UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T Z.121

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

(02/2003)

SÉRIE Z: LANGAGES ET ASPECTS GÉNÉRAUX LOGICIELS DES SYSTÈMES DE TÉLÉCOMMUNICATION

Techniques de description formelle – Diagrammes des séquences de messages

Rattachement des données SDL aux diagrammes MSC

Recommandation UIT-T Z.121

RECOMMANDATIONS UIT-T DE LA SÉRIE Z

LANGAGES ET ASPECTS GÉNÉRAUX LOGICIELS DES SYSTÈMES DE TÉLÉCOMMUNICATION

TECHNIQUES DE DESCRIPTION FORMELLE	
Langage de description et de spécification (SDL)	Z.100–Z.109
Application des techniques de description formelle	Z.110–Z.119
Diagrammes des séquences de messages	**Z.120–Z.129**
Langage étendu de définition d'objets	Z.130–Z.139
Notation combinée arborescente et tabulaire	Z.140–Z.149
Notation de prescriptions d'utilisateur	Z.150–Z.159
LANGAGES DE PROGRAMMATION	
CHILL: le langage de haut niveau de l'UIT-T	Z.200–Z.209
LANGAGE HOMME-MACHINE	
Principes généraux	Z.300–Z.309
Syntaxe de base et procédures de dialogue	Z.310–Z.319
LHM étendu pour terminaux à écrans de visualisation	Z.320–Z.329
Spécification de l'interface homme-machine	Z.330–Z.349
Interfaces homme-machine orientées données	Z.350–Z.359
Interfaces homme-machine pour la gestion des réseaux de télécommunication	Z.360–Z.369
QUALITÉ	
Qualité des logiciels de télécommunication	Z.400–Z.409
Aspects qualité des Recommandations relatives aux protocoles	Z.450–Z.459
MÉTHODES	
Méthodes de validation et d'essai	Z.500–Z.519
INTERGICIELS	
Environnement de traitement réparti	Z.600–Z.609

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

Recommandation UIT-T Z.121

Rattachement des données SDL aux diagrammes MSC

Résumé

La présente Recommandation présente l'instanciation en langage SDL (Z.100) des éléments syntaxiques et sémantiques de l'interface de données MSC (Z.120) et définit les types par défaut ainsi que la syntaxe applicable aux définitions autorisées des données SDL pouvant être utilisées dans un document MSC.

Source

La Recommandation Z.121 de l'UIT-T, élaborée par la Commission d'études 17 (2001-2004) de l'UIT-T, a été approuvée le 13 février 2003 selon la procédure définie dans la Résolution 1 de l'AMNT.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de l'AMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

Introduction

Le rattachement des données en langage SDL aux diagrammes MSC est décrit en deux parties. La première est la partie syntaxique de l'interface qui définit les déclarations de documents MSC pour lesquelles le SDL est le langage de données, la deuxième partie décrivant la sémantique de l'interface. Cette deuxième partie comporte une définition d'un certain nombre de fonctions qui sont utilisées pour l'évaluation syntaxique, sémantique statique et sémantique dynamique des données SDL utilisées en MSC.

Recommandation UIT-T Z.121

Rattachement des données SDL aux diagrammes MSC

1 Domaine d'application

La présente Recommandation décrit une instanciation en langage SDL (Z.100) de l'interface de données pour les diagrammes des séquences de messages (Z.120) qui est défini comme étant le langage par défaut pour la Rec. UIT-T Z.120.

Dans la Rec. UIT-T Z.120 sur les diagrammes MSC est définie une interface de données ouverte qui permet à différents utilisateurs du langage d'utiliser différents langages de données dans les diagrammes et documents MSC. La présente Recommandation définit un rattachement de l'interface ouverte à un fragment de langage de données en SDL, qui doit être utilisé comme langage de données par défaut. C'est-à-dire, en l'absence d'une instanciation explicite de l'interface de données dans un document MSC, on doit supposer que le rattachement défini par la présente Recommandation et le langage des chaînes de définition de données autorisé doivent s'appliquer.

Le présent texte est la première version de la Rec. UIT-T Z.121, fondée sur la définition de l'interface de données apparue d'abord dans l'actuelle Rec. UIT-T Z.120, publiée en 1999.

2 Références normatives

La présente Recommandation se réfère à certaines dispositions des Recommandations UIT-T et textes suivants qui, de ce fait, en sont partie intégrante. Les versions indiquées étaient en vigueur au moment de la publication de la présente Recommandation. Toute Recommandation ou tout texte étant sujet à révision, les utilisateurs de la présente Recommandation sont invités à se reporter, si possible, aux versions les plus récentes des références normatives suivantes. La liste des Recommandations de l'UIT-T en vigueur est régulièrement publiée. La référence à un document figurant dans la présente Recommandation ne donne pas à ce document en tant que telle statut d'une Recommandation.

– Recommandation UIT-T Z.100 (2002), *SDL: Langage de description et de spécification*.

– Recommandation UIT-T Z.120 (1999), *Diagramme des séquences de messages*.

3 Interface syntaxique

Le document MSC contient une déclaration du nom du langage de données utilisé, afin que les outils qui prennent en charge plusieurs langages puissent le distinguer. Il définit également des déclarations de séquences de parenthèses utilisées par le langage de données de manière à ce que les outils puissent correctement identifier les limites d'une chaîne de données intégrée dans des messages MSC. Dans le présent paragraphe sont données les déclarations par défaut pour le rattachement des données SDL.

La déclaration de langage est effectuée à l'intérieur des définitions de données, et contient également les déclarations de données utiles: types de données (avec les méthodes et les opérateurs), les syntypes et les synonymes.

3.1 Déclaration de langage

La déclaration de langage peut être effectuée dans le document MSC et elle identifie le langage de données qui est utilisé dans tous les diagrammes MSC dans le domaine de visibilité du document MSC englobant. La déclaration de langage suivante est définie pour le SDL, bien qu'étant le langage par défaut, son utilisation est optionnelle:

langage SDL;

Si cette déclaration de langage est présente, les déclarations de parenthèses et d'échappement sont implicites et n'ont pas besoin d'être explicitement spécifiées dans chaque document MSC, il en est de même pour les types par défaut accompagnés de leurs caractères génériques correspondants.

3.2 Déclaration de parenthèse et d'échappement

Lorsque des données SDL sont utilisées dans des diagrammes MSC conformément à la déclaration de langage ci-dessus, on suppose la présence des déclarations de parenthèse et d'échappement suivantes:

parenthesis

nestable '()', '[]', '()', '{}';

nonnestable '/*'*/', '/#'#/', '<<'>>';

equalpar /'/;

escape '?';

;

Cette déclaration signifie que:

- les <comment> et <note> (/# commentaire #/ et /* note */) en langue locale sont acceptés dans les chaînes de données;
- les qualificatifs Z.100 (par exemple, <<package BasicTypes>>) peuvent être utilisés pour résoudre les conflits de nom;
- les chaînes de caractères Z.100 ('hello world!') sont autorisées dans des chaînes d'expression, en utilisant la forme locale Z.100 d'une apostrophe répétée (") pour permettre une seule apostrophe dans une chaîne;
- '?' est utilisé comme caractère d'échappement général pour permettre la présence de jetons d'arrêt MSC dans les chaînes de données.

Si un jeton MSC qui peut indiquer la fin d'une chaîne de données, tel un point-virgule, doit être *utilisé ouvertement* à l'intérieur de la chaîne de données, il doit faire l'objet d'un échappement avec le caractère '?'. Par *utilisé ouvertement*, on entend qu'il n'est pas "dissimulé" entre des parenthèses déclarées en SDL. Par exemple, pour un point-virgule, un échappement n'est pas nécessaire lorsque ce point-virgule apparaît dans une chaîne SDL car il est dissimulé par les caractères délimiteurs de chaîne.

Le seul jeton SDL qui doit faire l'objet d'un échappement dans un contexte du SDL est le guillemet simple dans une expression chaîne. Les chaînes SDL n'exigent pas de traitement particulier lorsqu'elles sont utilisées à l'intérieur d'une chaîne de données MSC.

Un interpréteur MSC recherche la correspondance de parenthèse la plus longue. Ainsi: les deux caractères '(.' et '(' sont des parenthèses imbricables, de sorte que l'interpréteur détectera une imbrication incorrecte de parenthèses.

3.3 Déclaration de données et utilisation

Les déclarations explicites de type réel de données en langage de données local sont effectuées avec la chaîne ouverte <data definition string> dans un document MSC. Le paragraphe suivant décrit la syntaxe SDL autorisée à l'intérieur de cette chaîne.

Etant donné que les <expression> SDL en général, autorisent des constructions qui peuvent introduire des effets secondaires, l'expression autorisée dans un diagramme MSC sera soumise à certaines contraintes: en particulier, elles ne devront pas contenir les éléments suivants:

- <create expression>;

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

Les commentaires SDL (<note> et <comment body>) peuvent être intégrés dans l'une quelconque des chaînes terminales MSC. Les fonctions sémantiques statiques définies au § 4 définissent les <expression> en SDL qui sont autorisées dans un diagramme MSC.

3.4 Type de données par défaut et caractères génériques

Lorsque le MSC est utilisé avec des données SDL, conformément à la déclaration de langage décrite au § 3.1, on suppose la présence du paquetage SDL "Predefined", de sorte que les types de données prédéfinis peuvent être référencés à partir de <data definition string> ou de <type ref string> et il peut être fait référence à des opérateurs prédéfinis à l'intérieur d'une chaîne <expression string>.

Le MSC exige des types de données Booléens, naturels et temporels (voir § 5.11/Z.120). Les types de données correspondant dans la Rec. UIT-T Z.100 à utiliser par défaut sont les suivants:

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

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

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

Pour chacun de ces types par défaut, un caractère générique par défaut est implicitement déclaré comme suit:

wildcards

anyB: Boolean;

anyN: Natural;

anyT: Time.

4 Interface sémantique

L'interface sémantique est définie en trois parties:

- bien formée, décrivant la grammaire des chaînes de données SDL autorisées à utiliser dans le MSC;
- la sémantique statique, identifiant les exigences en matière de conformité du type de chaîne de données SDL bien constituée;
- la sémantique dynamique qui définit comment évaluer la bonne constitution et le type des chaînes d'expression SDL correctes utilisées en MSC.

4.1 Définitions de la bonne constitution

L'interface de données spécifie quatre fonctions, Wf1, … , Wf4 qui sont requises pour définir des chaînes de données dont la syntaxe est valide. Le Tableau 4-1 identifie la grammaire SDL qui correspond à des chaînes de données valides. Par exemple Wf1(ν) est vrai si et seulement si ν est une chaîne peut être produite par la règle de grammaire SDL applicable à <variable name>. Les chaînes contiennent des commentaires de la façon normale de sorte que les définitions s'appliquent à toutes chaînes respectant le filtrage de commentaires tel que défini par <note> ou <comment body>.

Tableau 4-1/Z.121 – Mappage des chaînes au niveau de l'interface de données MSC

Nom de la fonction de la bonne constitution	Chaînes terminales MSC	Grammaire 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>

Dans le cas de chaînes de définition de données Wf2, les chaînes valides doivent être définies par une règle de grammaire auxiliaire en termes de production SDL de <data definition>, puisqu'il n'existe pas de production SDL qui correspond à la condition imposée. La règle permet d'encadrer toute séquence de <data definitions> par une paire d'accolades (définie par <left curly bracket> et <right curly bracket>). Les accolades évitent d'appliquer un échappement sur les caractères points-virgules apparaissant dans les définitions de données, étant donné qu'un point-virgule agit comme caractère délimiteur pour la chaîne de définition de données. Ainsi une chaîne <data definition string> forme un corps de paquetage SDL valide – car c'est un sous-ensemble des éléments autorisés dans un paquetage.

4.2 Fonctions d'interface de sémantique statique

L'interface de données spécifie quatre prédicats, Tc1, … , Tc4, à raison d'un par classe de chaîne de données, qui sont exigés pour affirmer que la sémantique statique des chaînes de données valides est respectée. C'est-à-dire que les prédicats ne seront appliqués qu'aux chaînes qui satisfont la fonction de bonne constitution correspondante. Les règles de sémantique statique qui déterminent les quatre prédicats sont décrites aux § 6.3/Z.100 et 12/Z.100. Il y a aussi EqVar, une relation auxiliaire, exigée par l'interface de données utilisée pour identifier l'équivalence entre chaînes variables.

4.2.1 Prédicat Tc1, chaînes de définition de données

Le prédicat Tc1(*d*) est vérifié pour une chaîne de données syntaxiquement valide *d* – c'est-à-dire que Wf2(*d*) est Vrai – si elle respecte la sémantique statique du SDL définie au § 6.3/Z.100. En particulier, si un corps d'un paquetage composé de la chaîne *d*, moins les accolades qui la délimitent, respecte la syntaxe et la sémantique des paquetages SDL, le prédicat Tc1(*d*) est alors Vrai, et inversement. C'est ce qu'illustre la Figure 4-1 dans laquelle la chaîne de données "{*D*}" satisfait le prédicat Tc1 seulement lorsque le paquetage SDL est syntaxiquement correct.

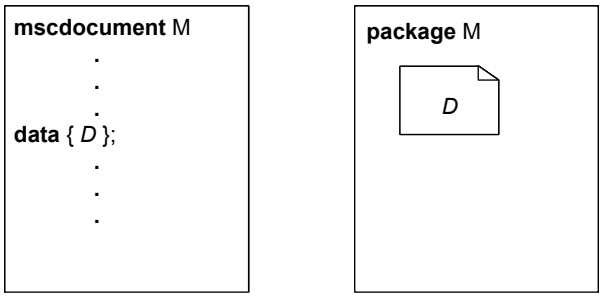


Figure 4-1/Z.121 – Conditions statiques SDL équivalentes à respecter pour les chaînes de données

4.2.2 Prédicat Tc2, chaînes de référence de type

Le prédicat Tc2(*d*)(*t*) est Vrai d'une chaîne de données syntaxiquement valide *d* et d'une chaîne de référence de type *t* – c'est-à-dire que Wf2(*d*) et Wf3(*t*) sont Vrais – si et seulement si:

- Tc1(*d*), c'est-à-dire *d* respecte la sémantique statique pour les données SDL;
- *t* est statiquement correct dans le contexte de la chaîne de données *d* conformément aux règles statiques du SDL.

Cette dernière condition équivaut à dire qu'une déclaration d'une variable ayant le type *t* dans le contexte d'un paquetage contenant *d* (moins les accolades qui le délimitent) est syntaxiquement correcte. C'est ce qu'illustre la Figure 4-2 dans laquelle la chaîne de référence de type *t* servant d'exemple de déclaration de variable MSC est conforme au prédicat Tc2(*d*) seulement lorsque le processus SDL est syntaxiquement correct. Le paquetage référencé par le processus SDL est construit comme indiqué dans la Figure 4-1.

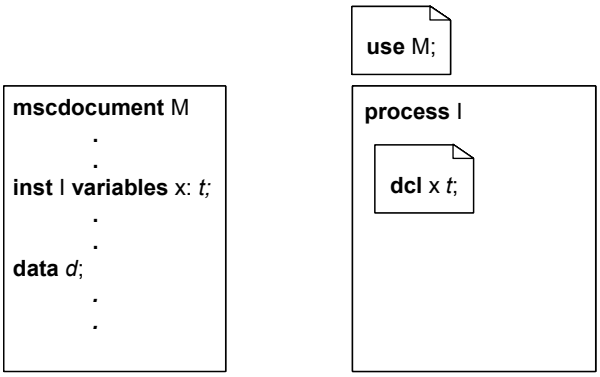


Figure 4-2/Z.121 – Conditions statiques SDL équivalentes pour une chaîne de référence de type

4.2.3 Prédicat Tc3, chaînes d'expression

Le prédicat Tc3(*d*)(*S*)(*e*) est vrai d'une chaîne de données *d* syntaxiquement valide et d'une chaîne d'expression *e* et d'un ensemble d'assignations variables typées *S*, si et seulement si:

- le prédicat Tc1(*d*) est vérifié, c'est-à-dire *d* est une chaîne de définitions de données statiquement valide;
- pour chaque paire de chaînes de type de chaînes variables valides (*v, t*) dans l'ensemble *S*:
  - le prédicat Tc2(*d*)(*t*) est vérifié, c'est-à-dire *t* est une référence de type statiquement valide dans le contexte d'une chaîne de données *d*;
- *e* est une expression SDL statiquement correcte;
  - dans le contexte d'une chaîne de données *d*;
  - compte tenu de l'assignation de types à des variables définies par *S*.

La définition est équivalente à une expression SDL *e* respectant les règles de type statiques du SDL dans le contexte d'un paquetage contenant la chaîne de définitions de données (moins les accolades qui la délimitent) *d*, et pour chaque appariement d'une variable *v* avec une chaîne de type *t* dans *S*, il existe une déclaration de variable SDL de ν ayant le type *t*. C'est ce qu'illustre la Figure 4-3 dans laquelle la chaîne d'expression *e* d'un paramètre de message MSC satisfait le prédicat Tc3(*d*)(*S*), dans lequel *S* est l'ensemble des variables typé { *(x1, t1), (x2, t2), ... , (xn, tn)* }, seulement lorsque le processus SDL représenté est statiquement correct. Le paquetage référencé par le processus SDL est construit comme indiqué dans la Figure 4-1.

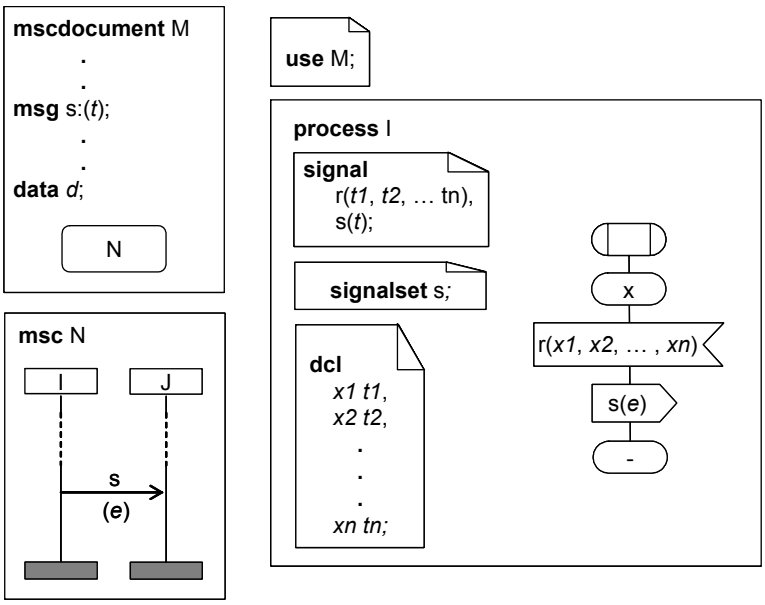


Figure 4-3/Z.121 – Conditions statiques SDL équivalentes pour la chaîne d'expression

Par exemple, si une expression contient une variable non déclarée dans l'ensemble *S*, elle ne respecte pas alors les règles statiques SDL et par conséquent le prédicat Tc3 sera également Faux.

Les chaînes variables apparaissant dans l'ensemble d'arguments *S* doivent désigner des variables uniques, et par conséquent nous définissons une forme canonique (*canonical form*) comme étant la chaîne variable, et liée à tout espace blanc environnant ou commentaires.

4.2.4 Prédicat Tc4, chaînes d'expression typée

Le prédicat Tc4(*d*)(*S*)(*t, e*) est Vrai d'une chaîne de données syntaxiquement valable et d'une chaîne d'expression *e*, et d'un ensemble d'assignations variables typées *S*, si et seulement si:

- le prédicat Tc2(*d*)(*t*) est vérifié, c'est-à-dire *t* est une chaîne de référence de type statiquement valable dans le contexte d'une chaîne de données *d*;
- le prédicat Tc3(*d*)(*S*)(*e*) est vérifié, c'est-à-dire *e* est une chaîne d'expression syntaxiquement et statiquement valide dans le contexte de *d* et d'une assignation de type variable *S*;
- *e* peut avoir le type *t* conformément aux règles statiques du SDL.

La dernière prescription revient à dire qu'en plus des règles données pour une expression *e* statiquement valable tel que défini par le prédicat Tc3, l'expression légalement peut apparaître comme argument associé à un signal de sortie SDL ayant le type de paramètre déclaré *t*. C'est ce qu'illustre également la Figure 4-3.

4.2.5 EqVar, chaînes égales variables

La relation EqVar(*v1, v2*) est Vraie de deux chaînes variables *v1* et *v2* syntaxiquement valides si et seulement si:

- Les chaînes *v1'* et *v2'* restant après suppression des commentaires et de l'espace blanc de *v1* et *v2* respectivement sont identiques.

Cette définition reflète les règles de nommage pour le SDL, dans lesquelles les variables étant différentes dans chaque caractère, incluant des différences en majuscule et minuscule, représentent, différentes variables. Les chaînes *v1'* et *v2'* sont dites être de forme canonique, tel que défini au § 4.2.3.

4.3 Fonctions d'interface avec sémantique dynamique

L'interface de données requiert la définition de quatre fonctions – trois sont auxiliaires et la dernière, la fonction *Eval*, est utilisée ici pour calculer la valeur de l'expression SDL identifiée avec le prédicat Wf2. Les fonctions auxiliaires sont immédiatement compréhensibles et pas nécessairement uniques – c'est-à-dire que différentes interprétations peuvent être faites sans affecter la fonction *Eval*. Les paragraphes pertinents des Recommandations sur le SDL sont données comme faisant partie de chaque description de fonction si nécessaire. La fonction renvoie aux prédicats de vérification statique précédemment définis, lesquels, en revanche, reposent sur des prédicats de bonne constitution initialement définis.

4.3.1 Vars, variables d'extraction

Vars est une fonction subsidiaire qui est requise dans le calcul des traces dynamiques d'un diagramme MSC. Dans un contexte donné, le résultat de l'application de *Vars* à une expression *e* est l'ensemble des variables que l'expression contient. Chaque variable est associée au nombre d'occurrences de la variable dans l'expression *e*. la définition complète de *Vars* peut être donnée par induction structurelle sur la grammaire SDL des expressions, mais seule une définition informelle est donnée ici; le domaine de *Vars* est tout d'abord défini.

Vars(*d*)(*S*)(*e*) est défini si:

- *d* est une chaîne de données syntaxiquement valide, *e* est une chaîne d'expressions syntaxiquement valide et *S* un ensemble de chaînes variables associées chacune à une chaîne type syntaxiquement valide;
- le prédicat Tc3(*d*)(*S*)(*e*) est Vrai, c'est-à-dire la chaîne d'expression *e* est conforme aux exigences statiques du SDL.

A noter que les prescriptions en matière de vérification statique définies par le prédicat Tc3 garantissent que *Vars* ne peut être défini que si toutes les variables contenues dans l'expression sont 'déclarées' dans l'ensemble *S*.

A condition de définir Vars(*d*)(*S*)(*e*), cette valeur est donnée par:

- si *e* est une constante: Vars(*d*)(*S*)(*e*) est l'ensemble vide;
- si *e* est une chaîne variable: Vars(*d*)(*S*)(*e*) = { (*v*, 1) }, où *v* est la forme canonique de la chaîne variable définie au § 4.2.3; n.b. *v* doit apparaître dans *S*;
- si *e* est une expression composite, Vars(*d*)(*S*)(*e*) est calculé en additionnant le nombre d'occurrences d'une variable dans ses expressions constitutives.

4.3.2 Replace, remplacement de variable

Replace est une fonction subsidiaire qui est requise pour calculer les traces dynamiques d'un diagramme MSC qui emploie des caractères génériques. En tant que telle, sa définition ne doit pas être définie de manière unique pour le langage SDL, étant donné que ses effets seront internes à tout outil prenant en charge le MSC avec des données SDL. Par conséquent, la présente Recommandation n'a pas besoin de donner sa définition complètement.

Replace(*d*)(*v1*, *n*, *v2*)(*e*) est défini si:

- *d* est une chaîne de données syntaxiquement valable, *v1* et *v2* sont des chaînes variables syntaxiquement valides, et *e* est une chaîne d'expression syntaxiquement valide;
- Tc3(*d*)(*S*)(*e*) est Vrai, c'est-à-dire la chaîne d'expression *e* est conforme aux prescriptions statiques du SDL.

Le résultat de Replace(*d*)(*v1*, *n*, *v2*)(*e*) est une chaîne d'expression *e′* dans laquelle l'énième occurrence de la variable *v1* a été remplacée par la variable *v2*. Il existe un certain nombre de façons permettant de définir l''énième occurrence', et la présente Recommandation n'impose pas de définition particulière.

4.3.3 NewVar, nouvelle variable

Tout comme *Replace*, *NewVar* est une fonction subsidiaire qui est requise pour calculer les traces dynamiques d'un diagramme MSC qui utilise les caractères génériques. En tant que telle, sa définition ne doit pas nécessairement être unique en SDL, étant donné que ses effets seraient internes à tout outil prenant en charge le MSC avec des données SDL. Par conséquent, la présente Recommandation n'impose pas de modalité particulière en la matière.

NewVar(*d*)(*S*) est défini si:

- *d* est une chaîne de données syntaxiquement valide, *S* un ensemble de chaînes de variables appariées chacune à une chaîne type syntaxiquement valide;
- le précidat Tc1(*d*) est Vrai, c'est-à-dire la chaîne de données *d* moins les accolades qui la délimitent est conforme aux prescriptions statiques d'un corps de paquetage SDL.

Le résultat de NewVar(*d*)(*S*) est une chaîne variable SDL syntaxiquement valide qui est différente de toute autre contenue dans l'ensemble *S*, tel que défini par la fonction EqVar. Il existe de nombreuse façons de définir comment la nouvelle chaîne variable est déterminée et la présente Recommandation n'impose pas de modalité particulière en la matière.

4.3.4 Eval, évaluation d'expressions

Eval est une fonction requise pour le calcul des traces dynamiques d'un diagramme MSC. Etant donnés un contexte de données et une assignation de valeurs aux variables – l'état courant – le résultat de l'application d'Eval à une expression *e* est sa valeur. Sa définition est formellement liée à:

a) la fonction *compute* du 3, comme définie au 2.1.3.1/Z.100 Annexe F3;

b) l'évaluation d'une fonction sur ses opérantes telle que définie au 2.1.3.1/Z.100 Annexe F3;

c) la fonction *Eval* au 3.5/Z.100 Annexe F3. La sémantique des expressions en SDL est décrite au 12.2/Z.100.

Eval(*d*)(*e*)(*A*) est défini seulement si:

- *d* est une chaîne de données syntaxiquement valide, *e* est une chaîne d'expression syntaxiquement valide et *A* un ensemble de chaînes variables chacune étant appariée à une valeur de données;
- le prédicat Tc3(*d*)(*S*)(*e*) est vrai, c'est-à-dire la chaîne d'expression *e* est conforme aux prescriptions statiques du SDL dans le contexte de *d* où ses variables ont des assignations de type définies par l'ensemble *S*.

A noter que les conditions ci-dessous peuvent être satisfaites, mais que la fonction *Eval* n'est pas définie si l'expression est calculée en dehors de son domaine; par exemple si *e* contient une division par zéro. La seconde condition contient un ensemble étiqueté par type de variables *S* qui doivent correspondre à l'ensemble étiqueté de valeurs de domaine des variables *A*; on suppose que les types peuvent être déduits des valeurs de domaines pour former l'ensemble *S* requis à partir de *A*.

La valeur de Eval(*d*)(*e*)(*A*) est la valeur que l'expression *e* en SDL prend dans le contexte d'un paquetage ayant un corps correspondant à la chaîne de données *d*, chaque fois que ses variables ont des valeurs définies par l'état *A*. Plus formellement, la fonction Eval(*d*)(*e*) est définie comme étant la même que la fonction définie par un simple système SDL dans lequel:

- il y a un signal d'entrée qui prend une liste de variables comme paramètres, chacune des variables correspondant à une variable dans l'état *A*;
- il y a un signal de sortie dont le seul paramètre est défini par l'expression *e*;
- il y a un paquetage dont le corps se compose d'une chaîne de données *d* moins ses accolades qui la délimitent.

Une exécution d'un tel système SDL se traduira par le calcul et l'attribution en sortie de la valeur de l'expression *e*, si le signal d'entrée contient des valeurs qui sont utilisées pour assigner des valeurs à chacune des variables d'expression. C'est ce qu'illustre la Figure 4-3 dans laquelle la valeur de Eval(*d*)(*e*)(*A*), où *A* est l'ensemble des assignations de variable { (*x1*, *a1*), (*x2*, *a2*), … , (*xn*, *an*) }, est égale à la valeur de sortie du signal s, chaque fois qu'est donné au signal d'entrée r la valeur (*a1*, *a2*, … , *an*). Etant donné que *A* est un ensemble mais que l'entrée r prend une liste de paramètres, une correspondance unique peut être établie par mise en ordre lexicographique des chaînes variables, de sorte que *x1* sera la chaîne variable qui, de manière lexicographique, vient en premier, et *a1* sera la première valeur du paramètre, etc.

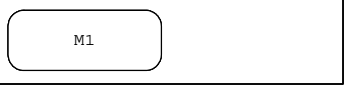
5 Exemple

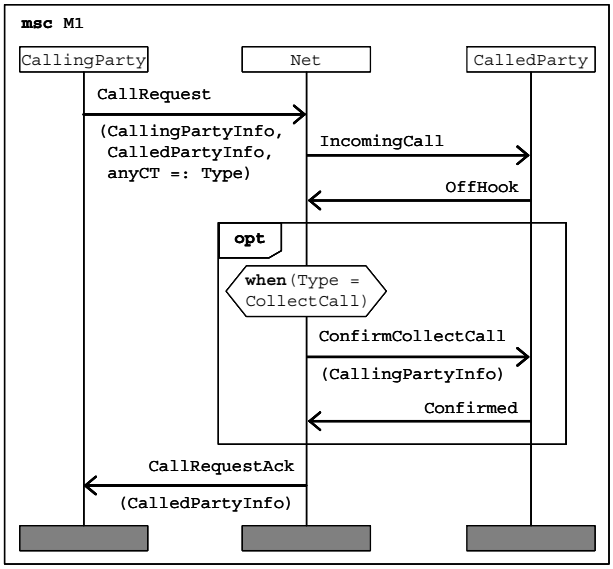
5.1 Utilisation de l'interface SDL par défaut

```
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;
  }
};
```





SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Réseaux câblés et transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, circuits téléphoniques, télégraphie, télécopie et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux de données et communication entre systèmes ouverts
Série Y	Infrastructure mondiale de l'information et protocole Internet
Série Z	**Langages et aspects généraux logiciels des systèmes de télécommunication**

Genève, 2003