UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T Z.121

(02/2003)

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

Vinculación entre los datos del lenguaje de especificación y descripción y los gráficos de secuencias de mensajes

Recomendación UIT-T Z.121

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**
Lenguaje ampliado de definición de objetos	Z.130–Z.139
Notación combinada arborescente y tabular	Z.140–Z.149
Notación de requisitos de usuarios	Z.150–Z.159
LENGUAJES DE PROGRAMACIÓN	
CHILL: el lenguaje de alto nivel 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.349
Interfaces hombre-máquina orientadas a datos	Z.350–Z.359
Interfaces hombre-máquina para la gestión de las redes de telecomunicaciones	Z.360–Z.369
CALIDAD	
Calidad de soportes lógicos de telecomunicaciones	Z.400–Z.409
Aspectos de la calidad de las Recomendaciones relativas a los protocolos	Z.450–Z.459
MÉTODOS	
Métodos para validación y pruebas	Z.500–Z.519
SOPORTE INTERMEDIO	
Entorno del procesamiento distribuido	Z.600–Z.609

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

Recomendación UIT-T Z.121

Vinculación entre los datos del lenguaje de especificación y descripción y los gráficos de secuencias de mensajes

Resumen

En esta Recomendación se describe el lenguaje de especificación y descripción (Z.100) (SDL) para los elementos sintácticos y semánticos de la interfaz de datos de los gráficos de secuencias de mensajes (Z.120) (MSC), se definen los tipos por defecto, y también la sintaxis de las posibles definiciones de datos SDL que se pueden utilizar en un documento MSC.

Orígenes

La Recomendación UIT-T Z.121, preparada por la Comisión de Estudio 17 (2001-2004) del UIT-T, fue aprobada por el procedimiento de la Resolución 1 de la AMNT el 13 de febrero de 2003.

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 Asamblea Mundial de Normalización de las Telecomunicaciones (AMNT), 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 AMNT.

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 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.

© UIT 2003
Reservados todos los derechos. Ninguna parte de esta publicación puede reproducirse por ningún procedimiento sin previa autorización escrita por parte de la UIT.

ÍNDICE

Página

1 Alcance 1

2 Referencias 1

3 Interfaz sintáctica 1
- 3.1 Declaración de lenguaje 1
- 3.2 Declaraciones de paréntesis y de escape 2
- 3.3 Declaración y utilización de datos 2
- 3.4 Tipos de datos por defecto y comodines 3

4 Interfaz semántica 3
- 4.1 Definiciones relativas a la buena formación 3
- 4.2 Funciones de interfaz de la semántica estática 4
- 4.2.1 Tc1, cadenas de definición de datos 4
- 4.2.2 Tc2, cadenas de referencia de tipo 5
- 4.2.3 Tc3, cadenas de expresión 5
- 4.2.4 Tc4, cadenas de expresiones tipificadas 6
- 4.2.5 EqVar, cadenas de variables iguales 6
- 4.3 Funciones de interfaz semántica dinámica 7
- 4.3.1 Vars, extracción de variables 7
- 4.3.2 Replace, sustitución de variable 7
- 4.3.3 NewVar, nueva variable 7
- 4.3.4 Eval, evaluación de expresiones 8

5 Ejemplo 9
- 5.1 Utilización de la interfaz SDL por defecto 9

Introducción

La vinculación entre los datos del SDL y los MSC se describe en dos partes. En la primera parte se describe la sintaxis de la interfaz, la cual define las instrucciones de documentos MSC relativas a la utilización del SDL como el lenguaje de datos, y en la segunda se describe la semántica de la interfaz. Esta última consiste en la definición de una serie de funciones que se utilizan para evaluar la sintaxis, la semántica estática y la semántica dinámica de los datos SDL utilizados en MSC.

Recomendación UIT-T Z.121

Vinculación entre los datos del lenguaje de especificación y descripción y los gráficos de secuencias de mensajes

1 Alcance

En esta Recomendación se describe el lenguaje de especificación y descripción (Z.100) para la interfaz de datos de los gráficos de secuencias de mensajes (Z.120) que es definido como el lenguaje por defecto para la Z.120.

La interfaz de datos definida en la Rec. UIT-T Z.120 para MSC es abierta y permite que los diferentes usuarios del lenguaje utilicen lenguajes de datos diferentes en los diagramas y documentos MSC. En la Rec. UIT-T Z.121 se vincula esa interfaz abierta a una parte del lenguaje de datos del SDL, que será el lenguaje de datos por defecto. Es decir, si en un documento MSC no hay una definición explícita de la interfaz de datos, se supondrá que se aplica la vinculación definida en esta Recomendación, ya que es el lenguaje de cadenas de definición de datos válidas.

Ésta es la primera versión de la Rec. UIT-T Z.121, y se basa en la definición de interfaz de datos que apareció por primera vez en la Rec. UIT-T Z.120 vigente, publicada en 1999.

2 Referencias

Las siguientes Recomendaciones del UIT-T y otras referencias contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y otras referencias son objeto de revisiones por lo que se preconiza que los usuarios de esta Recomendación investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y otras referencias citadas a continuación. Se publica periódicamente una lista de las Recomendaciones UIT-T actualmente vigentes. En esta Recomendación, la referencia a un documento, en tanto que autónomo, no le otorga el rango de una Recomendación.

– Recomendación UIT-T Z.100 (2002), *Lenguaje de especificación y descripción*.

– Recomendación UIT-T Z.120 (1999), *Gráficos de secuencias de mensajes*.

3 Interfaz sintáctica

En los documentos MSC se puede declarar, mediante un enunciado, el nombre del lenguaje de datos utilizado, de manera que las aplicaciones que soportan más de un lenguaje puedan distinguirlos. En otros enunciados se declaran las secuencias de paréntesis utilizadas en el lenguaje de datos, para que las aplicaciones puedan determinar correctamente el alcance de las cadenas de datos incluidas en un MSC. En esta cláusula se describen los enunciados por defecto para la vinculación con los datos del SDL.

El lenguaje se declara en una definición de datos que también contiene las declaraciones de datos deseados: tipos de datos (con métodos y operadores), sintipos y sinónimos.

3.1 Declaración de lenguaje

En el documento MSC se puede declarar el lenguaje, determinándose así qué lenguaje de datos se utiliza en todos MSC incluidos en el documento MSC global. Para el SDL, la declaración del lenguaje es la siguiente (es opcional incluirla porque se trata de la opción por defecto):

language SDL;

Si se hace esta declaración de lenguaje, las declaraciones de paréntesis y de escape quedan implícitamente declaradas, y por consiguiente no es necesario declararlas explícitamente en cada documento MSC, ni tampoco los tipos por defecto o sus correspondientes comodines.

3.2 Declaraciones de paréntesis y de escape

Cuando se utilizan los datos SDL en los MSC, según la declaración de lenguaje anterior, se supondrán las declaraciones de paréntesis y de escape siguientes:

```
parenthesis
    nestable '()', '[]', '()', '{}';
    nonnestable '/*'*/', '/#'#/', '<<'>>';
    equalpar /'/;
    escape '?';
;
```

Esta declaración significa que:

- en las cadenas de datos se aceptan los <comment> y <note> Z.100 (/# un comentario #/ y /* una nota */);
- se puede utilizar los calificadores Z.100 (por ejemplo, <<package BasicTypes>>) para resolver conflictos de nombre;
- se pueden utilizar cadenas de caracteres Z.100 ('hello world') en cadenas de expresiones, para lo cual las comillas dobles (") se utilizarán de la misma manera que en la Z.100, de modo que se puedan emplear las comillas sencillas en las cadenas;
- se utiliza el signo '?' como carácter de escape general para los símbolos de terminación MSC en las cadenas de datos.

Cuando en la cadena de datos se desea *utilizar libremente* un símbolo MSC que puede indicar el final de una cadena de datos, por ejemplo un punto y coma, se deberá utilizar junto con el carácter escape '?'. Por *utilizar libremente* se entiende que no está "oculto" dentro de un par de paréntesis declarados SDL. Por ejemplo no es necesario utilizar el carácter de escape para un punto y coma que está dentro de una cadena SDL, porque está oculto entre los caracteres que delimitan la cadena.

El único símbolo SDL que necesita el carácter de escape en un contexto SDL es la comilla sencilla dentro de una expresión de cadena. Las cadenas SDL no requieren ningún tratamiento especial cuando se utiliza dentro de una cadena de datos MSC.

Los analizadores sintácticos MSC comprueban la correspondencia de paréntesis. Por ejemplo: '(.' y '(' son paréntesis que pueden ir anidados, por lo que el analizador sintáctico detectará si estos paréntesis están anidados de manera incorrecta.

3.3 Declaración y utilización de datos

Las declaraciones de tipo de datos en el lenguaje de datos nativo, se hacen efectivamente en la cadena abierta <data definition string> de un documento MSC. En la siguiente cláusula se describe la sintaxis SDL que se puede utilizar en esta cadena.

Hay limitaciones para las expresiones autorizadas en el MSC, porque con una <expression> SDL se pueden crear estructuras lingüísticas que pueden causar efectos colaterales. En particular, no contendrán:

- <create expression>;
- <value returning procedure call>;
- <imperative expression>.

Los comentarios SDL (<note> y <comment body>) se pueden incluir en cualquiera de las cadenas terminales MSC. Las funciones semánticas estáticas definidas en la cláusula 4, indican las <expression> SDL que se pueden utilizar en un MSC.

3.4 Tipos de datos por defecto y comodines

Cuando el MSC se utiliza con los datos SDL de acuerdo con la declaración del lenguaje descrita en 3.1, se supone un lote SDL predefinido, de modo que los tipos de datos predefinidos son los que se indican en <data definition string> o <type ref string>, y los operadores predefinidos pueden estar indicados dentro de una <expression string>.

Los MSC requieren los tipos de datos booleano, número natural y de tiempo (véase 5.11/Z.120). Los tipos de datos correspondientes en la Rec. UIT-T Z.100 que se utilizan por defecto son:

<<**package** Predefined>> **value type** Boolean;

<<**package** Predefined>> **syntype** Natural;

<<**package** Predefined>> **value type** Time.

Para cada uno de estos tipos por defecto están declarados implícitamente los siguientes comodines por defecto:

wildcards

anyB: Boolean;

anyN: Natural;

anyT: Time.

4 Interfaz semántica

La interfaz semántica se define en tres partes:

- la buena formación, que describe la gramática de cadenas de datos SDL válidas que puedan utilizarse en MSC;
- semántica estática, que identifica los requisitos de tipo para establecer si es correcta una SDL bien formada;
- semántica dinámica, que define cómo evaluar cadenas de expresiones SDL bien formadas y de tipo correcto utilizadas en MSC.

4.1 Definiciones relativas a la buena formación

La interfaz de datos especifica cuatro funciones, Wf1, ... , Wf4, necesarias para definir cadenas de datos sintácticamente válidas. En el cuadro 4-1 se indica la gramática SDL correspondiente a las cadenas de datos válidas. Por ejemplo, Wf1(*v*) es verdadera, únicamente si *v* es una cadena que se puede producir mediante la regla gramatical SDL para <variable name>. Las cadenas pueden contener comentarios SDL en la forma usual, de modo que las definiciones se aplican a las cadenas después de haber filtrado los comentarios indicados mediante <note> o <comment body>.

Cuadro 4-1/Z.121 – Correspondencia de cadenas en la interfaz de datos MSC

Nombre de la función de buena formación	Cadena terminal MSC	Gramática SDL
Wf1	<variable string>	<variable name>
Wf2	<data definition string>	<left curly bracket> <data definition>* <right curly bracket>
Wf3	<type ref string>	<basic sort>
Wf4	<expression string>	<expression>

Para las cadenas de definición de datos, Wf2, es preciso definir las cadenas válidas mediante una regla gramatical auxiliar, utilizando <data definition> para generación de SDL, ya que no existe una generación de SDL que corresponda al requisito. La regla permite cualquier secuencia de <data definitions> encerrados entre un par de corchetes (definidos como <left curly bracket> y <right curly bracket>). Los corchetes evitan que se tenga que utilizar el carácter de escape para los caracteres punto y coma en las definiciones de datos, puesto que el punto y coma actúa como carácter delimitante para las propias cadenas de definición de datos. Así pues, una <data definition string> válida forma un cuerpo de lote SDL válido (un subconjunto de los elementos posibles en un lote).

4.2 Funciones de interfaz de la semántica estática

En la interfaz de datos se especifican cuatro predicados, Tc1, ... , Tc4, uno para cada clase de cadenas de datos, que son necesarios para afirmar que se ha respetado la semántica estática en las cadenas de datos válidos. Es decir, los predicados sólo se aplicarán a cadenas que satisfacen su correspondiente función de buena formación. Las reglas semánticas estáticas que determinan los cuatro predicados se describen en 6.3/Z.100 y cláusula 12/Z.100. También hay una relación auxiliar, EqVar, necesaria para la interfaz de datos y que se utiliza para determinar la equivalencia de cadenas de variables.

4.2.1 Tc1, cadenas de definición de datos

Tc1(*d*) es verdadera para una cadena de datos *d* sintácticamente válida – es decir, Wf2(*d*) es verdadera – si satisface la semántica estática del SDL, definida en 6.3/Z.100. En particular, si un cuerpo de lote formado por la cadena *d* menos los corchetes que la encierran satisface la sintaxis y la semántica estática de los lotes SDL, Tc1(*d*) es verdadera, y viceversa. Se ha representado esta condición en la figura 4-1, en la que la cadena de datos "{ *D* }" satisface Tc1 sólo cuando el lote SDL es correcto conforme a las reglas estáticas.

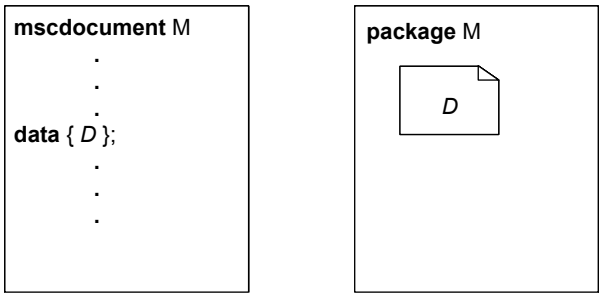


Figura 4-1/Z.121 – Requisitos estáticos del SDL equivalentes para cadenas de datos

4.2.2 Tc2, cadenas de referencia de tipo

Tc2(*d*)(*t*) es verdadera para una cadena de datos *d* sintácticamente válida y una cadena de referencia de tipo *t* – es decir Wf2(*d*) y Wf3(*t*) son verdaderas – únicamente si:

- Tc1(*d*), es decir *d*, es conforme a las semánticas estáticas para datos SDL,
- *t* es correcta según las reglas estáticas en el contexto de la cadena de datos *d*, de conformidad con las reglas estáticas del SDL.

El último requisito es equivalente a decir que la declaración de una variable de tipo *t* en el contexto de un lote que contenga *d* (menos los corchetes que lo encierran) es estáticamente correcta. Esto se ilustra en la figura 4-2, en la que el ejemplo de cadena de referencia de tipo *t* de una declaración de variable MSC satisface Tc2(*d*) sólo cuando el proceso SDL es estáticamente correcto. El lote al que hace referencia el proceso SDL se construye como en la figura 4-1.

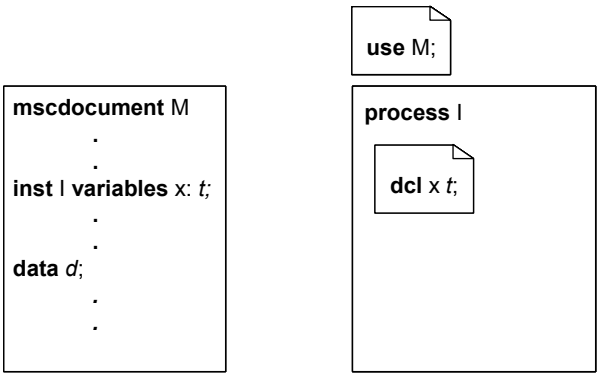


Figura 4-2/Z.121 – Requisitos estáticos del SDL equivalentes para cadenas de referencia de tipos

4.2.3 Tc3, cadenas de expresión

Tc3(*d*)(*S*)(*e*) es verdadera para una cadena de datos sintácticamente válida *d*, una cadena de expresión *e* y un conjunto de asignaciones de variables tipificadas *S*, únicamente si:

- Tc1(*d*), es decir *d*, es una cadena de definición de datos válida según las reglas estáticas;
- para cada par válido de cadena de tipo y cadena de variable (*v, t*) en el conjunto *S*:
  - Tc2(*d*)(*t*), es decir *t*, es una referencia de tipo válida según las reglas estáticas en el contexto de la cadena de datos *d*;
- *e* es una expresión SDL correcta según las reglas estáticas:
  - en el contexto de la cadena de datos *d*;
  - para las asignaciones de tipos a variables definidas por *S*.

La definición es equivalente a una expresión SDL *e* que satisface las reglas de tipo estáticas del SDL en el contexto de un lote que contenga la cadena de definición de datos (menos los corchetes que la encierran) *d*, y para cada variable *v* emparejada con la cadena de tipo *t* en *S*, existe una declaración de variable SDL de *v* que tenga el tipo *t*. Esto se ilustra en la figura 4-3 en la que la cadena de expresión *e* de un parámetro de mensajes MSC sólo satisface Tc3(*d*)(*S*), donde *S* es el conjunto de variables de tipo { (*x1*, *t1*), (*x2*, *t2*), … , (*xn*, *tn*) }, cuando el proceso SDL considerado es estáticamente correcto. El lote al que hace referencia el proceso SDL se construye como muestra la figura 4-1.

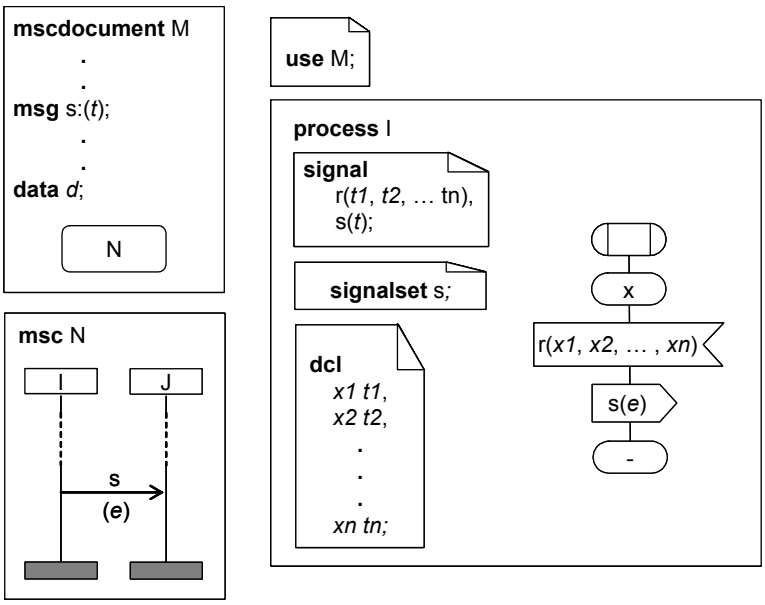


Figura 4-3/Z.121 – Requisitos estáticos SDL para cadenas de expresión

Por ejemplo, si una expresión contiene una variable no declarada en el conjunto *S*, no cumplirá las reglas estáticas SDL y por tanto el predicado Tc3 tampoco será verdadero.

Las cadenas de variables que aparecen en el conjunto de argumentos *S* deben indicar variables únicas. Por consiguiente, la *forma canónica* es la cadena de variables sin espacios en blanco o comentarios.

4.2.4 Tc4, cadenas de expresiones tipificadas

Tc4(*d*)(*S*)(*t*, *e*) es verdadera para una cadena de datos *d* y una cadena de expresión *e*, sintácticamente válidas, y un conjunto de asignaciones de variable tipificadas *S*, únicamente si:

- Tc2(*d*)(*t*), es decir *t*, es una cadena de referencia de tipo estáticamente válida en el contexto de la cadena de datos *d*;
- Tc3(*d*)(*S*)(*e*), es decir *e*, es una cadena de expresión sintáctica y estáticamente válida en el contexto de *d* y de una asignación de tipo de variable *S*;
- *e* puede tener el tipo *t* de acuerdo con las reglas estáticas del SDL.

La última cláusula es equivalente a decir que, además de las reglas definidas por el predicado Tc3, para una expresión estáticamente válida *e*, es válido que la expresión aparezca como argumento de la señal de salida del SDL que tenga el tipo de parámetro *t* declarado. Esto también se ilustra en la figura 4-3.

4.2.5 EqVar, cadenas de variables iguales

EqVar(*v1*, *v2*) es verdadera para dos cadenas de variables sintácticamente válidas *v1* y *v2*, únicamente si:

- las cadenas *v1'* y *v2'* que se obtienen al suprimir los comentarios y los espacios en blanco de *v1* y *v2* son idénticas.

Esta definición responde a las reglas de denominación del SDL, según las cuales son diferentes las variables que difieren en algún carácter, incluida la diferencia entre mayúsculas y minúsculas. Se dice que las cadenas *v1'* y *v2'* están en forma canónica, según se define en 4.2.3.

4.3 Funciones de interfaz semántica dinámica

La interfaz de datos requiere que se definan cuatro funciones –tres auxiliares y la función *Eval* que se utiliza para calcular el valor de la expresión SDL identificada con el predicado Wf2. Las funciones auxiliares tienen una finalidad evidente y no son necesariamente únicas– es decir, se pueden hacer diferentes interpretaciones sin afectar la función *Eval*. Cuando es necesario, la descripción de cada función se acompaña con las cláusulas pertinentes de las Recomendaciones SDL. Las funciones tienen que ver con los predicados de comprobación estática definidos anteriormente, que a la vez se basan en los predicados de buena formación definidos anteriormente.

4.3.1 Vars, extracción de variables

Vars es una función auxiliar necesaria para el cálculo de los trazos dinámicos de un MSC. Dado un contexto, el resultado de aplicar *Vars* a una expresión *e* es el conjunto de variables que contiene la expresión. Cada variable se empareja con el número de veces que aparece en la expresión *e*. Se puede dar la definición completa de *Vars* por inducción estructural a partir de la gramática SDL de expresiones, pero en esta cláusula sólo se da una definición informal; en primer lugar, se define el dominio de *Vars*.

Vars(*d*)(*S*)(*e*) está definida si:

- *d* es una cadena de datos sintácticamente válida, *e* es una cadena de expresiones sintácticamente válida y *S* es un conjunto de cadenas de variables emparejadas con una cadena de tipos sintácticamente válida;
- Tc3(*d*)(*S*)(*e*) es verdadera; es decir, si la cadena de expresión *e* es conforme a los requisitos estáticos del SDL.

Obsérvese que los requisitos de comprobación estática definidos por Tc3 garantizan que *Vars* sólo estará definida si todas las variables contenidas en la expresión están "declaradas" en el conjunto *S*.

Cuando Vars(*d*)(*S*)(*e*) esté definida, su valor vendrá dado por:

- si *e* es una constante, Vars(*d*)(*S*)(*e*) es un conjunto vacío;
- si *e* es una cadena de variables, Vars(*d*)(*S*)(*e*) = { (*v*, 1) }, donde *v* es la forma canónica de la cadena de variables definida en 4.2.3; (n.b. *v* debe estar en *S*);
- si *e* es una expresión compuesta, Vars(*d*)(*S*)(*e*) se calcula sumando el número de veces que aparece la variable en sus expresiones constituyentes.

4.3.2 Replace, sustitución de variable

Replace es una función auxiliar necesaria para calcular los trazos dinámicos de un MSC que utiliza comodines. No es necesario dar una definición unívoca de esta función para el lenguaje SDL, puesto que su efecto sería interno a toda aplicación que soporte MSC con datos SDL. Por consiguiente no es necesario definirla completamente en esta Recomendación.

Replace(*d*)(*v1*, *n*, *v2*)(*e*) está definida si:

- *d* es una cadena de datos sintácticamente válida, *v1* y *v2* son cadenas de variables sintácticamente válidas y *e* es una cadena de expresión sintácticamente válida;
- Tc3(*d*)(*S*)(*e*) es verdadera, es decir, si la cadena de expresión *e* es conforme a los requisitos estáticos del SDL.

El resultado de aplicar Replace(*d*)(*v1*, *n*, *v2*)(*e*) es una cadena de expresión *e′* en la que la *n*-ésima vez que aparece la variable *v1* se ha sustituido por la variable *v2*. Hay varias maneras de definir "la *n*-ésima vez" y no se especifica nada en esta Recomendación.

4.3.3 NewVar, nueva variable

Al igual que *Replace*, *NewVar* es una función auxiliar necesaria para calcular los trazos dinámicos de un MSC que utiliza comodines. No es necesario dar una definición unívoca de esta función para

el lenguaje SDL, puesto que su efecto sería interno a toda aplicación que soporte MSC con datos SDL. Por consiguiente, en esta Recomendación no se obliga a utilizarla de un modo concreto.

NewVar(*d*)(*S*) está definida si:

- *d* es una cadena de datos sintácticamente válida y *S* es un conjunto de cadenas de variables, cada una de ellas emparejada con una cadena de tipo sintácticamente válida;
- Tc1(*d*) es verdadera, es decir, si la cadena de datos *d* menos los corchetes que la encierran es conforme a los requisitos estáticos para el cuerpo del lote SDL.

El resultado de aplicar NewVar(*d*)(*S*) es una cadena de variables SDL sinstácticamente válida distinta a todas las contenidas en el conjunto *S*, según se define mediante la función EqVar. Hay muchas maneras de definir cómo se determina la nueva cadena de variables, y en esta Recomendación no se obliga a utilizar ningún modo concreto.

4.3.4 Eval, evaluación de expresiones

Eval es una función necesaria para el cálculo de los trazos dinámicos de un MSC. Dado un contexto de datos y una asignación de valores a las variables –el estado actual– el resultado de aplicar *Eval* a una expresión *e* es su valor. Hay una relación formal entre la definición de esta función y:

a) la función *compute* según se define en 2.1.3.1/Z.100 anexo F3;

b) la evaluación de una función a partir de sus operandos según se define en 2.1.3.1/Z.100 anexo F3 y

c) la función *Eval* de 3.5/Z.100 anexo F3. La semántica de las expresiones en SDL se describe en 12.2/Z.100.

Eval(*d*)(*e*)(*A*) está definida sólo si:

- *d* es una cadena de datos sintácticamente válida, *e* es una cadena de expresión sintácticamente válida y *A* es un conjunto de cadenas de variables, cada una emparejada con un valor de datos;
- Tc3(*d*)(*S*)(*e*) es verdadera, es decir, si la cadena de expresión *e* es conforme a los requisitos estáticos del SDL en el contexto de *d*, en el que sus variables tienen asignaciones de tipo definidas por el conjunto *S*.

Obsérvese que se pueden satisfacer las condiciones anteriores sin que esté definida la función *Eval*, si la expresión se calcula fuera de su dominio; por ejemplo, si *e* contiene una división por cero. La segunda cláusula contiene un conjunto de variables *S* etiquetadas por el tipo que debe corresponder al conjunto de variables *A* etiquetadas por el valor del dominio; se supone que los tipos se pueden deducir a partir de los valores de dominio para formar el conjunto *S* necesario a partir de *A*.

El valor de Eval(*d*)(*e*)(*A*) es el valor que toma la expresión SDL *e* en el contexto de un lote cuyo cuerpo corresponde a la cadena de datos *d*, y cuyas variables tienen los valores definidos por el estado *A*. Más formalmente, se considera que la función Eval(*d*)(*e*) corresponde a una función definida por un sistema SDL sencillo en el que:

- hay una señal de entrada que toma como parámetros una lista de variables, que corresponden a las variables en el estado *A*;
- hay una señal de salida cuyo único parámetro se define mediante la expresión *e*;
- hay un lote cuyo cuerpo consta de la cadena de datos *d* menos los corchetes que la encierran.

Ejecutando un sistema SDL de este tipo se calculará y obtendrá como resultado el valor de la expresión *e*, para una señal de entrada con valores que se asignan a cada una de las variables de la expresión. Esto se muestra en la figura 4-3: el valor de Eval(*d*)(*e*)(*A*), donde *A* es el conjunto de asignaciones de variables { (*x1*, *a1*), (*x2*, *a2*), … , (*xn*, *an*) }, es igual al valor de salida de la señal s según el valor de la señal de entrada r (*a1*, *a2*, … , *an*). Dado que *A* es un conjunto y que la

entrada r toma una lista de parámetros, la correspondencia unívoca se puede formar mediante la ordenación lexicográfica de las cadenas variables, de modo que *x1* sea la cadena variable que lexicográficamente es la primera, y *a1* sea el primer valor de parámetro, etc.

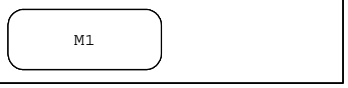
5 Ejemplo

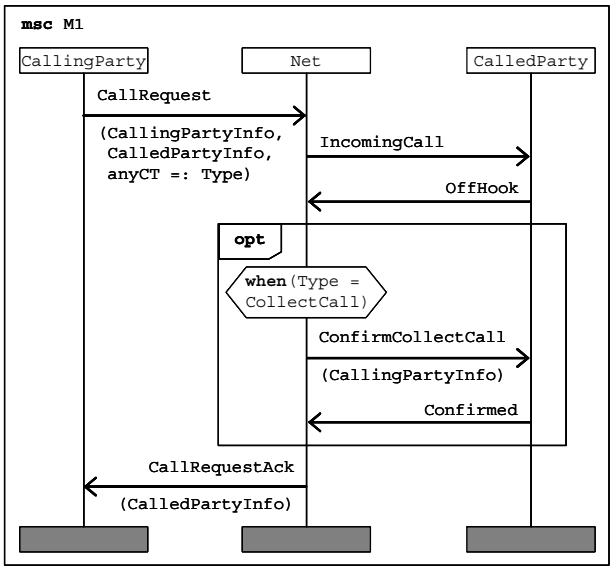
5.1 Utilización de la interfaz SDL por defecto

```
mscdocument M
inst Net
  variables Type: CallType;
inst CallingParty;
inst CalledParty;
msg  CallRequest: (SubscriberInfoType,
                   SubscriberInfoType,
                   CallType);
msg  ConfirmCollectCall,
     CallRequestAck: (SubscriberInfoType);

language  SDL;
wildcards anyCT: CallType;

data {
  value type SubscriberInfoType {
    literals CallingPartyInfo, CalledPartyInfo;
  }
  value type CallType {
    literals CollectCall, OrdinaryCall;
  }
};
```





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**

Ginebra, 2003