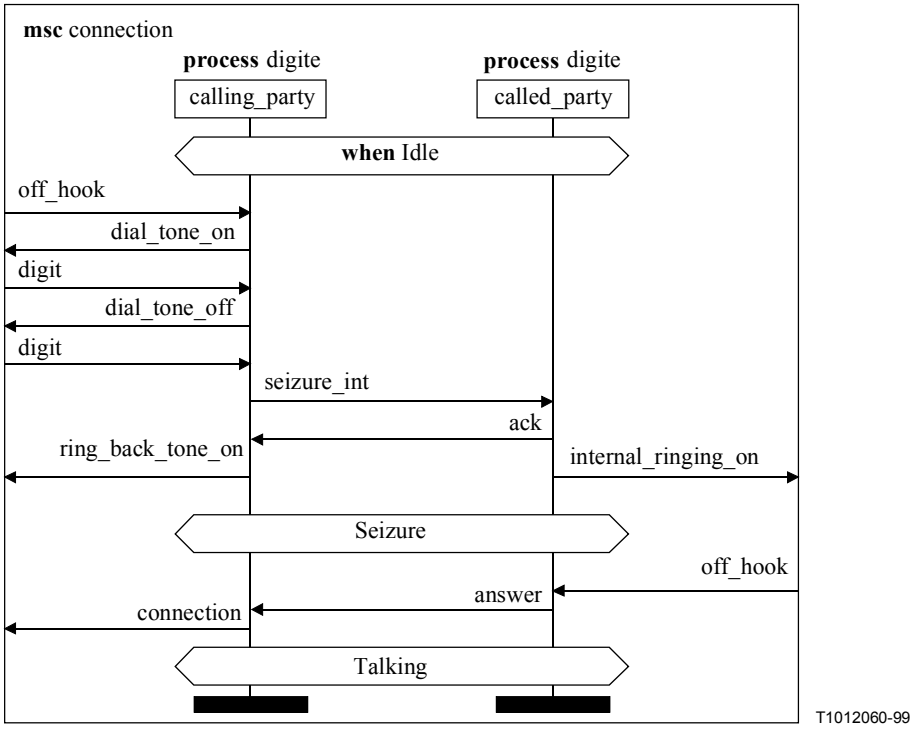


Figura 29/Z.120 – MSC conexión

```
msc connection;
inst calling_party: process digite;
inst  called_party: process digite;
gate out off_hook to calling_party;
gate in dial_tone_on from calling_party;
gate out digit to calling_party;
gate in dial_tone_off from calling_party;
gate out digit to calling_party;
gate in ring_back_tone_on from calling_party;
gate in internal_ringing_on from called_party;
gate out off_hook to called_party;
gate in connection from calling_party;

        calling_party : instance process digite;
                condition when Idle shared all;
                in off_hook from env;
                out dial_tone_on to env;
                in digit from env;
                out dial_tone off to env;
                in digit from env;
                out seizure_int to called_party;
                in ack from called_party;
                out ring_back_tone on to env;
                condition Seizure shared all;
                in answer from called_party;
                out connection to env;
                condition Talking shared all;
        endinstance;
        called_party: instance process digite;
                condition when Idle shared all;
                in seizure_int from calling_party;
                out ack to calling_party;
                out internal_ringing_on to env;
                condition Seizure shared all;
                in off_hook from env;
                out answer to calling_party;
                condition Talking shared all;
        endinstance;
endmsc;
```

9.2 Adelantamiento de un mensaje por otro mensaje

En este ejemplo se muestra el adelantamiento de un mensaje que tiene por nombre 'message1' por otro mensaje que tiene el mismo nombre. En la representación textual se emplean los nombres de ejemplar de mensaje (a, b) para una correspondencia unívoca entre entrada y salida de mensaje. En la representación gráfica, los mensajes se representan, sea por flechas horizontales, una de las cuales está doblada para indicar que un mensaje se adelanta a otro, sea por flechas inclinadas hacia abajo que se cruzan. Véase la figura 30.

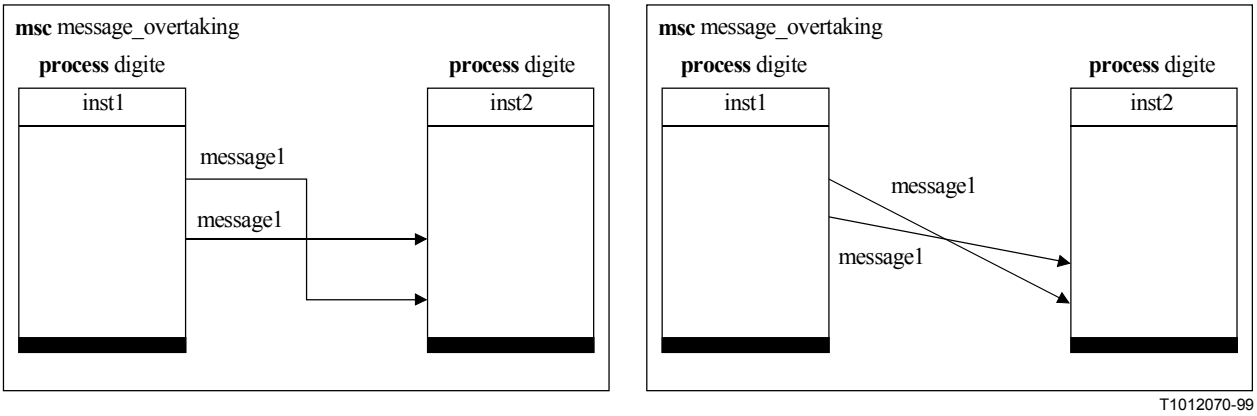


Figura 30/Z.120 – MSC message_overtaking

```
msc message_overtaking;
inst inst1;
inst inst2;
        inst1: instance process digite;
            out message1, a to inst2;
            out message1, b to inst2;
        endinstance;
        inst2: instance process digite;
            in message1, b from inst1;
            in message1, a from inst1;
        endinstance;
endmsc;
```

9.3 Conceptos básicos de MSC

Este ejemplo contiene los constructivos básicos de MSC: ejemplares, entorno, mensajes, condiciones, acciones y expiración de periodo de temporización. En la representación gráfica se utilizan los dos tipos de símbolos de ejemplar: la forma de línea simple y la forma de columna. Véase la figura 31.

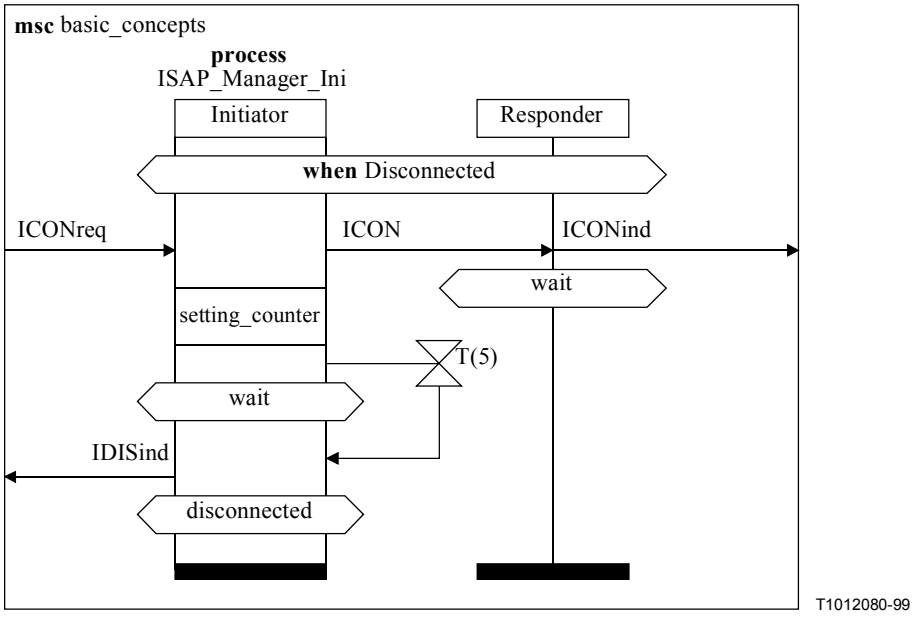


Figura 31/Z.120 – MSC basic_concepts

Sintaxis textual orientada a ejemplares

```
msc basic_concepts;
inst Initiator: process ISAP_Manager_Ini;
inst Responder;
gate out ICONreq to Initiator;
gate in ICONind from Responder;
gate in IDISind from Initiator;

        Initiator: instance process ISAP_Manager_Ini;
            condition when Disconnected shared all;
            in ICONreq from env;
            out ICON to Responder;
            action "setting_counter";
            starttimer T (5);
            condition wait shared;
            timeout T;
            out IDISind to env;
            condition disconnected shared;
        endinstance;
        Responder: instance;
            condition when Disconnected shared all;
            in ICON from Initiator;
            out ICONind to env;
            condition wait shared;
        endinstance;
endmsc;
```

Sintaxis textual orientada a eventos

```
msc basic_concepts;
inst Initiator: process ISAP_Manager_Ini;
inst Responder;
gate out ICONreq to Initiator;
gate in ICONind from Responder;
gate in IDISind from Initiator;

Initiator:  instance process ISAP_Manager_Ini;
Responder:  instance;
all:        condition when Disconnected;
Initiator:  in ICONreq from env;
            out ICON to Responder;
Responder:  in ICON from Initiator;
            out ICONind to env;
            condition wait;
            endinstance;
Initiator:  action "setting_counter";
            starttimer T (5);
            condition wait;
            timeout T;
            out IDISind to env;
            condition disconnected;
            endinstance;
endmsc;
```

9.4 Composición de MSC mediante condiciones etiquetadas

En este ejemplo se ilustra la composición de MSC mediante condiciones globales. La condición global final 'Wait_For_Resp' de MSC 'connection_request' es idéntica a la condición global inicial de la confirmación de conexión de MSC. Por consiguiente, ambos MSC pueden componerse para formar el MSC 'connection' resultante (véase 9.5).

La composición se define mediante el HMSC 'con_setup' ilustrado en la figura 34.

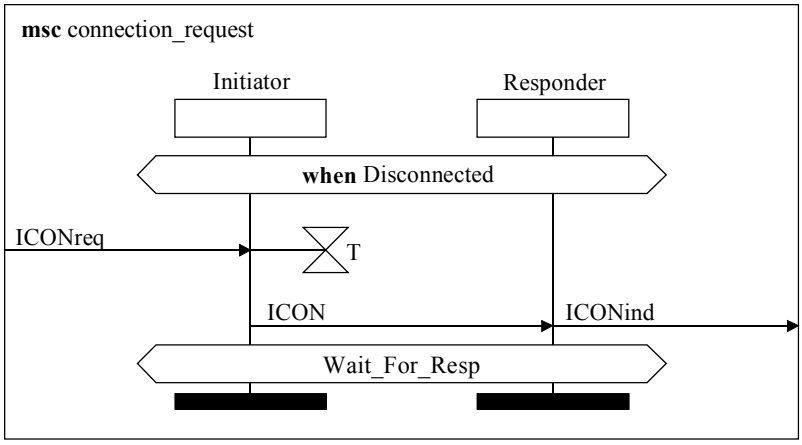


Figura 32/Z.120 – MSC connection_request

```
msc connection_request;
inst Initiator;
inst Responder;
gate out ICONreq to Initiator;
gate in ICONind from Responder;

        instance Initiator;
            condition when Disconnected shared all;
            in ICONreq from env;
            starttimer T;
            out ICON to Responder;
            condition Wait_For_Resp shared all;
        endinstance;
        instance Responder;
            condition when Disconnected shared all;
            in ICON from Initiator;
            out ICONind to env;
            condition Wait_For_Resp shared all;
        endinstance;
endmsc;
```

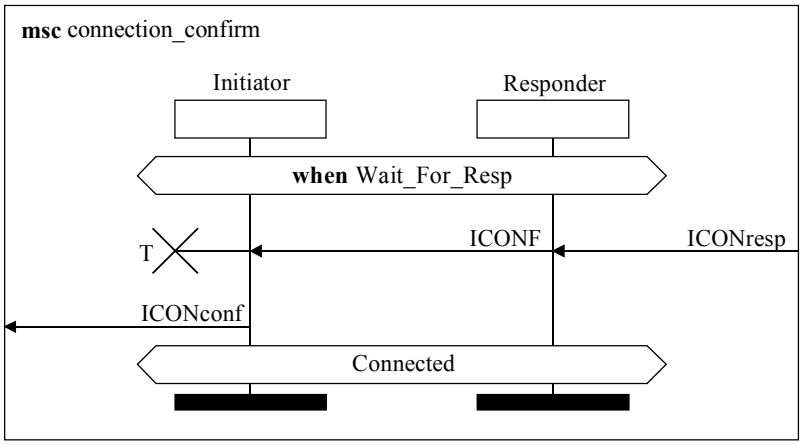


Figura 33/Z.120 – MSC connection_confirm

```
msc connection_confirm;
inst Initiator;
inst Responder;
gate in ICONconf from Initiator;
gate out ICONresp to Responder;

        instance Initiator;
            condition when Wait_For_Resp shared all;
            in ICONF from Responder;
            stoptimer T;
            out ICONconf to env;
            condition Connected shared all;
        endinstance;
        instance Responder;
            condition when Wait_For_Resp shared all;
            in ICONresp from env;
            out ICONF to Initiator;
            condition Connected shared all;
        endinstance;
endmsc;
```

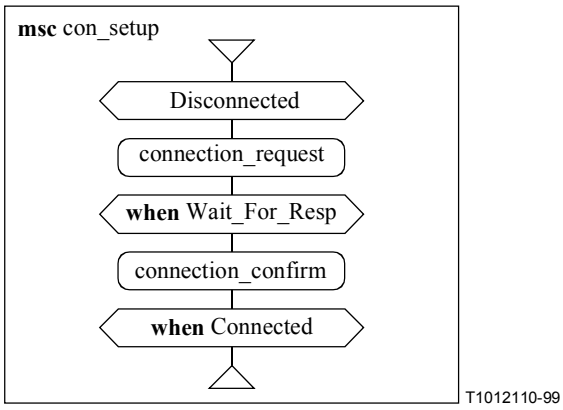


Figura 34/Z.120 – MSC con_setup

```
msc con_setup;
expr L1;
        L1: condition Disconnected seq ( L2 );
        L2: (connection_request) seq (L3);
        L3: condition when Wait_For_Resp seq ( L4);
        L4: (connection_confirm) seq ( L5 );
        L5: condition when Connected seq ( L6 );
        L6: end;
endmsc;
```

9.5 MSC con supervisión de tiempo

El MSC 'connection' de este ejemplo contiene una parada de temporizador. Véase la figura 35.

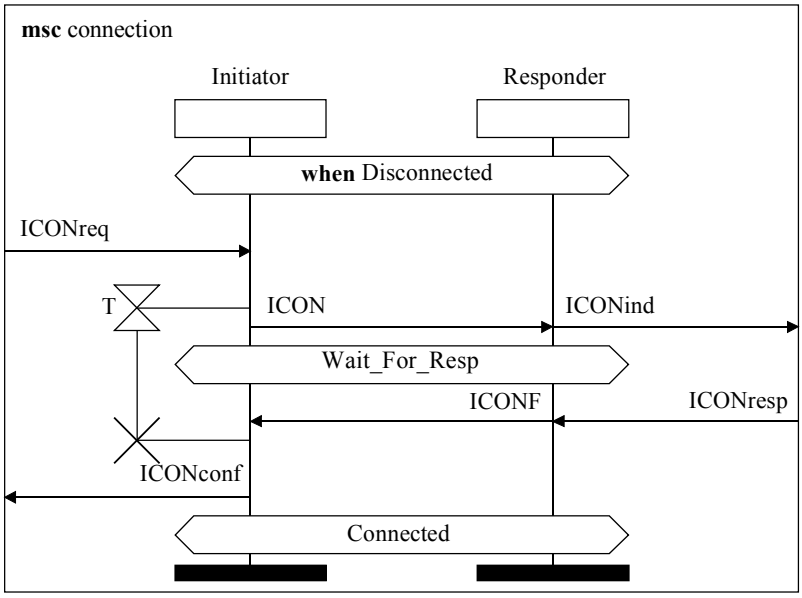


Figura 35/Z.120 – MSC connection

```
msc connection;
inst Initiator;
inst Responder;
gate out ICONreq to Initiator;
gate in ICONind from Responder;
gate out ICONresp to Responder;
gate in ICONinf from Initiator;

        Initiator: instance;
            condition when Disconnected shared all;
            in ICONreq from env;
            starttimer T;
            out ICON to Responder;
            condition Wait_For_Resp shared all;
            in ICONF from Responder;
            stoptimer T;
            out ICONconf to env;
            condition Connected shared all;
        endinstance;
        Responder: instance;
            condition when Disconnected shared all;
            in ICON from Initiator;
            out ICONind to env;
            condition Wait_For_Resp shared all;
            in ICONresp from env;
            out ICONF to Initiator;
            condition Connected shared all;
        endinstance;
endmsc;
```

9.6 MSC con pérdida de mensajes

El MSC 'failure' de este ejemplo contiene una expiración de temporizador debida a un mensaje perdido. Véase la figura 36.

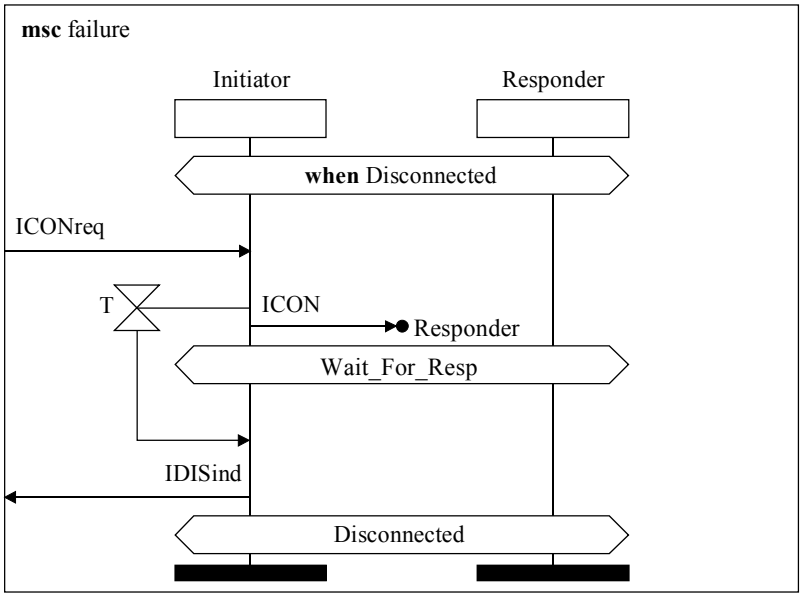


Figura 36/Z.120 – MSC failure

```
msc failure;
inst Initiator;
inst Responder;
gate out ICONreq to Initiator;
gate in IDISind from Initiator;

        Initiator: instance;
            condition when Disconnected shared all;
            in ICONreq from env;
            starttimer T;
            out ICON to lost Responder;
            condition Wait_For_Resp shared all;
            timeout T;
            out IDISinf to env;
            condition Disconnected shared all;
        endinstance;
        Responder: instance;
            condition when Disconnected shared all;
            in ICON from Initiator;
            condition Wait_For_Resp shared all;
            condition Disconnected shared all;
        endinstance;
endmsc;
```

9.7 Condiciones locales

En este ejemplo se emplean condiciones locales referentes a un ejemplar para indicar estados locales de este ejemplar. Véase la figura 37.

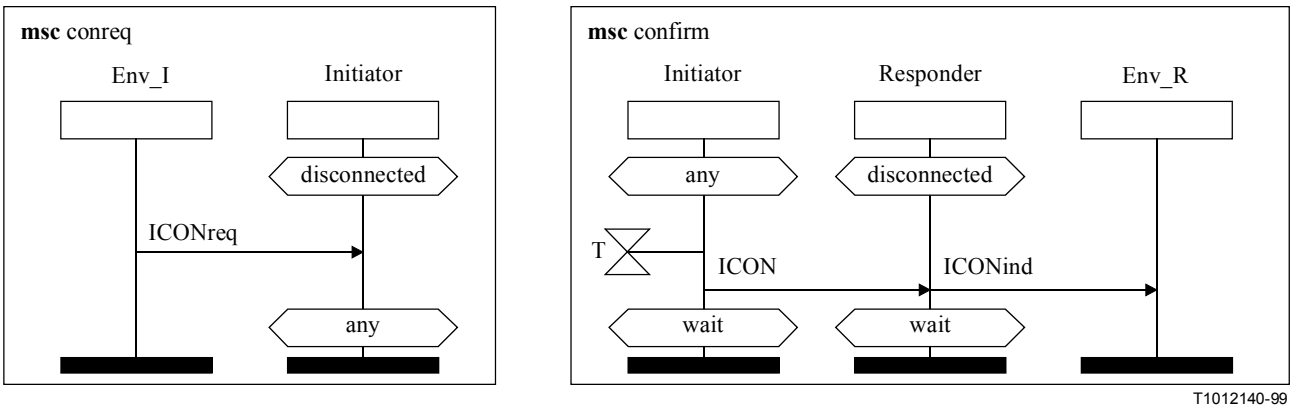


Figura 37/Z.120 – Condiciones locales

```
msc conreq;
inst Env_I;
inst Initiator;
        Env_I: instance;
            out ICONreq to Initiator;
        endinstance;
        Initiator: instance;
            condition disconnected shared;
            in ICONreq from Env_I;
            condition any shared;
        endinstance;
endmsc;

msc confirm;
inst Initiator;
inst Responder, Env_R;
        Initiator: instance;
            condition any shared;
            starttimer T;
            out ICON to Responder;
            condition wait shared;
        endinstance;
        Responder: instance;
            condition disconnected shared;
            in ICON from Initiator;
            out ICONind to Env_R;
            condition wait shared;
        endinstance;
        Env_R: instance;
            in ICONind from Responder;
        endinstance;
endmsc;
```

10 Datos

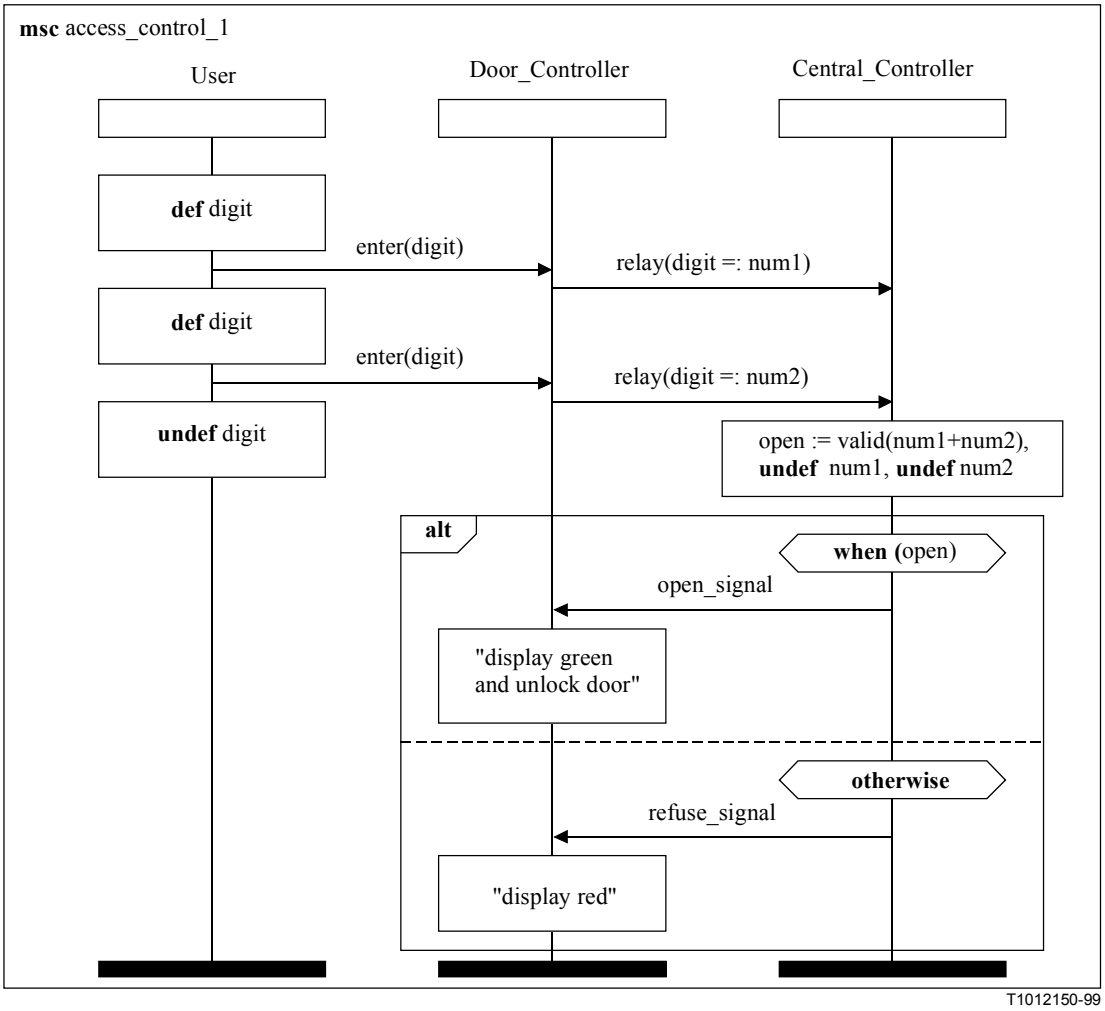


Figura 38/Z.120 – Datos en casillas de acción, mensajes y condiciones

En la figura 38 se supone que el ejemplar *User* posee la variable *digit*, y que el ejemplar *Central_Controller* posee las variables *num1* y *num2*; la declaración de estas variables aparecería en el documento MSC circundante que se muestra en la figura 39.

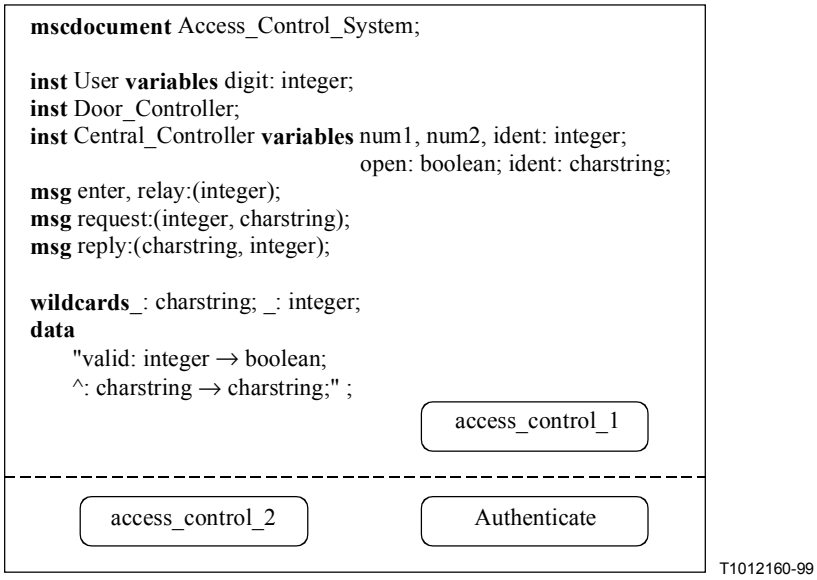


Figura 39/Z.120 – Documento MSC para Access_Control_System

La primera casilla de acción en el ejemplar *User* define la variable *digit*, que tiene por efecto asignar a *digit* un valor no especificado; desígnese este valor por *v1*. Este valor se pasa entonces al ejemplar *Door_Controller* a través del parámetro uno del mensaje *enter*. El parámetro *digit* aparece como una expresión; no es una vinculación porque no hay símbolo de vincular, y los parámetros estáticos prohíben que sea un patrón, pues *enter* es un mensaje completado, es decir, su fuente y su destino son ejemplares. Este valor *v1* de *digit* está ahora disponible para el ejemplar *Door_Controller* hasta que se reciba un nuevo valor de *digit*, independientemente de lo que le suceda al valor de *digit* en el ejemplar que lo posee.

En el primer mensaje *relay* el valor *v1* de *digit* se pasa como un parámetro a través de la parte expresión de la vinculación *digit* =: *num1*. La vinculación hace que el valor *v1* se vincule a *num1* en el ejemplar *Central_Controller*.

La segunda casilla de acción en *User* redefine el valor de *digit* a otro valor no especificado, por ejemplo *v2*. La semántica permite que los valores *v1* y *v2* sean iguales. Este segundo valor se envía a *Door_Controller* a través del segundo mensaje *enter*. Obsérvese que los dos valores de *digit* pueden coexistir en un trazo a través del MSC, aunque sólo un valor es conocido por el ejemplar en un momento dado cualquiera.

En el segundo mensaje *relay* el valor *v2* de *digit* se vincula a la variable *num2* en el ejemplar *Central_Controller*. Por tanto, el valor de *num1* es *v1* y el valor de *num2* es *v2*.

La acción final en *User* indefine *digit*, lo que significa que esta variable se convierte en una variable no vinculada de *User*. Por tanto, toda ulterior referencia a *digit* en este ejemplar es ilegal, lo que podría suceder si otro MSC fuera secuenciado después de *access_control_1*. La operación de indefinir una variable explícitamente retira dicha variable del alcance hasta que sea vinculada mediante una vinculación o un enunciado (*statement*) def.

La casilla de acción en *Central_Controller* contiene tres enunciados que se evalúan en paralelo. Toda expresión se evalúa primero utilizando el estado antiguo, por lo que la expresión *valid*(*num1* + *num2*) se evalúa utilizando *v1* como valor de *num1* y *v2* como valor de *num2*. Después de evaluada cualquier expresión, el estado se actualiza de acuerdo con las vinculaciones, los enunciados def (si están presentes) y los enunciados undef. En este caso *open*, está vinculado al resultado de la expresión *valid*(*num1* + *num2*), donde la función booleana *valid* se declara/define en la sección datos del documento MSC circundante (figura 39), y las variables *num1* y *num2* pasan a estar no vinculadas, esto es, fuera de alcance. Como los enunciados se evalúan en paralelo, pueden

darse en cualquier orden sin que por ello se cambie el significado. En consecuencia, el enunciado undef podría haberse dado antes que el enunciado que vincula a *open*.

El valor de *open* se utiliza como una guarda en la expresión alternativa en línea. La guarda satisface los requisitos estáticos en cuanto a que las variables de guarda (en este caso, precisamente, *open*) están poseídas por el ejemplar activo, que es *Central_Controller*. La alternativa que se seleccione depende del valor de *open*; si es *true* se elige la primera alternativa, y en caso contrario la segunda alternativa.

Obsérvese que las casillas de acción que aparecen en la expresión en línea contienen texto informal, como se indica por los caracteres de comillas que encierran el texto.

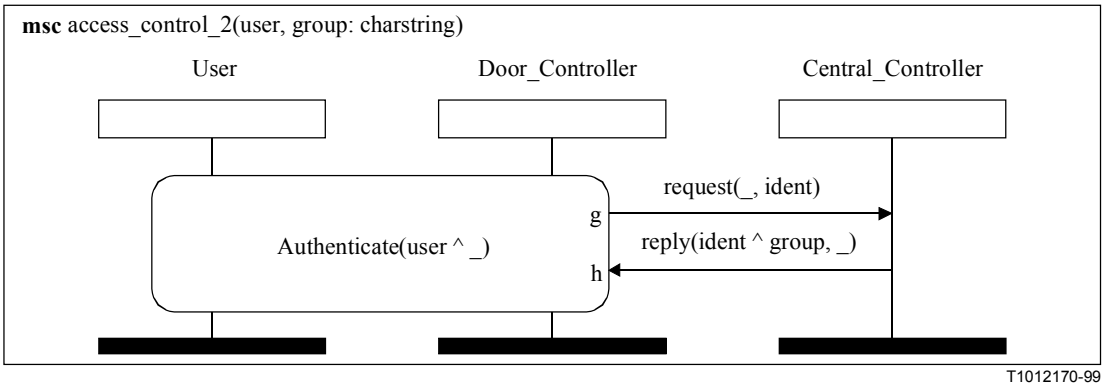


Figura 40/Z.120 – Datos estáticos y datos de mensaje incompleto

La figura 40 muestra MSC *access_control_2*, que tiene dos parámetros formales estáticos, *user* y *group*, ambos de tipo *charstring*. Si *access_control_2* es referenciada en otro MSC, se deben dar dos parámetros reales para estos dos parámetros formales. La referencia de MSC *Authenticate* tiene un parámetro real, la expresión *user* ^ _, donde "^" representa concatenación de cadenas (definida/declarada en la sección datos del documento MSC, figura 39), y "_" representa un comodín de tipo *charstring* (declarado en el documento MSC circundante, figura 39). el MSC *access_control_2* utiliza también una variable dinámica *ident*, que es poseída por el ejemplar *Central_Controller*.

La expresión de parámetro real dada a *Authenticate* no puede contener variables dinámicas, pero sí comodines y variables estáticas, como en este caso. La expresión *user* ^ _ denota cualquier cadena que comienza con cualquier valor dado en la variable *user*.

El mensaje *request* es un mensaje incompleto que tiene dos parámetros. Es incompleto porque se origina en la puerta *g*. Los requisitos estáticos para un mensaje incompleto que termina en un ejemplar prescriben que los parámetros deben constar solamente de patrones. Por tanto, el primer parámetro "_" es un patrón de comodín que no produce ninguna vinculación, y el segundo es el patrón *ident*, lo que dará por resultado que *ident* se vinculará dinámicamente a cualquier valor que haya sido enviado desde el MSC *Authenticate*. Obsérvese que sólo variables dinámicas poseídas por el ejemplar *Central_Controller* o comodines pueden aparecer como patrones en mensajes recibidos por este ejemplar.

El mensaje *reply* es también un mensaje incompleto que tiene dos parámetros, pero esta vez es incompleto porque termina en una puerta (en este caso *h*). Los requisitos estáticos prescriben que los parámetros deben constar solamente de expresiones, por lo que el primer parámetro es la expresión *ident* ^ *group*, y el segundo es el comodín "_". Obsérvese que las expresiones de parámetro pueden contener variables estáticas (en este caso *ident*), variables dinámicas (en este caso *group*) y comodines.

El valor que se dé a la expresión *ident ^ group* es una cadena cuya parte inicial consiste en todo lo que se envíe en el segundo parámetro del mensaje *request*, y la segunda parte consiste en cualquier valor que se suministra para la variable estática *group* en una referencia al MSC *access_control_2*. El segundo parámetro de *reply* puede ser cualquier cadena. Los valores de estos dos parámetros se envían con el mensaje *reply* y serán dinámicamente vinculados a los patrones encontrados en la correspondiente lista de parámetros *reply* en recepción encontrada en el MSC *Authenticate*.

11 Tiempo

En el ejemplo de esta subcláusula se muestran las principales características de los conceptos de tiempo en los MSC. Los ejemplos se han tomado de una especificación de prueba de la calidad de funcionamiento para un servidor de sesión de acceso TINA (*telecommunication information networking architecture*). Los MSC incluyen un componente de prueba (*TC, test component*) y un sistema sometido a prueba (*SUT, system under test*). La prueba se efectúa en tres pasos: obtención de un acceso nombrado a un servidor de sesión de acceso, fijación de un contexto de usuario para un usuario, y puesta en marcha de un servicio seleccionado para este usuario. Estas pasos se definen como los MSC de utilidad.

Los MSC definidores abarcan la especificación de comportamiento, incluidas las constricciones de tiempo (MSC *Black_Box_Behavior*), la especificación de medición (MSC *Blackballed_Measurement_Scenario*), y los trazos de ejecuciones de pruebas (MSC *Black_Box_Test_Execution*).

En las mediciones efectuadas por el componente de prueba se utilizan las variables de tiempo *rel1, ..., abs2,* que se declaran como variables dinámicas de tipo Time del ejemplar *TC*.

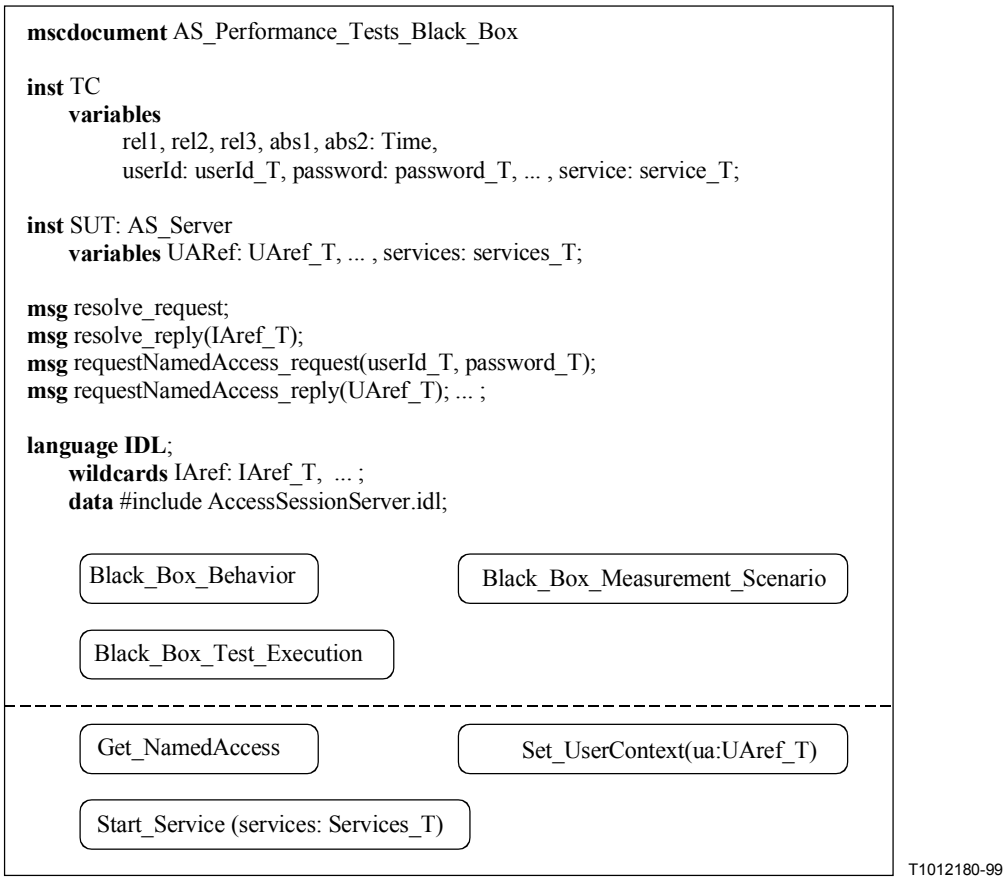


Figura 41/Z.120 – Documento MSC y declaración de variables de tiempo

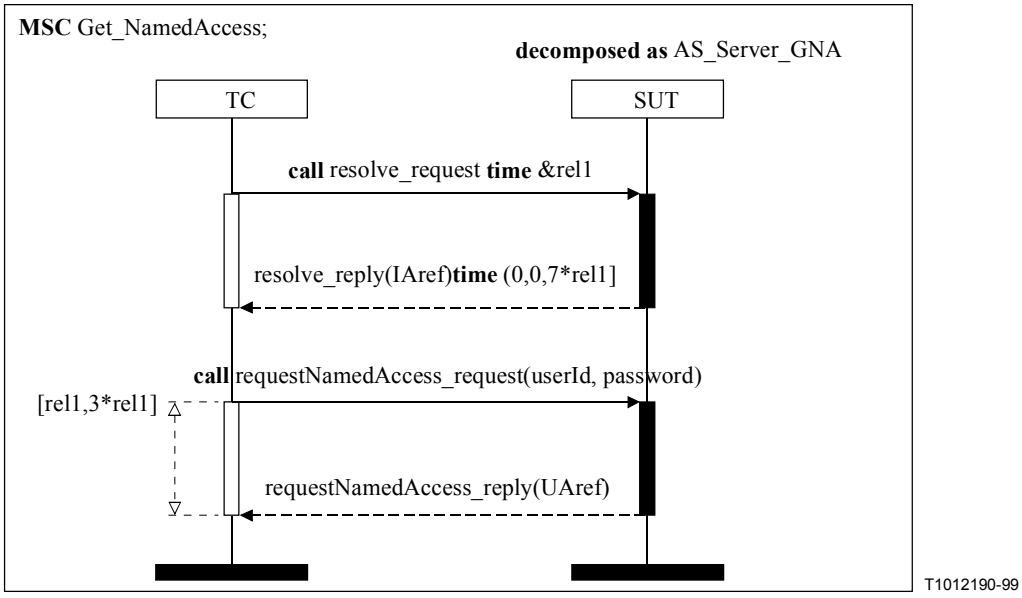


Figura 42/Z.120 – Constricciones de tiempo y mediciones

El MSC de utilidad representado en la figura 42 emplea una medición de tiempo relativo para observar la duración de mensaje de la llamada resolve_request. La variable de tiempo *rel1* contendrá el tiempo transcurrido desde que la llamada sale de *TC* hasta que entra en *SUT*. La variable *rel1* es vinculada cuando se produce la entrada de la llamada.

La medición de la duración de la llamada se utiliza ulteriormente para restringir la duración de mensaje para *resolve_reply*. La constricción de tiempo relativo *(0,0,7*rel1]* permite que el mensaje tome, como máximo, el 70 por ciento del tiempo que tomó para emitir la llamada resolve_request de *TC* a *SUT*.

Además, la medición de la duración de la llamada se utiliza para constreñir la ejecución del ejemplar *TC:* la constricción de tiempo relativo *(rel1,3*rel1]* requiere que después de la salida de la llamada requestNamedAccess se necesite *rel1* como mínimo y *3*3rel1* como máximo para obtener la respuesta.

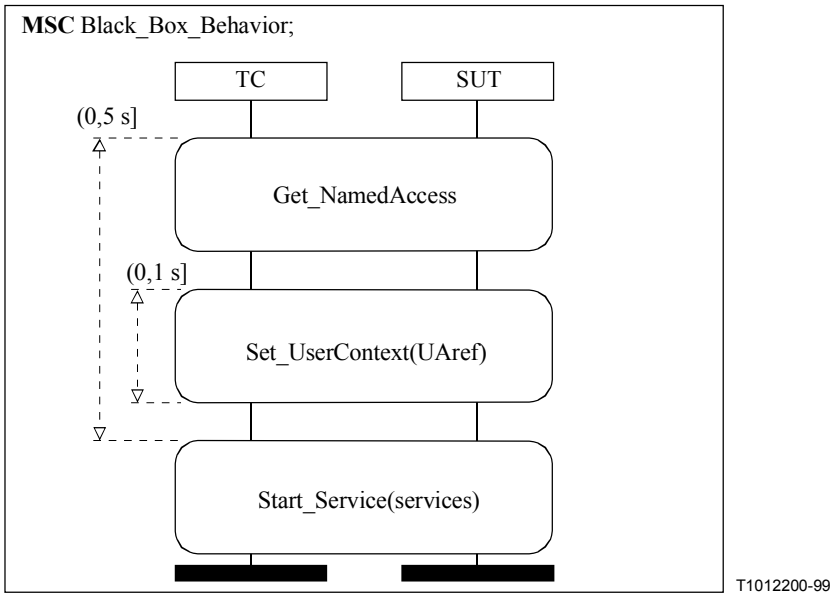


Figura 43/Z.120 – Constricciones de tiempo para referencias de MSC

La figura 43 presenta un ejemplo de la utilización de constricciones de tiempo para referencias de MSC: la constricción de tiempo relativo *(0,1 s]* especifica que se requiere como máximo *1 s* para obtener la ejecución del MSC *Set_UserContext*, es decir, desde el evento comienzo hasta el evento final de *Set_UserContext* transcurrirá *1 s* como máximo.

La constricción de tiempo relativo *(0,5 s]* relaciona el comienzo de *Get_NamedAccess* con el comienzo de *Start_Service* y especifica que se requieren como máximo *5 s* para poder poner en marcha un servicio desde el punto de tiempo en que se ha iniciado el procedimiento para obtener acceso nombrado. Esta constricción influye también en la ejecución de *Set_UserContext*.

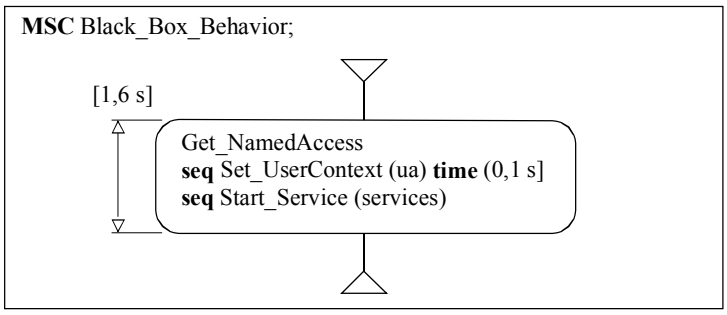


Figura 44/Z.120 – Constricciones de tiempo para HMSC

La figura 44 muestra las constricciones de tiempo para HMSC. Es semánticamente igual a la figura 43 de la que se diferencia en que, en esta figura, la constricción de tiempo relativo *[1,6 s]* se aplica desde el comienzo de *Get_NamedAccess* hasta el final de *Start_Service*, esto es, para la puesta en marcha de un servicio se requiere como mínimo *1 s* y como máximo *6 s*.

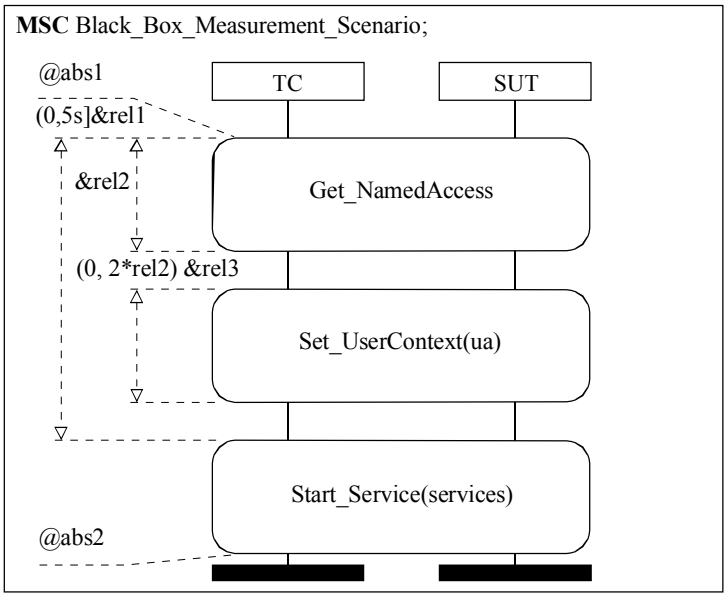


Figura 45/Z.120 – Observación de tiempo por mediciones

El MSC *Black_Box_Measurement_Scenario* en la figura 45 utiliza mediciones absolutas (indicadas por @) y mediciones relativas (indicadas por &). Las mediciones absolutas observan los valores del reloj global.

Las mediciones absolutas *@abs1* y *@abs2* dan el comienzo y el final absolutos de la prueba descrita como una secuencia de *Get_NamedAccess*, *Set_UserContext* y *Start_Service*. Además, se utilizan mediciones relativas para medir la distancia de tiempo entre parejas de eventos.

Por ejemplo, *&rel1* mide la distancia entre el evento comienzo de *Get_NamedAccess* y el evento comienzo de *Start_Service*. Esta medición se combina con una constricción *(0,5 s]* lo que significa que el lapso entre el comienzo de *Get_NamedAccess* y el comienzo de *Start_Service* está constreñido y el tiempo que transcurre realmente (dentro de los límites de la constricción) se observa por medio de la medición relativa.

*0, 2*rel2) &rel3* es también una constricción de tiempo relativo combinada con una medición relativa. Esta constricción hace referencia a una medición del lapso entre el comienzo de *Get_NamedAccess* y su final, efectuada antes.

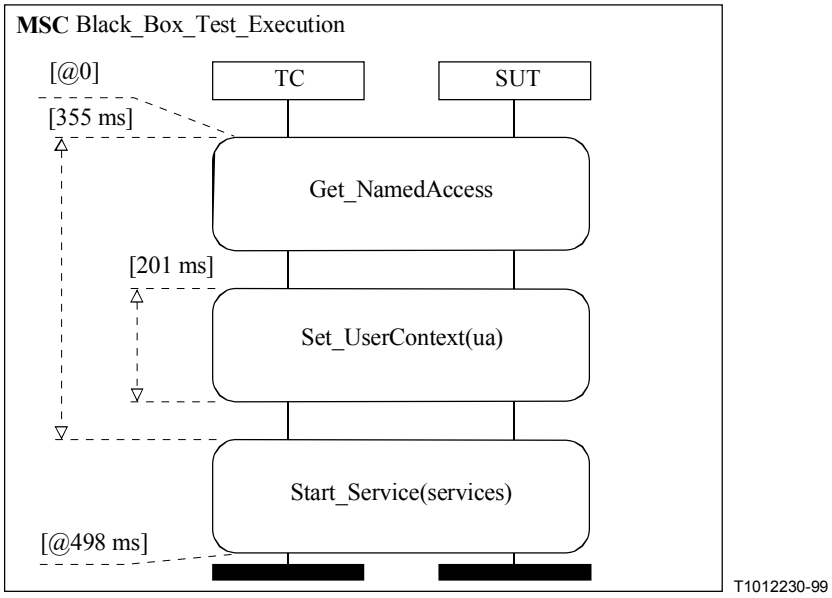


Figura 46/Z.120 – Tiempo singular utilizado para trazos de sistema

La figura 46 representa trazos de sistema (por ejemplo, tomados de ejecuciones, simulaciones o pruebas de sistema) utilizando puntos de tiempo absolutos y relativos. Los puntos de tiempo absoluto se indican por @. *[@0]* significa que *Get_NamedAccess* fue comenzada en el tiempo 0 del reloj global) (es decir, en el instante en que el reloj global fue reiniciado). La ejecución de la secuencia *Get_NamedAccess*, *Set_UserContext* y *Start_Service* tomó *498 ms*. La ejecución de *Set_UserContext* tomó *201 ms*. El evento comienzo de *Start_Service* se produjo *355 ms* después del evento comienzo de *Get_NamedAccess*.

12 Creación y terminación de procesos

Este ejemplo ilustra la creación dinámica del ejemplar 'subscriber' debida a una petición de conexión y su correspondiente terminación debida a una petición de desconexión. Véase la figura 47.

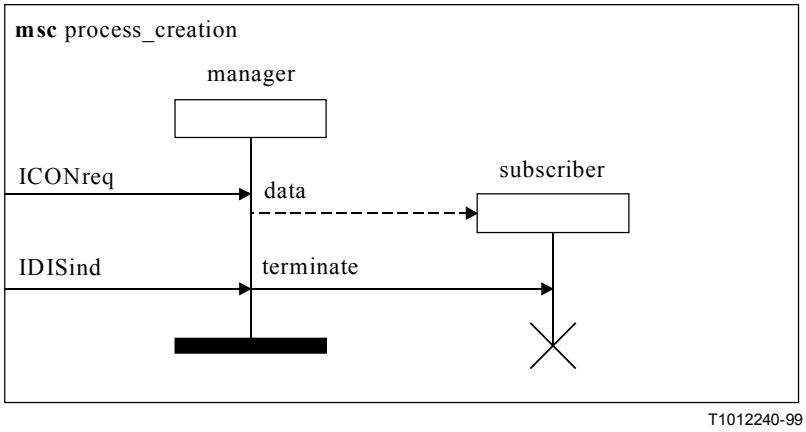


Figura 47/Z.120 – MSC process_creation

```
msc process_creation;
inst manager;
inst subscriber;
gate out ICONreq to manager;
gate out IDISind to manager;

        manager: instance;
            in ICONreq from env;
            create subscriber(data);
            in IDISind from env;
            out terminate to subscriber;
        endinstance;
        subscriber: instance;
            in terminate from manager;
            stop;
        endinstance;
endmsc;
```

13 Corregión

Este ejemplo muestra una región concurrente, que indicará que el consumo de 'process_data1' y el consumo de 'process_data2' no están ordenados en el tiempo, es decir, 'process_data1' puede ser consumido antes o después de 'process_data2'. Véase la figura 48.

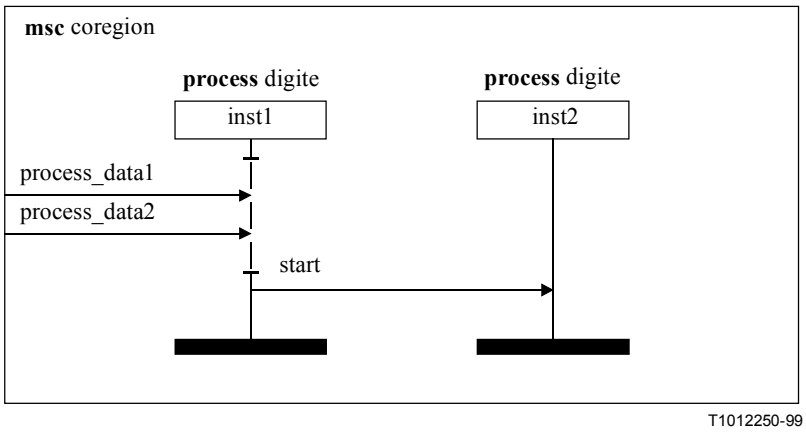


Figura 48/Z.120 – MSC corregión

```
msc coregion;
inst inst1;
inst inst2;
gate out process_data1 to inst1;
gate out process_data2 to inst1;

        inst1: instance process digite;
            concurrent;
                in process_data1 from env;
                in process_data2 from env;
            endconcurrent;
            out start to inst2;
        endinstance;
        inst2: instance process digite;
            in start from inst1;
        endinstance;
endmsc;
```

14 Ordenación general

14.1 Ordenación generalizada dentro de una corregión

Este ejemplo muestra una ordenación generalizada dentro de una corregión mediante 'connections', es decir se describe la ordenación mediante las conexiones que relacionan los eventos dentro de la corregión. Dentro del MSC 'connectivity', se define la siguiente ordenación: ICONreq <ICONind <ICONresp, ICONreq < IDISind.

Se muestra la situación en la que IDISind no está ordenada respecto de ICONind ni de ICONresp. Véase la figura 49.

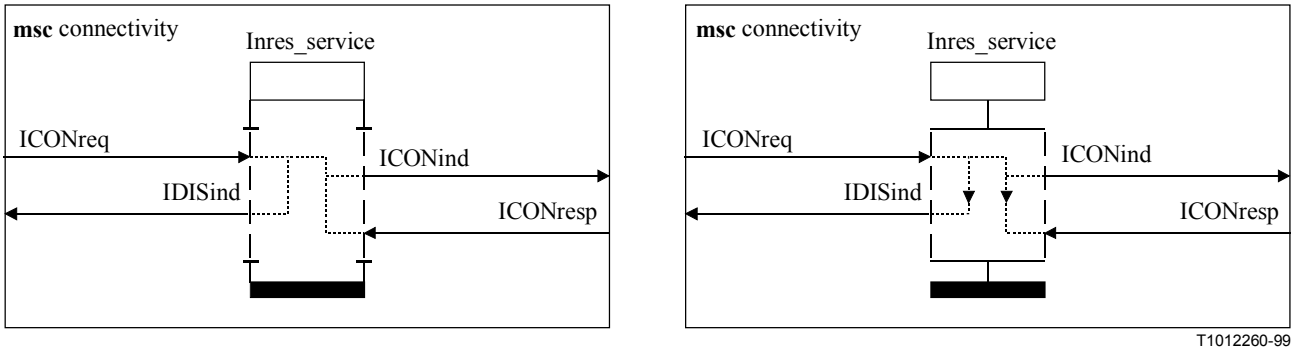


Figura 49/Z.120 – MSC conectividad en dos variantes gráficas

```
msc connectivity;
inst Inres_service;
gate out ICONreq to Inres_service;
gate in ICONind from Inres_service;
gate in IDISind from Inres_service;
gate out ICONresp to Inres_service;

        Inres_service: instance;
            concurrent;
                in ICONreq from env before Label1, Label2;
                label Label1; out ICONind to env before Label3;
                label Label2; out IDISind to env;
                label Label3; in ICONresp from env;
            endconcurrent;
        endinstance;
endmsc;
```

14.2 Ordenación generalizada entre ejemplares diferentes

En este ejemplo se muestra la utilización de constructivos de sincronización tomados de los diagramas de secuencia en el tiempo para describir una ordenación generalizada entre ejemplares diferentes. La línea (doblada o inclinada hacia abajo) entre la entrada de mensaje 'ICONreq' y la salida de mensaje 'ICONind', designa la ordenación ICONreq < ICONind. Véase la figura 50.

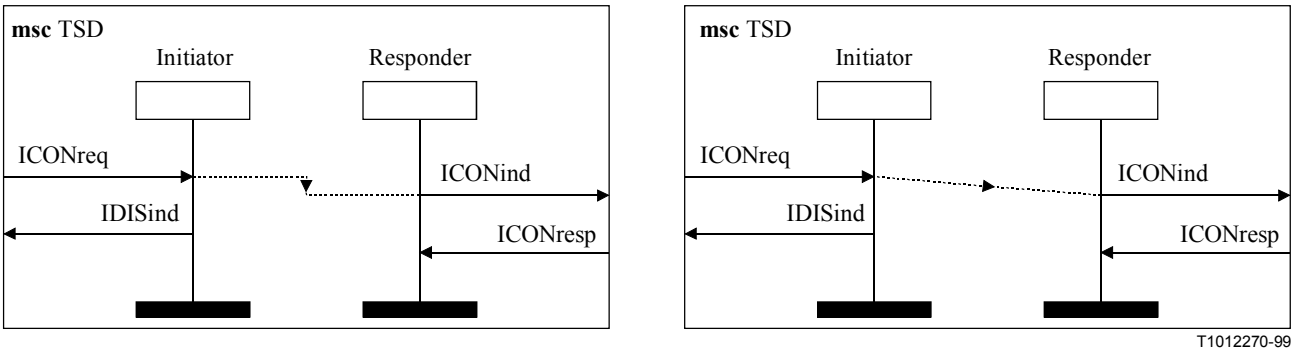


Figura 50/Z.120 – MSC diagrama de secuencia de tiempo en dos variantes gráficas

```
msc TSD;
inst Initiator;
inst Responder;
gate out ICONreq to Initiator;
gate in ICONind from Responder;
gate in IDISind from Initiator;
gate out ICONresp to Responder;

        instance Initiator;
            in ICONreq from env before Label1;
            out IDISind to env;
        endinstance;
        instance Responder;
            label Label1; out ICONind to env;
            in ICONresp from env;
        endinstance;
endmsc;
```

15 Expresiones en línea

15.1 Expresión en línea con composición alternativa

En este ejemplo se combina el caso de conexión exitosa con el caso de fallo dentro de un MSC mediante la expresión en línea alternativa (MSC 'alternative'). Dentro del MSC 'exception' se describe la misma situación mediante una expresión en línea de excepción equivalente. Véase la figura 51.

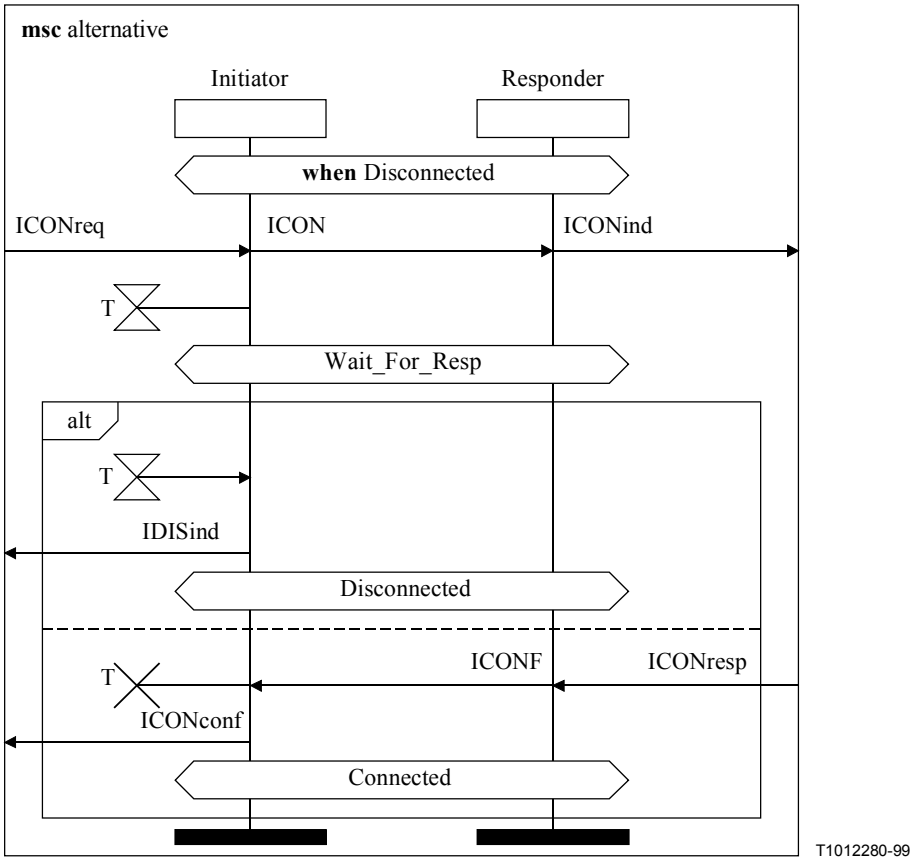


Figura 51/Z.120 – Expresión en línea alternativa

```
msc alternative;
inst Initiator;
inst Responder;
gate out ICONreq to Initiator;
gate in ICONind from Responder;
gate in IDISind from Initiator;
gate out ICONresp to Responder;
gate in ICONconf from Initiator;

        Initiator:      instance;
        Responder:      instance;
        all:            condition when Disconnected;
        Initiator:      in ICONreq from env;
                        out ICON to Responder;
                        starttimer T;
        Responder:      in ICON from Initiator;
                        out ICONind to env;
        all:            condition Wait_for_Resp;
                        alt begin;
            Initiator:          timeout T;
                                out IDISind to env;
            all:                condition Disconnected;
                        alt;
            Responder:          in ICONresp from env;
                                out ICONF to Initiator;
            Initiator:          in ICONF from Responder;
                                stoptimer T;
                                out ICONconf to env;
            all:                condition Connected;
                        alt end;
        Initiator:      endinstance;
        Responder:      endinstance;
endmsc;
```

El operador **exc** es igual que una alternativa en la que el segundo operando es todo el resto del MSC, como se muestra en el siguiente ejemplo: (MSC 'alternative' significa exactamente lo mismo que MSC 'exception'). Véase la figura 52.

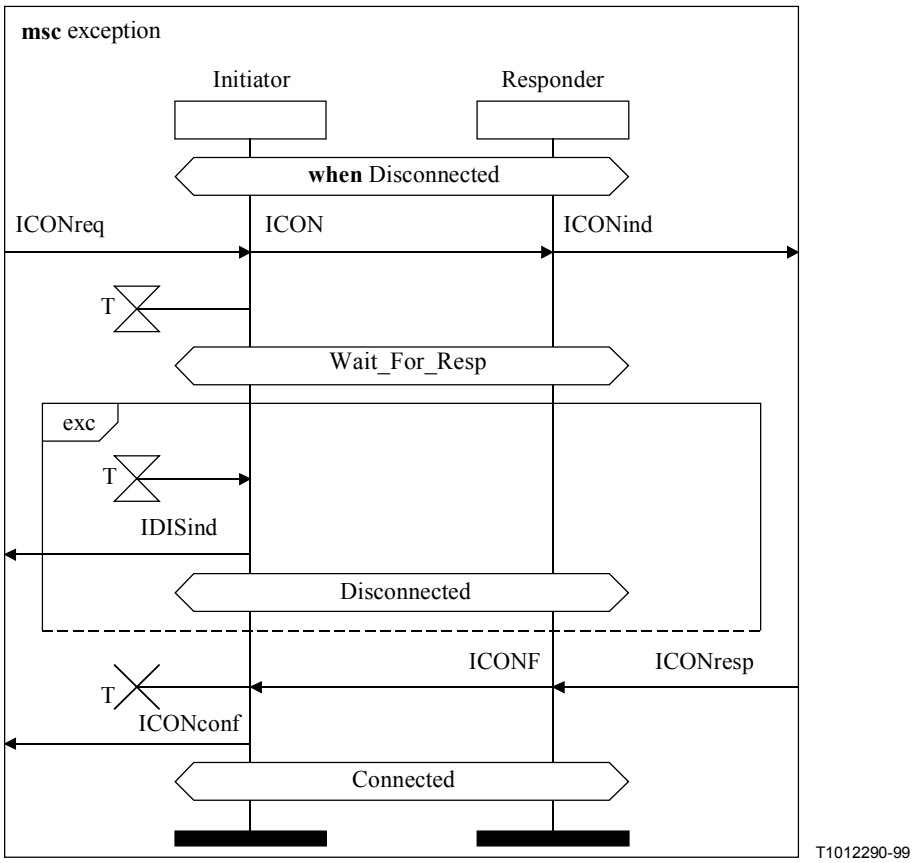


Figura 52/Z.120 – Expresión en línea de excepción

```
msc exception;
inst Initiator;
inst Responder;
gate out ICONreq to Initiator;
gate in ICONind from Responder;
gate in IDISind from Initiator;
gate out ICONresp to Responder;
gate in ICONconf from Initiator;

        Initiator:      instance;
        Responder:      instance;
        all:            condition when Disconnected;
        Initiator:      in ICONreq from env;
                        out ICON to Responder;
                        starttimer T;
        Responder:      in ICON from Initiator;
                        out ICONind to env;
        all:            condition Wait_for_Resp;
                        exc begin;
            Initiator:      timeout T;
                            out IDISind to env;
            all:            condition Disconnected;
                        exc end;
        Responder:      in ICONresp from env;
                        out ICONF to Initiator;
        Initiator:      in ICONF from Responder;
                        stoptimer T;
                        out ICONconf to env;
```

```
        all:            condition Connected;
        Initiator:      endinstance;
        Responder:      endinstance;
endmsc;
```

15.2 Expresión en línea con puertas

En este ejemplo se describe el escenario de 15.1 de una forma ligeramente modificada: el mensaje 'ICONF' procedente de 'Responder' se conecta a través de puertas con la expresión en línea alternativa ligada a 'Initiator'. El mensaje 'ICONF' procedente de 'Responder' se transfiere a través de la puerta g1 (en ambas alternativas) a 'Initiator'. Se describe la situación en la que 'Initiator' espera una respuesta de 'Responder'. En MSC 'very_advanced' se combinan dos casos:

a) el 'Responder' no contesta a tiempo: la entrada de mensaje 'ICONF' se descarta después de transcurrido el periodo de temporización y efectuada la desconexión de 'Initiator',

b) el 'Responder' contesta a tiempo, lo que conduce a una conexión exitosa.

Véase la figura 53.

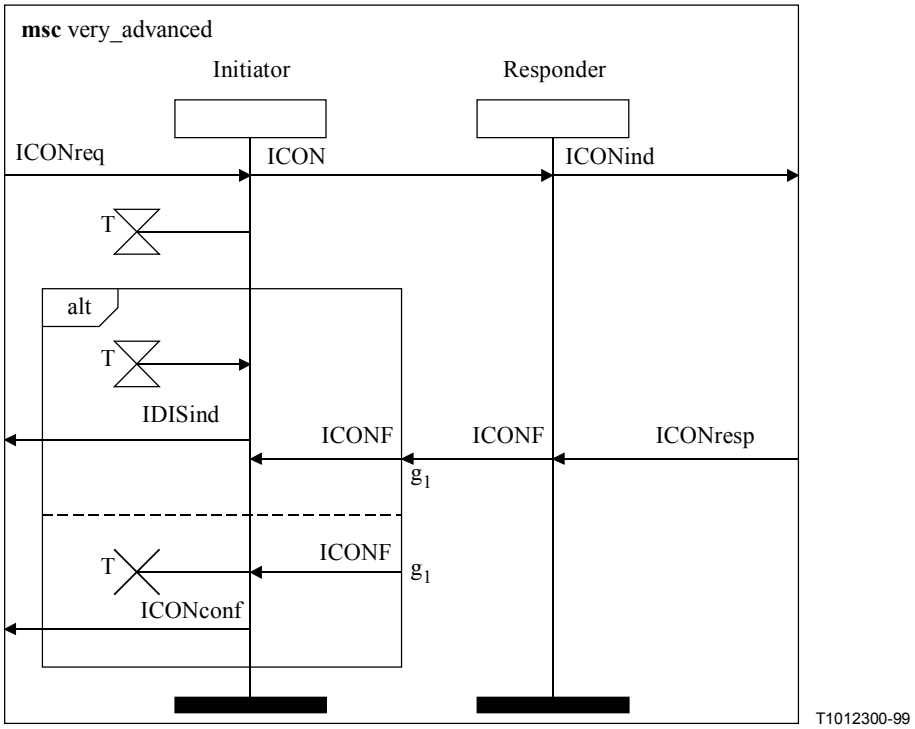


Figura 53/Z.120 – Expresión en línea con puertas

```
msc very_advanced;
inst Initiator;
inst Responder;

gate out ICONreq to Initiator;
gate in ICONind from Responder;
gate in IDISind from Initiator;
gate out ICONresp to Responder;
gate in ICONconf from Initiator;
        Initiator:      instance;
        Responder:      instance;
        Initiator:      in ICONreq from env;
                        out ICON to Responder;
                        starttimer T;
        Responder:      in ICON from Initiator;
                        out ICONind to env;
```

```
                    in ICONresp from env;
                    out ICONF to inline altref via g1;
        Initiator:  alt begin altref;
                        gate g1 out ICONF to Initiator
                        external in ICONF from Responder;
                        timeout T;
                        out IDISind to env;
                        in ICONF from env via g1;
                    alt;
                        gate g1 out ICONF to Initiator;
                        in ICONF from env via g1;
                        stoptimer T;
                        out ICONconf to env;
                    alt end;
        Initiator:  endinstance;
        Responder:  endinstance;
endmsc;
```

15.3 Expresión en línea con composición paralela

En este ejemplo se muestra cómo una expresión en línea paralela describe el entrelazado de una petición de conexión iniciada por 'Station_Res', es decir 'MDAT(CR)' seguida de 'ICONind', con una petición de desconexión iniciada por el entorno, esto es, 'IDISreq' seguida de 'MDAT(DR)'. Véase la figura 54.

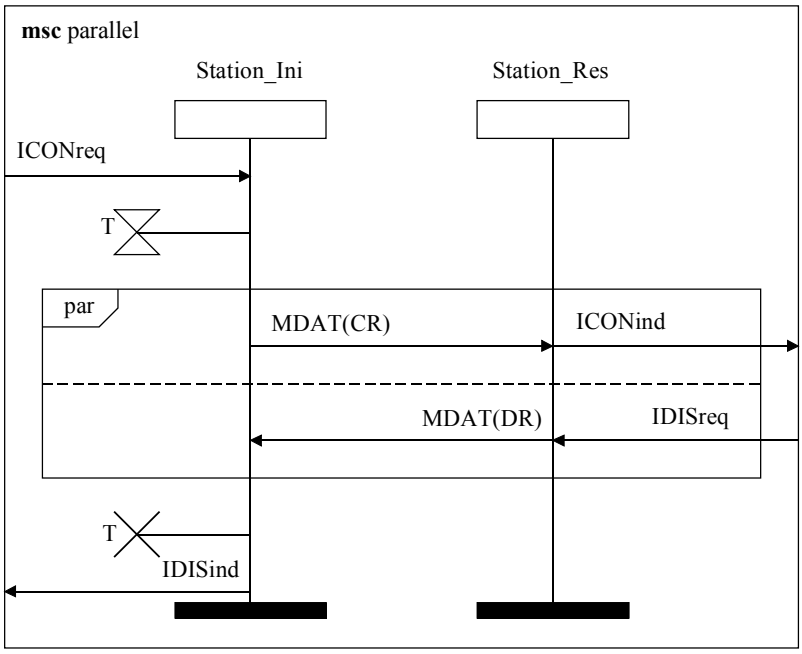


Figura 54/Z.120 – Expresión en línea paralela

```
msc parallel;
inst Station_Ini;
inst Station_Res;
gate out ICONreq to Station_Ini;
gate in ICONind from Station_Res;
gate out IDISreq to Station_Res;
gate in IDISind from Station_Ini;

        Station_Ini: instance;
                    in ICONreq from env;
                    starttimer T;
        Station_Res: instance;
        all:        par begin;
```

```
                Station_Ini:        out MDAT(CR) to Station_Res;
                Station_Res:  in MDAT(CR) from Station_Ini;
                                    out ICONind to env;
                              par;
                Station_Res:  in IDISreq from env;
                                    out MDAT(DR) to Station_Ini;
                Station_Ini:        in MDAT(DR) from Station_Res;
                              par end;
          Station_Res:        endinstance;
          Station_Ini: stoptimer T;
                              out IDISind to env;
                              endinstance;
endmsc;
```

16 Referencias de MSC

16.1 Referencia de MSC

Se utilizan en este ejemplo las referencias de MSC 'data_transmission' para designar dos transmisiones de datos exitosas que hacen referencia a la misma definición de MSC.

El MSC *ref* tiene puertas ambiguas porque tiene nombres implícitos, pero esto es correcto siempre que *ref* no se utilice como referencia de MSC en otro diagrama. Véanse las figuras 55 y 56.

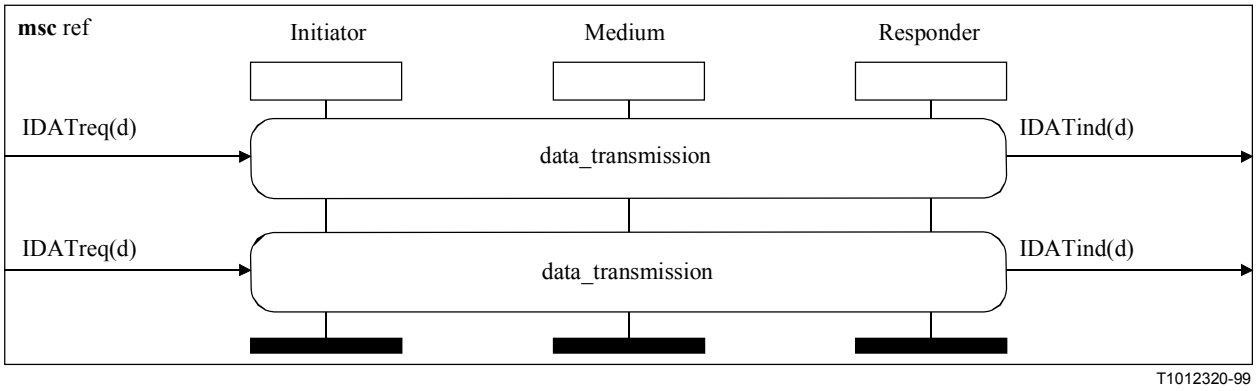


Figura 55/Z.120 – MSC que incluye referencias a otros MSC

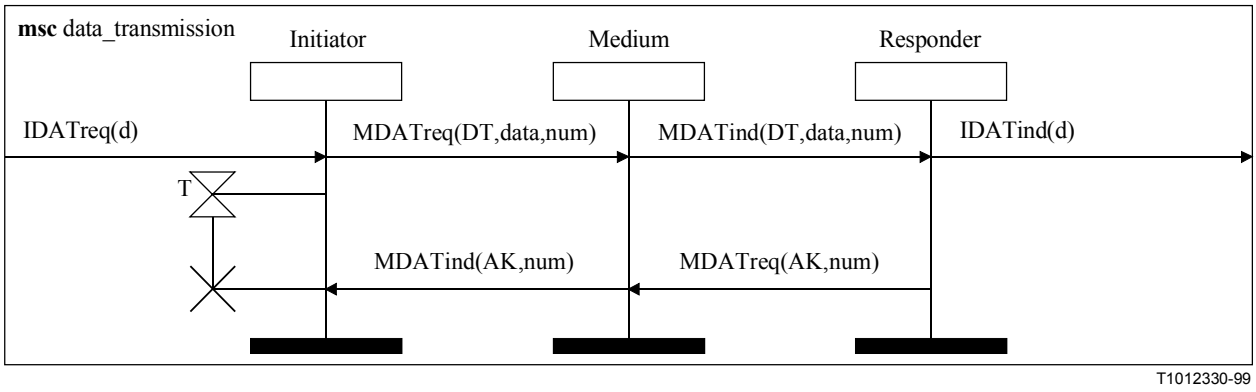


Figura 56/Z.120 – Un MSC al que se hace referencia

```
msc ref;
inst Initiator;
inst Medium;
inst Responder;
gate out L1 to reference data_trans1;
gate in L2 from reference data_trans1;
gate out L3 to reference data_trans2;
gate in L4 from reference data_trans2;

        Initiator:      instance;
        Medium:         instance;
        Responder:      instance;
        all:            reference data_trans1:data_transmission;
                        gate in IDATreq, L1 from env;
                        gate out IDATind, L2 to env
                        reference data_trans2:data_transmission;
                        gate in IDATreq, L3 from env
                        gate out IDATind, L4 to env;
        Initiator:      endinstance;
        Medium:         endinstance;
        Responder:      endinstance;
endmsc;

msc data_transmission;
inst Initiator;
inst Medium;
inst Responder;
gate out IDATreq(d) to Initiator;
gate in IDATind(d) from Responder;

        Initiator:      instance;
        Medium:         instance;
        Responder:      instance;
        Initiator:      in IDATreq(d) from env;
                        out MDATreq(DT, data, num) to Medium;
                        starttimer T;
        Medium:         in MDATreq(DT, data, num) from Initiator;
                        out MDATind(DT, data, num) to Responder;
        Responder:      in MDATind(DT, data, num) from Medium;
                        out IDATind(d) to env;
                        out MDATreq(AK, num) to Medium;
        Medium:         in MDATreq(AK, num) from Responder;
                        out MDATind(AK, num) to Initiator;
        Initiator:      in MDATind(AK, num) from Medium;
                        stoptimer T;
        Initiator:      endinstance;
        Medium          endinstance;
        Responder:      endinstance;
endmsc;
```

16.2 Referencia de MSC con puerta

En este ejemplo se representa una referencia de MSC conectada con el exterior a través de una puerta. En el ejemplo de la figura 61 se presentan los MSC referenciados 'message_lost' y 'time_out'.

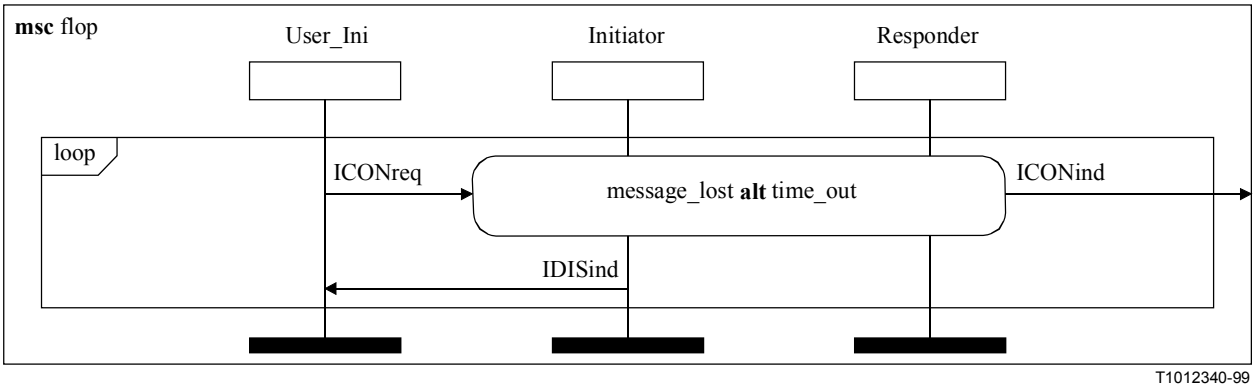


Figura 57/Z.120 – Referenciación de un MSC con puertas de mensaje

```
msc flop;
inst User_Ini;
inst Initiator;
inst Responder;
gate in ICONind from failed;

        User_Ini:       instance;
        Initiator:      instance;
        Responder:      instance;
        all:            loop begin;
            User_Ini:       out ICONreq to reference failed;
            Initiator,
            Responder:      reference failed: message_lost alt time_out;
                            gate in ICONreq from User_Ini;
                            gate out ICONind to env;


            Initiator:      out IDISind to User_Ini;
            User_Ini:       in IDISind from Initiator;
                        loop end;
        User_Ini:       endinstance;
        Initiator:      endinstance;
        Responder:      endinstance;
endmsc;
```

17 MSC de alto nivel (HMSC)

17.1 MSC de alto nivel con bucle libre

El MSC 'setup_attach' de este ejemplo muestra el modelado del establecimiento de la conexión mediante un bucle libre. Véase la figura 58.

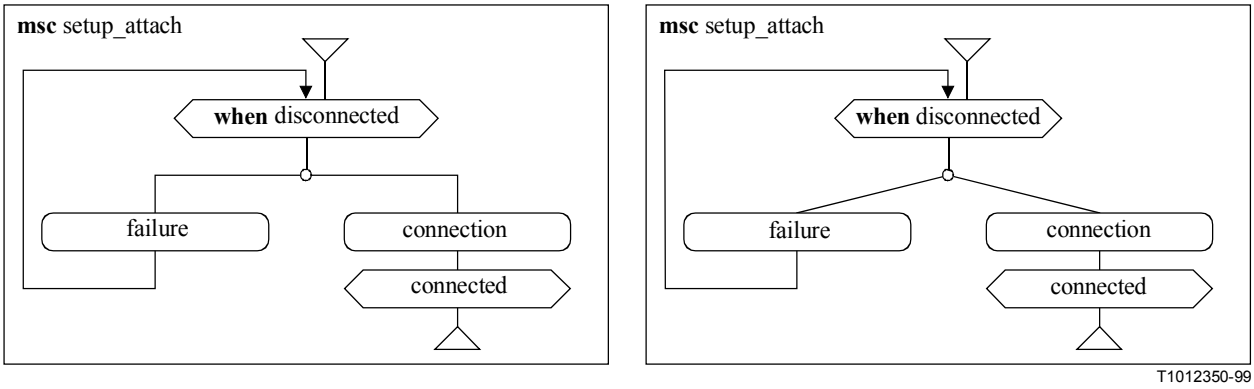


Figura 58/Z.120 – HMSC setup_attach (dos variantes gráficas diferentes)

```
msc setup_attach;
expr L1;
        L1: condition when disconnected seq ( L2 );
        L2: connect seq ( L3 alt L4 );
        L3: (failure); seq ( L1 );
        L4: (connection); seq ( L5 );
        L5: condition connected seq ( L6 );
        L6: end;
endmsc;
```

17.2 MSC de alto nivel con bucle

El MSC 'setup_loop' de este ejemplo muestra el modelado del establecimiento de la conexión mediante un bucle ligado a la referencia de MSC 'failure'. Véase la figura 59.

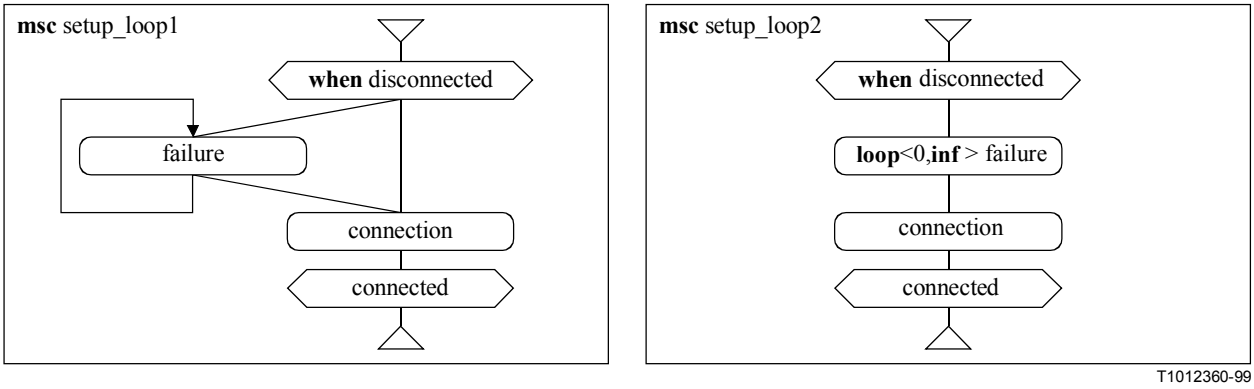


Figura 59/Z.120 – setup_loop en dos variantes diferentes

```
msc setup_loop1;
expr L1;
        L1: condition when disconnected seq (L2 alt L3);
        L2: (failure) seq (L2 alt L3);
        L3: (connection) seq (L4);
        L4: condition connected seq (L5);
        L5: end;
endmsc;

msc setup_loop2;
expr L1;
        L1: condition when disconnected seq (L2);
        L2: (loop <0,inf> failure) seq (L3);
        L3: (connection) seq (L4);
        L4: condition connected seq (L5);
        L5: end;
endmsc;
```

17.3 MSC de alto nivel con composición alternativa

El MSC 'alternative' de este ejemplo muestra un constructivo alternativo con parentización de corrección en el que las ramas tienen un constructivo de unión correspondiente.

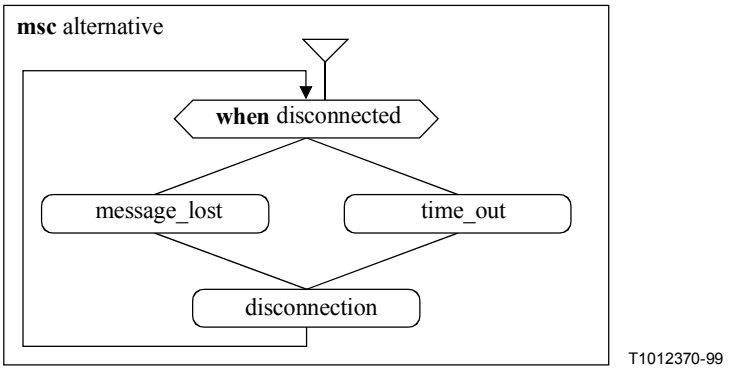


Figura 60/Z.120 – HMSC con expresión alternativa

```
msc alternative;
expr L1;
        L1: condition disconnected seq ( L2 alt L3 );
        L2: (message_lost); seq ( L4 );
        L3: (time_out); seq ( L4 );
        L4: (disconnection); seq ( L1 );
endmsc;
```

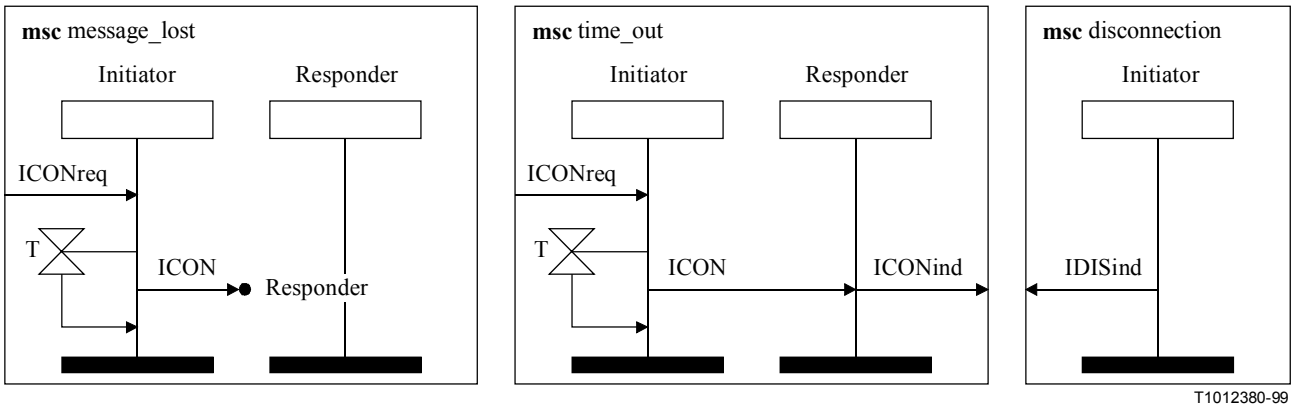


Figura 61/Z.120 – MSC simples referenciados a partir del HMSC de la figura 60

```
msc message_lost;
inst Initiator;
inst Responder;
gate out ICONreq to Initiator;

        Initiator: instance;
            in ICONreq from env;
            starttimer T;
            out ICON to lost Responder;
            timeout T;
        endinstance;
        Responder: instance;
            in ICON from Initiator;
        endinstance;
endmsc;
```

```
msc time_out;
inst Initiator;
inst Responder;
gate out ICONreq to Initiator;
gate in ICONind from Responder;

        Initiator: instance;
            in ICONreq from env;
            starttimer T;
            out ICON to Responder;
            timeout T;
        endinstance;
        Responder: instance;
            in ICON from Initiator;
            out ICONind to env;
        endinstance;
endmsc;

msc disconnection;
inst Initiator;
gate in IDISind from Initiator;

        Initiator: instance;
            out IDISind to env;
        endinstance;
endmsc;
```

17.4 MSC de alto nivel con composición paralela

En este ejemplo, la petición de conexión procedente del 'Initiator' se combina en paralelo con la petición de desconexión procedente del 'Responder' mediante el HMSC con expresión paralela.

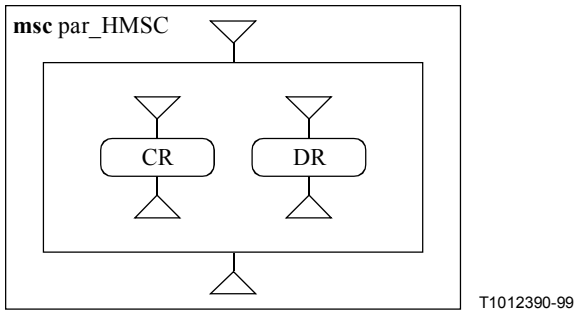


Figura 62/Z.120 – HMSC con expresión paralela

```
msc par_HMSC;
expr L1;
        L1:  expr L2;
                 L2: (CR); seq ( L3 );
                 L3: end;
             endexpr
        par
             expr L4;
                 L4: (DR); seq ( L5 );
                 L5: end;
             endexpr
        seq ( L6 );
        L6: end;
endmsc;
```

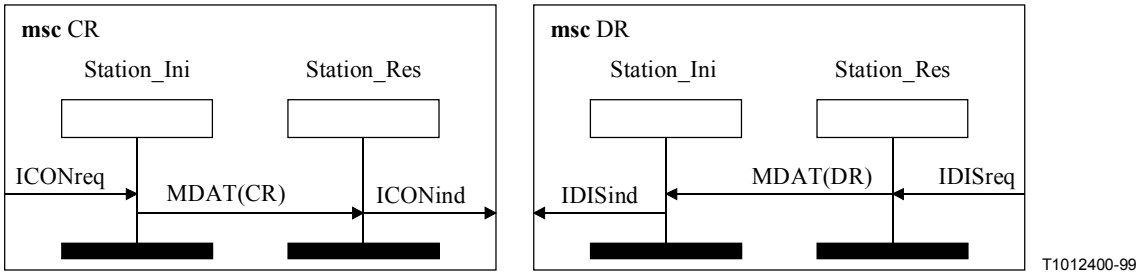


Figura 63/Z.120 – MSC referenciados a partir del HMSC de la figura 62

```
msc CR;
inst Station_Ini;
inst Station_Res;
gate out ICONreq to Station_Ini;
gate in ICONind from Station_Res;

        Station_Ini: instance;
                        in ICONreq from env;
                        out MDAT(CR) to Station_Res;
                        endinstance;
        Station_Res:    instance;
                        in MDAT(CR) from Station_Ini;
                        out ICONind to env;
                        endinstance;
endmsc;

msc DR;
inst Station_Ini;
inst Station_Res;
gate out IDISreq to Station_Res;
gate in IDISind from Station_Ini;

        Station_Res:    instance;
                        in IDISreq from env;
                        out MDAT(DR) to Station_Ini;
                        endinstance;
        Station_Ini: instance;
                        in MDAT(DR) from Station_Res;
                        out IDISind to env;
                        endinstance;
endmsc;
```

ANEXO A

Índice alfabético

Las entradas son las <palabra clave> y los símbolos no terminales de la *Gramática textual concreta* y la *Gramática gráfica concreta*. Los números de las páginas indicados hacen referencia a la versión inglesa del texto.

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie B	Medios de expresión: definiciones, símbolos, clasificación
Serie C	Estadísticas generales de telecomunicaciones
Serie D	Principios generales de tarificación
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedios
Serie I	Red digital de servicios integrados
Serie J	Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedios
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	RGT y mantenimiento de redes: sistemas de transmisión, circuitos telefónicos, telegrafía, facsímil y circuitos arrendados internacionales
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de transmisión telefónica, instalaciones telefónicas y redes locales
Serie Q	Conmutación y señalización
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	Redes de datos y comunicación entre sistemas abiertos
Serie Y	Infraestructura mundial de la información y aspectos del protocolo Internet
Serie Z	**Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación**

Impreso en Suiza
Ginebra, 2001