Remplacée par une version plus récente

UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T Z.120

(10/96)

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

SÉRIE Z: LANGAGES DE PROGRAMMATION

Techniques de description formelle – Diagrammes des séquences de messages

---

Diagramme de séquence des messages

Recommandation UIT-T Z.120
Remplacée par une version plus récente

(Antérieurement Recommandation du CCITT)

Remplacée par une version plus récente

RECOMMANDATIONS UIT-T DE LA SÉRIE Z

LANGAGES DE PROGRAMMATION

Langage de description et de spécification (SDL)	Z.100–Z.109
Critères d'utilisation et d'applicabilité des techniques de description formelle	**Z.110–Z.199**
Langage de haut niveau de l'UIT-T (CHILL)	Z.200–Z.299
LANGAGE HOMME-MACHINE	Z.300–Z.499
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.399
Divers	Z.400–Z.499

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

Remplacée par une version plus récente

AVANT-PROPOS

L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'Union internationale des télécommunications (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.

La Conférence mondiale de normalisation des télécommunications (CMNT), qui se réunit tous les quatre ans, détermine les thèmes d'études 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 n° 1 de la CMNT (Helsinki, 1er-12 mars 1993).

La Recommandation révisée UIT-T Z.120, que l'on doit à la Commission d'études 10 (1993-1996) de l'UIT-T, a été approuvée par la CMNT (Genève, 9-18 octobre 1996).

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 de télécommunications.

Remplacée par une version plus récente

TABLE DES MATIÈRES

Page

Remplacée par une version plus récente

CHAMP D'APPLICATION/OBJECTIFS

L'objet de la recommandation diagramme de séquence des messages (MSC, *message sequence chart*) est de fournir un langage de trace pour la spécification et la description du comportement de la communication entre les composants d'un système et leur environnement au moyen d'échanges de messages. Puisque la représentation du comportement d'une communication à l'aide des MSC est faite de manière intuitive et claire, en particulier au niveau de la représentation graphique, le langage MSC est facile à apprendre, à pratiquer et à interpréter. Il peut être utilisé en relation avec d'autres langages en tant que support méthodologique pour la spécification, la conception, la simulation, le test et la documentation de système.

CONTENU

La présente Recommandation définit la syntaxe des diagrammes de séquence des messages sous forme de représentation textuelle et graphique. Une description informelle de la sémantique est également fournie.

CHAMP D'APPLICATION

Le champ d'application principal du MSC est la spécification d'une vue d'ensemble des comportements de la communication au sein des systèmes temps réel et plus particulièrement des systèmes de télécommunication. Tout d'abord, des traces choisies du comportement d'un système, correspondant aux cas "standards", peuvent être spécifiées au moyen de MSC. Les cas non standards correspondant aux cas de fonctionnement exceptionnels peuvent alors être construits à partir des précédents. De ce fait, les MSC peuvent être utilisés pour la spécification des besoins, la spécification des interfaces, la simulation et la validation, la spécification des cas de tests et la documentation des systèmes temps réel. Un MSC peut être utilisé en relation avec d'autres langages de spécification, en particulier le SDL. Dans cette optique, les MSC sont une base pour la conception des systèmes SDL.

STATUTS/STABILITÉ

Le statut du MSC de base est quasi stable en ce qui concerne les constructions utilisées par une instance, la création et la terminaison d'instance, l'échange de messages, l'action, la gestion du temps et la condition. Pour les nouveaux concepts structurels – la corégion généralisée, l'expression en ligne, la référence MSC, le HMSC – des développements supplémentaires seront réalisés.

TRAVAIL EN RAPPORT

– Recommandation UIT-T Q.65 (1997), *Méthode fonctionnelle unifiée de caractérisation des services et des capacités des réseaux.*

– Recommandation UIT-T X.210 (1993), *Technologies de l'information – Interconnexion des systèmes ouverts – Modèle de référence de base: Conventions de définition des services de l'interconnexion des systèmes ouverts.* (Texte commun à l'UIT et à l'ISO/CEI.)

– Recommandation UIT-T Z.100 (1993), *Langage de description et de spécification du CCITT.*

Recommandation Z.120

DIAGRAMME DE SÉQUENCE DES MESSAGES

(révisée en 1996)

1 Introduction

1.1 Introduction au MSC

Un diagramme de séquence des messages (MSC, *message sequence chart*) représente des séquences de messages échangés entre les composants d'un système et leur environnement. En SDL (Recommandation Z.100, UIT 1996), les composants d'un système sont des services, des processus et des blocs. Les MSC sont depuis longtemps utilisés, d'une part, dans des Recommandations élaborées par des groupes d'études de l'UIT-T (ancien CCITT) et, d'autre part, dans l'industrie, selon diverses conventions et sous différents noms tels que diagramme de séquences de signaux, diagramme de flux d'information, flux de messages et diagramme de direction.

Normaliser les MSC permet de développer des outils de support, d'échanger des MSC entre différents outils, de rendre plus facile la correspondance avec les spécifications SDL et d'en harmoniser l'utilisation au sein de l'UIT.

Une partie du travail de normalisation est de fournir une définition claire d'un MSC. Ceci est fait dans la présente Recommandation [en Annexe B (1995)] par la définition d'une sémantique formelle basée sur des processus algébriques.

Une autre façon d'expliquer la signification des MSC peut être en les liant aux spécifications SDL de la manière suivante: un MSC décrit une ou plusieurs traces d'exécution de la spécification d'un système SDL. Selon cette définition, un MSC peut être dérivé de la spécification d'un système SDL et peut servir, par exemple, à mémoriser le résultat d'une exécution.

Grâce à la normalisation, l'importance des MSC pour les systèmes d'ingénierie a augmenté considérablement. Ainsi les MSC peuvent être utilisés pour:

a) décrire une vue d'ensemble du service offert par plusieurs entités;

b) formuler les besoins requis par des spécifications SDL;

c) servir de base à l'élaboration de spécifications SDL;

d) servir de base à la simulation et à la validation de système;

e) servir de base à la sélection et à la spécification de cas de tests;

f) spécifier une communication;

g) spécifier une interface;

h) formaliser des cas d'utilisation dans l'analyse et la conception orientée objet.

Comme un MSC décrit généralement un comportement partiel, la sélection de comportements partiels est un problème crucial. Tout d'abord, on utilise les MSC pour décrire des cas "standards". Puis des cas représentant des comportements exceptionnels, causés par des erreurs diverses, sont bâtis à partir de ceux-ci.

Dans la suite du document, une syntaxe pour les diagrammes de séquence des messages est présentée sous forme textuelle et graphique. Une description de la sémantique correspondante est fournie de manière informelle (en langage naturel).

La présente Recommandation est construite de la façon suivante: le paragraphe 2 donne des règles générales concernant la syntaxe, les règles applicables aux dessins et à la mise en page. Au paragraphe 3, la définition syntaxique d'un document de diagramme de séquence des messages qui est un ensemble de diagrammes de séquence des messages est fournie. Le paragraphe 4 contient la définition syntaxique d'un diagramme de séquence des messages et les règles syntaxiques des éléments de base, c'est-à-dire l'instance, le message, l'environnement, l'ordonnancement général, la condition, le temporisateur, l'action, la création et la terminaison d'une instance. Au paragraphe 5, des concepts de haut niveau concernant la structure et la modularité sont introduits. Ces concepts permettent de construire une spécification au moyen de la méthode de conception descendante (approche top down) en raffinant les instances individuelles à l'aide de la corégion (voir 5.1) et de la décomposition d'une instance (voir 5.2). Les concepts les plus avancés – expression en ligne (voir 5.3), la référence à un MSC (voir 5.4), le MSC de haut niveau (voir 5.5) – permettent de composer les MSC et de les réutiliser entièrement ou en partie à différents niveaux. Le paragraphe 6 donne des exemples d'utilisation de toutes les constructions MSC. L'Annexe A contient des références croisées pour les mots clés <keyword> et les non terminaux.

1.2 Métalangage pour la grammaire textuelle

Dans la forme de Backus-Naur (BNF, *Backus-Naur form*) un symbole terminal est soit un symbole non délimité par les signes inférieur et supérieur soit l'un des deux symboles <name> et <character string>. Notons que les deux symboles terminaux spéciaux <name> ou <character string> peuvent aussi avoir des sémantiques plus fortes dont un exemple est donné ci-dessous.

Les noms délimités par les signes inférieurs et supérieurs sont soit un symbole non terminal soit l'un des deux terminaux <character string> ou <name>. Les catégories syntaxiques sont les non terminaux indiqués par un ou plusieurs mots délimités par les signes inférieurs et supérieurs. Pour chaque symbole non terminal, une règle de production est fournie soit dans la grammaire textuelle concrète, soit dans la grammaire graphique concrète. Par exemple:

<msc inst interface> ::=
inst <instance list> <end>

Une règle de production pour les symboles non terminaux est composée d'un symbole non terminal en membre gauche du symbole **::=**, et d'une ou plusieurs constructions, composée(s) de symboles non terminaux et/ou terminaux en membre droit. Par exemple, <msc inst interface> et <instance list> dans l'exemple ci-dessus sont des symboles non terminaux; **inst** est un symbole terminal.

Parfois une partie du symbole est soulignée. Ceci souligne l'aspect sémantique de ce symbole, par exemple <msc name> est syntaxiquement identique à <name>, mais sémantiquement il faut que le nom soit un nom de diagramme de séquence des messages.

Le membre droit du symbole **::=** peut être composé de plusieurs alternatives séparées par une barre verticale (|). Par exemple:

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

signifie que <incomplete message area> est soit un <lost message area> soit un <found message area>.

Les éléments syntaxiques peuvent être groupés en utilisant les accolades ({ et }). Un groupe entre accolades peut contenir une ou plusieurs barres verticales indiquant le choix entre les éléments syntaxiques. Par exemple:

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

La répétition des groupes entre accolades est indiquée par un astérisque (*) ou le signe plus (+). Un astérisque indique que le groupe est optionnel et peut être répété un nombre quelconque de fois; un signe plus indique que le groupe est obligatoire et peut être répété un nombre quelconque de fois. L'exemple ci-dessus exprime qu'un <msc document body> peut être vide mais peut aussi contenir un certain nombre de <message sequence chart>s et/ou de <msc diagram>s.

Si des éléments syntaxiques sont groupés à l'aide de crochets ([et]), alors le groupe est optionnel. Par exemple:

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

exprime qu'un <identifier> peut, mais pas nécessairement, contenir un <qualifier>.

1.3 Métalangage pour la grammaire graphique

Peu de constructions constituent le métalangage de la grammaire graphique. Elles sont décrites de manière informelle ci-dessous. La syntaxe graphique n'est pas assez précise pour décrire les éléments graphiques. C'est pour cela qu'il n'y a pas de variations graphiques. De petites variations dans les formes réelles des symboles pour terminaux graphiques sont autorisées. Il s'agira, par exemple, du hachurage des symboles pleins, de la forme d'une pointe de flèche et de la taille relative d'éléments graphiques. Chaque fois que cela sera nécessaire, la syntaxe graphique sera complétée d'une explication informelle sur les représentations graphiques des constructions.

Le métalangage consiste en une sorte de notation BNF composée de métaconstructions spéciales: ***contains***, ***is followed by***, ***is associated with***, ***is attached to, above*** et ***set.*** Ces constructions se comportent comme les règles de production BNF normales, mais impliquent en plus des relations logiques ou géométriques entre les arguments. Le comportement de la construction ***is attached to***, expliqué ci-dessous, est un peu différent. Le membre gauche de toutes les constructions doit être un symbole sauf ***above***. Un symbole est un non terminal produisant après plusieurs séquences de production exactement un terminal graphique. Nous considérerons un symbole qui **est attaché à** (***is attached to***) d'autres zones ou qui **est associé à** (***is associated with***) un texte également comme un symbole. L'explication est informelle et le métalangage ne décrit pas avec précision les dépendances géométriques.

contains:

"<area1> ***contains*** <area2>" signifie que <area2> est contenu dans <area1>, géométriquement, mais aussi logiquement. Voir Figure 1-1.

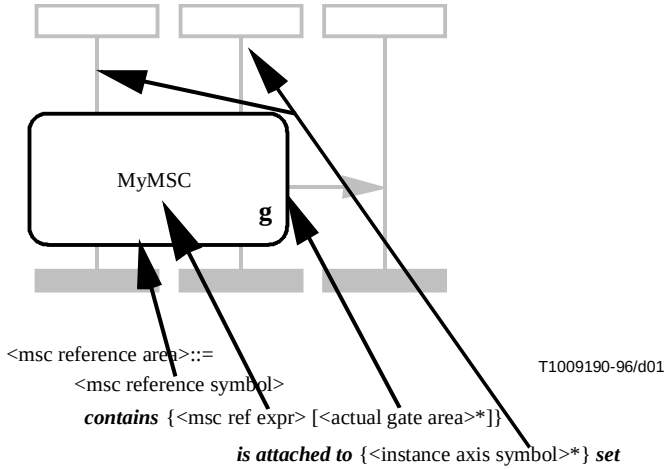


Figure 1-1/Z.120 – Exemple pour "contains"

is followed by:

"<area1> ***is followed by*** <area2>" signifie que <area2> est logiquement et géométriquement rattaché à <area1>. Voir Figure 1-2.

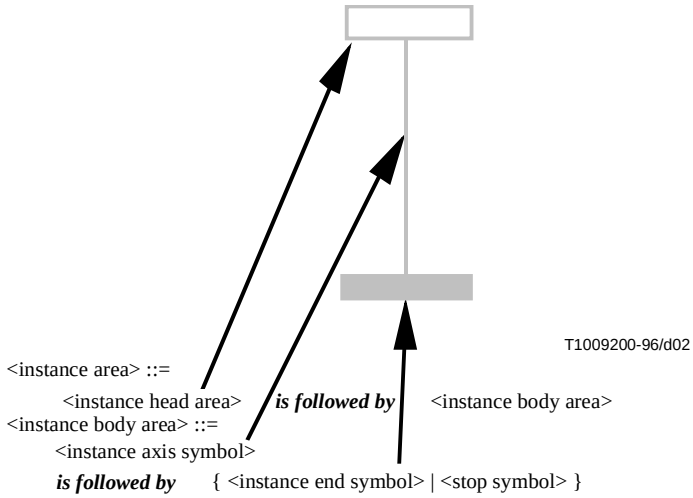


Figure 1-2/Z.120 – Exemple pour "is followed by"

is associated with:

"<area1> ***is associated with*** <area2>" signifie que <area2> est un texte affilié à <area1>. Il n'y a pas d'association géométrique plus étroite entre les zones que lorsqu'elles sont associées l'une avec l'autre. Voir Figure 1-3.

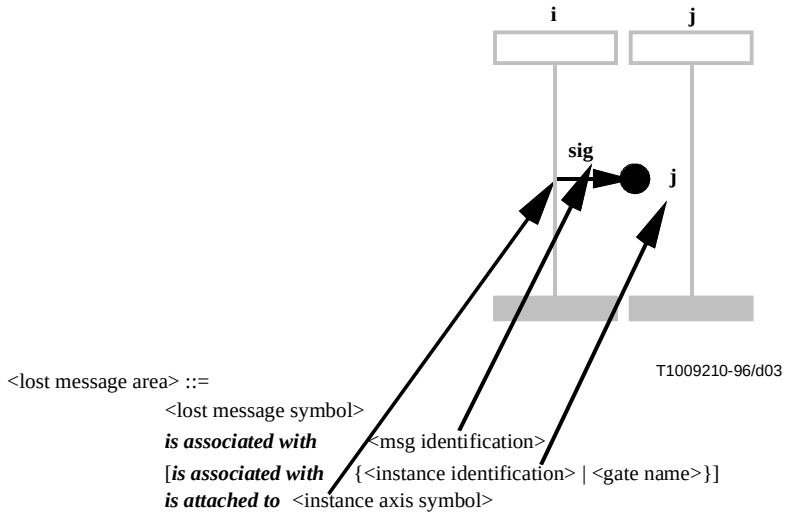
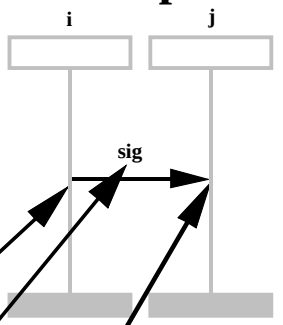


Figure 1-3/Z.120 – Exemple pour "is associated with"

is attached to:

"<area1> ***is attached to*** <area2>" n'est pas une règle de production BNF normale et son utilisation est restreinte. <area2> doit être un symbole ou un ensemble de symboles. La règle de production "<P>::=<area1> ***is attached to*** <area2>" signifie que, si le symbole non terminal <P> est dérivé en utilisant cette règle, seul le symbole <area1> est produit. Plutôt que de produire également <area2>, une occurrence de <area2> est identifiée comme ayant été produite par une autre règle. Le résultat de la construction ***is attached to*** est une relation logique et géométrique entre <area1> et <area2>. Si le membre droit est un ensemble de symboles, il y a une relation entre <area1> et tous les éléments de l'ensemble. Voir Figure 1-4.



<message area> ::=
<message symbol>
is associated with <msg identification>
is attached to { <message start area> <message end area> }

-- *Notons que si le symbole A* ***est attaché au*** *symbole B, alors le symbole B* ***est*** *aussi* ***attaché au*** *symbol A.*
Dorénavant, un attachement entre deux symboles est toujours défini deux fois dans la grammaire.
-- *Nous suivons l'un des non terminaux ci-dessus pour obtenir ceci:*
<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>

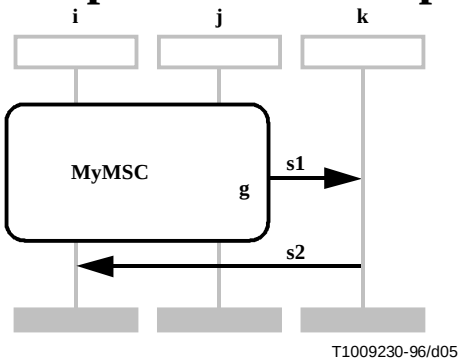
<message in symbol> ::=
<void symbol>

Ainsi un symbole <message symbol> e***st attaché à*** un symbole <message in symbol> et vice versa.

Figure 1-4/Z.120 – Exemple pour "is attached to"

above:

"<area1> ***above*** <area2>" signifie que <area1> et <area2> sont logiquement ordonnés. Géométriquement on exprime cette idée en ordonnant verticalement <area1> et <area2>. Si deux <area>s ont les mêmes coordonnées verticales alors aucun ordre entre eux n'est défini, mis à part le fait qu'un envoi de message doit avoir lieu avant une réception. Voir Figure 1-5.



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

La figure décrit une séquence d'événements. Lorsque les événements sont sur des instances différentes, la coordonnée géométrique verticale n'a aucune signification sémantique.

La figure ci-dessus décrit la séquence suivante:
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.

Figure 1-5/Z.120 – Exemple pour "above"

set:

Le métasymbole ***set*** est un opérateur postfixé portant sur les éléments syntaxiques entre accolades se trouvant immédiatement avant, et indiquant un ensemble (non ordonné) d'éléments. Chaque élément peut être un groupe d'éléments syntaxiques, auquel cas il doit être développé avant l'application du métasymbole ***set.***

Exemples:

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

est un ensemble de 0 ou plusieurs symboles <text area>s.

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

est un ensemble non ordonné des éléments entre accolades.

2 Règles générales

2.1 Règles lexicales

Les règles lexicales définissent des unités lexicales. Les unités lexicales sont des symboles terminaux de la *syntaxe textuelle concrète*.

<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> ::= }

<overline> ::= ~

<upward arrow head> ::= ^

<full stop> ::= .

<underline> ::= _

<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> ::= << | >>

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

<name> ::= <word> { <underline> <word> }*

<keyword> ::= **action**
| **all**
| **alt**
| **as**
| **before**
| **begin**
| **block**
| **by**
| **comment**
| **concurrent**
| **condition**
| **connect**
| **create**
| **decomposed**
| **empty**
| **end**
| **endconcurrent**
| **endinstance**
| **endmsc**
| **endexpr**
| **env**
| **exc**
| **expr**
| **external**
| **found**
| **from**
| **gate**
| **in**
| **inf**
| **inline**
| **inst**
| **instance**
| **loop**
| **lost**
| **msc**
| **mscdocument**
| **msg**
| **opt**
| **order**
| **out**
| **par**
| **process**
| **reference**
| **related**
| **reset**
| **service**
| **seq**
| **set**
| **shared**
| **stop**
| **subst**
| **system**

| **text**
| **timeout**
| **to**
| **via**

Les caractères <national> sont représentés ci-dessus de la même façon que la version internationale de référence de l'Alphabet n° 5 du CCITT (Recommandation T.50). La responsabilité de la définition des représentations nationales de ces caractères relève des organismes nationaux de normalisation.

Les caractères de contrôle sont définis conformément à la Recommandation T.50. Une séquence de caractères de contrôle peut apparaître là où un espace <space> peut apparaître, et a la même signification qu'un espace <space>. L'espace <space> représente le caractère espace de l'Alphabet n° 5 du CCITT.

L'occurrence d'un caractère de contrôle n'est pas significative dans une <note>. Un caractère de contrôle ne peut pas apparaître dans une chaîne de caractères si sa présence est significative.

Dans tous les <lexical unit>s à l'exception de <character string>, les <letter>s sont toujours traitées comme des majuscules. (Le traitement des caractères <national>s peut être défini par les organismes nationaux de normalisation.)

Une unité <lexical unit> se termine par le premier caractère qui ne peut pas faire partie de l'unité <lexical unit> conformément à la syntaxe spécifiée ci-dessous. Lorsqu'un caractère souligné <underline> est suivi par un ou plusieurs caractères <space>, tous ces caractères (y compris le <underline>) sont ignorés, par exemple A_B définit le même nom <name> AB. Cette utilisation de <underline> permet de répartir les <lexical unit>s sur plus d'une ligne.

Le caractère / immédiatement suivi par le caractère * en dehors d'une <note> marque le début d'une <note>. Le caractère * immédiatement suivi par le caractère / dans une <note> termine toujours la <note>. Une <note> peut être insérée avant ou après toute unité <lexical unit>.

2.2 Règles de visibilité et de nommage

Les entités sont identifiées et référencées au moyen des noms correspondants. Les entités sont groupées par classes d'entités pour permettre une plus grande souplesse des règles de nommage. Les classes d'entités sont les suivantes:

a) MSC document (document MSC);
b) MSC (MSC);
c) instance (instance);
d) condition (condition);
e) timer (temporisateur);
f) message (message);
g) gate (porte).

L'unité de portée pour MSC est le document MSC. Deux entités au sein d'une unité de portée qui appartiennent à la même classe d'entités ne peuvent avoir le même nom. Différentes occurrences d'un nom de condition, d'un nom de temporisateur et d'un nom de message désignent la même entité. Le nom d'une entité est visible par une unité de portée englobante mais pas à l'extérieur. Seuls les noms visibles peuvent être utilisés dans le référençage des entités.

Les noms des portes sont toujours associés à un MSC spécifique. Par conséquent les mêmes noms de portes peuvent être appliqués à des MSC différents.

2.3 Commentaire

Il existe trois sortes de commentaires.

Tout d'abord, la *note* se rencontre uniquement dans la syntaxe textuelle (voir les règles lexicales).

La deuxième forme de commentaire est *comment*. Il s'agit d'une notation pour représenter les explications informelles associées aux symboles ou au texte.

La *syntaxe textuelle concrète* des commentaires est:

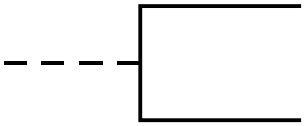
<end> ::= [<comment>]**;**

<comment> ::= **comment** <character string>

Dans la *grammaire graphique concrète*, la syntaxe suivante est utilisée:

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

<comment symbol> ::=



T1009240-96/d06

Un symbole <comment symbol> peut être attaché aux symboles graphiques suivants: <text symbol>, <instance head symbol>, <instance end symbol>, <message out symbol>, <message in symbol>, <lost message symbol>, <found message symbol>, <general order symbol>, <condition symbol>, <set symbol>, <reset symbol>, <timeout symbol>, <action symbol>, <createline symbol>, <stop symbol>, <coregion symbol>, <inline expression symbol>, <separator symbol>, <msc reference symbol>, <hmsc start symbol>, <hmsc end symbol>, <par frame symbol>, <connection point symbol>.

La troisième forme de commentaire est le *text* et peut être utilisée pour des commentaires au niveau global.

Text dans la *grammaire textuelle* est défini par:

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

Dans la *grammaire graphique* text est défini par:

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

<text symbol> ::=

T1009250-96/d07

2.4 Règles applicables aux dessins

La taille des symboles graphiques peut être choisie par l'utilisateur. Les frontières des symboles ne doivent pas se chevaucher ni se croiser. Les exceptions à cette règle sont les suivantes:

a) un symbole de message et un symbole d'ordonnancement général peuvent croiser un symbole de message, un symbole d'ordonnancement général, les lignes des symboles de temporisateur, un symbole de création, un symbole d'axe d'instance, une ligne pointillée d'un symbole de commentaire et un symbole de condition;

b) les lignes des symboles de temporisateur peuvent croiser un symbole de message, un symbole d'ordonnancement général, les lignes des symboles de temporisateur, un symbole de création, une ligne pointillée d'un symbole de commentaire et un symbole de condition;

c) un symbole de création peut croiser un symbole de message, un symbole d'ordonnancement général, les lignes des symboles de temporisateur, un symbole d'axe d'instance et une ligne pointillée d'un symbole de commentaire;

d) un symbole de condition peut croiser un symbole d'axe d'instance, un symbole de message, un symbole d'ordonnancement général, les lignes des symboles de temporisateur;

e) un symbole d'expression en ligne peut croiser un symbole d'axe d'instance;

f) un symbole de référence peut croiser un symbole d'axe d'instance;

g) un symbole d'action peut croiser un symbole d'axe d'instance dans sa forme linéaire et chevaucher un symbole d'axe d'instance dans sa forme colonne;

h) une ligne d'un diagramme hmsc peut croiser une autre ligne du diagramme hmsc.

Il existe deux formats pour représenter le symbole d'axe d'instance et le symbole de corégion: le format ligne simple et le format colonne. Il est interdit d'utiliser les deux formats au sein d'une même instance, excepté l'axe linéaire avec la forme colonne des corégions.

Si une condition partagée (voir 4.6) croise une instance non concernée par cette condition, le symbole d'axe d'instance la traverse.

Si une référence partagée (voir 5.4) croise une instance non concernée par cette référence, l'axe de l'instance la traverse.

Si une expression en ligne partagée (voir 5.3) croise une instance non concernée par cette expression en ligne, l'axe de l'instance ne doit pas la traverser.

Dans le cas où le symbole d'axe d'instance est au format colonne, les frontières verticales du symbole d'action doivent coïncider avec les lignes de la colonne.

Les lignes représentant les messages peuvent être horizontales ou obliques dirigées vers le bas (conformément à la direction donnée par la flèche), et doivent être des segments de lignes droites connectés en séquence.

Si un événement entrant et un événement sortant se trouvent sur le même point d'un axe d'instance, ceci est interprété comme si l'événement entrant était dessiné en dessous de l'événement sortant. Un symbole d'ordonnancement général ne peut pas être attaché à ce point. Deux ou plusieurs événements sortants ne doivent pas être dessinés sur le même point. Deux ou plusieurs événements entrants ne doivent pas être dessinés sur le même point. Les événements suivants sont des événements entrants: une réception de message, un message spontané et une expiration d'un temporisateur. Les événements suivants sont des événements sortants: un envoi de message, une perte de message, un armement de temporisateur, un désarmement de temporisateur et une création d'instance.

2.5 Règles d'impression des MSC

Les MSC peuvent être répartis sur plusieurs pages. La répartition peut être à la fois horizontale et verticale, circulaire au moyen d'une composition de MSC ou d'une décomposition d'instance (voir paragraphe 5).

Quand un MSC est réparti sur plusieurs pages, le <msc heading> est répété sur chaque page, mais les symboles de fin d'instance ne doivent apparaître que sur une page (sur la "dernière page" pour l'instance en question). Pour chaque instance, la <instance head area> doit apparaître sur la première page où l'instance en question commence et doit être répétée en forme pointillée sur toutes les pages suivantes où elle continue.

Si des messages, des temporisateurs, des instructions de création ou des conditions doivent être représentés d'une page à la suivante, le nom du message, du temporisateur, de la création ou de la condition doit apparaître entièrement sur la première page et tout ou une partie du texte peut être répété sur la page suivante.

La numérotation des pages doit être indiquée sur les pages d'un MSC afin de mentionner la séquence correcte des pages. Les pages doivent être numérotées par paire: "v-h" où "v" est le numéro de la page verticale et "h" le numéro de la page horizontale. Les nombres arabes doivent être utilisés pour la numérotation verticale et les lettres majuscules ('A' à 'Z') pour la numérotation horizontale. Si le rang 'A'-'Z' n'est pas suffisant, alors il peut être étendu avec 'AA' à 'AZ', 'BA' à 'BZ', etc.

3 Document de diagramme de séquence des messages

L'en-tête d'un document de diagramme de séquence des messages contient le nom du document éventuellement suivi du mot clé **related to**, puis de l'identificateur (nom du chemin d'accès) du document SDL auquel les MSC se réfèrent.

Grammaire concrète

<msc document> ::=

<msc document head> <msc document body>

<msc document head> ::=
<document head> | <document head area>

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

Grammaire textuelle concrète

<document head> ::=
mscdocument <msc document name> [**related to** <sdl reference>] <end>

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

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

<qualifier> ::= **<<** <text> **>>**

Le texte <text> dans un qualificateur <qualifier> ne doit pas contenir l'un des caractères '<<' ou '>>'.

Grammaire graphique concrète

<document head area> ::=
<frame symbol> ***contains*** <document head>

Sémantique

Un document de diagramme de séquence des messages est un ensemble de diagrammes de séquence des messages, faisant éventuellement référence à un document SDL correspondant.

4 Le MSC de base

4.1 Diagramme de séquence des messages

Un diagramme de séquence des messages, dont l'abréviation est MSC, décrit le flux des messages entre instances. Un diagramme de séquence des messages décrit un comportement partiel d'un système. Bien que le nom *message sequence chart* ait pour origine, de manière évidente, sa représentation graphique, celui-ci est utilisé pour désigner à la fois les représentations textuelle et graphique.

L'en-tête d'un diagramme de séquence des messages est constitué d'un nom de diagramme de séquence des messages et (optionnellement) d'une liste d'instances contenues dans le corps d'un MSC et d'une liste de portes (gates) du MSC.

Un MSC est décrit soit par des instances et des événements, soit par une expression se rapportant aux références MSC ne nommant pas explicitement les instances [voir 5.5, MSC de haut niveau (HMSC, *high-level MSC*)].

Grammaire concrète textuelle

<message sequence chart> ::=
msc <msc head> { <msc body> | **expr** <msc expression> } **endmsc** <end>

<msc head> ::= <msc name> <end> [<msc interface>]

<msc interface> ::= [<msc inst interface>] [<msc gate interface>]

<msc inst interface> ::=
inst <instance list> <end>

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

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

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

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

<msc gate def> ::= **gate** { <msg gate> | <order gate> } <end>

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

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

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

<msc statement> ::=
<text definition> | <event definition> | <old instance head statement> <instance event list>

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

<instance event list> ::=
{ <instance event> <end> }+

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

<orderable event> ::=
[<event name>]
{ <message event> | <incomplete message event> | <create> | <timer statement> | <action> }
[**before** <event name list>]

Le nom optionnel <event name> est utilisé lorsque l'événement est généralement ordonné.

<event name list> ::=
{ <event name> | <gate name> } [, <event name list>]

Le nom <gate name> fait référence à <def order out gate>, <actual order in gate>, <inline order out gate> ou <def order out gate>. Le nom <event name> fait référence à un événement <orderable event>.

<non-orderable event> ::=
<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> <end> }+

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

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

La règle non terminale <old instance head statement> n'est présente que pour la compatibilité ascendante avec la norme MSC-92 existante.

Pour chaque instruction <instance head statement>, ou <old instance head statement>, il doit également exister l'instruction de fin correspondante <instance end statement> ou un événement <stop>. Pour chaque instance, aucun événement avant sa déclaration par <instance head statement> ne doit être spécifié. Pour chaque instance, aucun événement après son instruction de fin <instance end statement> ne doit être spécifié. Pour chaque instance, il ne doit exister qu'une seule définition d'en-tête <instance head statement> et qu'une seule définition de fin <instance end statement>. Pour chaque instance, aucune instruction <instance head statement> ne doit apparaître après une instruction <old instance head statement>.

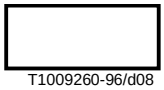
Si la liste d'instances <instance list> est présente dans la zone de définition <msc interface>, elle doit contenir les mêmes instances spécifiées dans le corps du diagramme <msc body>.

Grammaire graphique concrète

<msc diagram> ::= <msc symbol> ***contains*** { <msc heading> { <msc body area> | <mscexpr area> } }

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

<frame symbol> ::=



T1009260-96/d08

<msc heading> ::= **msc** <msc name>

<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>* } ***set***

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

<instance event area> ::=
<message event area>
| <timer area>
| <concurrent area>
| <action area>

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

Sémantique

Un MSC décrit la communication entre des composants d'un système, et entre ces composants et le reste du monde, appelé environnement. Pour chaque composant de système décrit par un MSC il existe un axe d'instance. La communication entre les composants d'un système est assurée au moyen de messages. L'envoi et la consommation de messages sont deux événements asynchrones. On suppose que l'environnement d'un MSC est capable de recevoir des messages du diagramme de séquence des messages et d'en envoyer à celui-ci; il n'y a pas d'ordre au niveau des événements liés aux messages dans l'environnement. Bien que le comportement de l'environnement soit non déterministe, on suppose qu'il obéit aux contraintes exprimées dans le diagramme de séquence des messages.

Un diagramme de séquence des messages ne comporte pas d'axe de temps global. Le long de chaque axe d'instance, le temps se déroule du haut vers le bas; cependant, il n'existe pas d'échelle des temps précise. S'il n'existe aucune corégion ou expression en ligne (voir 5.1, 5.3), les événements sont totalement ordonnés dans le temps, le long de chaque axe d'instance. Les événements des différentes instances sont uniquement ordonnés à travers les messages: un message doit d'abord être envoyé avant d'être consommé – ou par ce qu'on appelle le mécanisme d'ordonnancement général. Avec ce mécanisme d'ordonnancement général, "les événements ordonnés" sur différentes instances (même dans différents MSC) peuvent être ordonnés de façon explicite. Aucun autre ordre n'est imposé. Par conséquent, un diagramme de séquence des messages impose un ordre partiel sur l'ensemble des événements qu'il contient. Une relation binaire transitive, antisymétrique et réflexive est appelée ordre partiel.

Par exemple, pour les messages en entrée du diagramme de séquence des messages de la Figure 4-1 a) [étiquetés par in(mi)] et ceux en sortie [étiquetés par out(mi)], nous avons la relation d'ordre suivante: out(m2) < in(m2), out(m3) < in(m3), out(m4) < in(m4), in(m1) < out(m2) < out(m3) < in(m4), in(m2) < out(m4) et sa fermeture transitive.

L'ordre partiel peut être décrit en forme minimale (sans représentation explicite de la fermeture transitive) au moyen de son graphe de connectivité [voir Figure 4-1 b)].

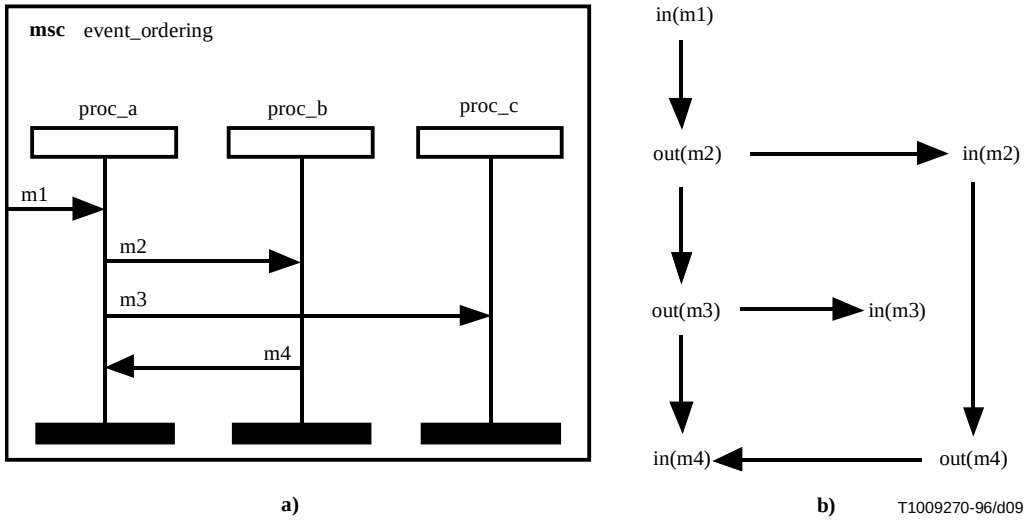


Figure 4-1/Z.120 – Diagramme de séquence des messages et son graphe de connectivité

Une sémantique formelle des MSC basée sur un processus algébrique est fournie en Annexe B (1995). La sémantique d'un MSC peut être apparentée à la sémantique du SDL par la notion de graphe d'accessibilité. Chaque séquencialisation d'un MSC décrit une trace d'un nœud à un autre nœud (ou à un ensemble de nœuds) du graphe d'accessibilité décrivant le comportement d'une spécification d'un système SDL. Un graphe d'accessibilité est constitué de nœuds et d'arcs. Les nœuds représentent les états globaux du système. L'état global d'un système est déterminé par les valeurs des variables et l'état d'exécution de chaque processus et les contenus des files de messages. Les arcs correspondent aux événements exécutés par le système, par exemple l'envoi et la consommation d'un message ou l'exécution d'une tâche. La séquencialisation d'un MSC représente un ordre total d'événements compatible avec l'ordre partiel défini par le MSC.

Dans la représentation textuelle, la définition <event definition> est fournie soit par un nom <instance name> suivi de la liste attachée <event list> soit par la liste <instance name list> suivie des événements attachés <multi instance event> séparés par une virgule. Le non terminal <instance event> représente soit un événement attaché à une instance simple, par exemple une <action> ou un objet partagé, comme une condition partagée <shared condition> où le mot clé **shared** suivi de la liste des instances ou du mot clé **all** est utilisé pour définir l'ensemble des instances partageant la condition. Un objet partagé peut être représenté alternativement par des événements <multi instance event>. Les notations différentes sont introduites pour faciliter une description *orientée instance*, d'une part, et une description *orientée événement*, d'autre part. Ces deux notations peuvent être utilisées ensemble. La description *orientée instance* énumère les événements associés à une instance. Avec la représentation textuelle de la description *orientée événement*, les événements peuvent être énumérés sous forme d'une trace d'exécution possible et ne sont pas ordonnés par rapport aux instances.

La déclaration optionnelle <msc interface> décrivant l'interface d'un MSC avec son environnement consiste en un <msc inst interface> et un <msc gate interface>. Le <msc inst interface> fournit une déclaration des instances, c'est-à-dire un nom <instance name> et éventuellement un type <instance kind>. Puisqu'un MSC ne représente que l'exécution d'une partie d'un système, le <msc inst interface> décrit les points de connexion des instances à l'environnement. <msc gate interface> fournit une définition des portes de message et des portes d'ordre contenues dans le MSC. Les "portes" associées aux messages définissent les points de connexion des messages à l'environnement. Eventuellement, des noms de porte peuvent être associés aux portes.

4.2 Instance

Un diagramme de séquence des messages est composé d'instances d'entités qui interagissent. L'instance d'une entité est un objet qui a les propriétés de cette entité. Si on se réfère au SDL, une entité peut être un processus SDL, un bloc ou un service. Au niveau de l'en-tête de l'instance, le nom de l'entité, par exemple le nom du processus, peut être ajouté au nom de l'instance. Au niveau du corps de l'instance, l'ordre des événements est spécifié.

Grammaire textuelle concrète

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

<instance end statement> ::=
endinstance

Grammaire graphique concrète

<instance 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>]

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

<instance head symbol> ::=

<instance body area> ::=

<instance axis symbol>

is followed by { <instance end symbol> | <stop symbol> }

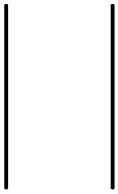
<instance axis symbol> ::=

{ <instance axis symbol1> | <instance axis symbol2> }

is attached to { <event area> * } ***set***

<instance axis symbol1> ::=

<instance axis symbol2> ::=



<instance end symbol> ::=

T1009280-96/d10

L'<instance heading> peut être placée au-dessus ou à l'intérieur du symbole <instance head symbol>, ou bien fractionnée de telle sorte que le nom <instance name> soit placé à l'intérieur du symbole <instance head symbol> lorsque le type <instance kind> ou la <decomposition> sont au-dessus. Dans ce dernier cas, le symbole en colonne est facultatif (et sera placé au-dessus s'il est utilisé).

Sémantique

Les instances sont définies dans le corps du diagramme de séquence des messages. Le symbole de fin d'instance correspond à la fin de la description de l'instance au sein du MSC. Il ne correspond pas à la terminaison de l'instance (voir 4.10, Terminaison d'instance). De même, le symbole de début d'instance correspond au commencement de la description de l'instance au sein du MSC. Il ne correspond pas à la création de l'instance (voir 4.9, Création d'instance).

Dans le cadre du SDL, une instance peut se rapporter à un processus (mot clé **process**), un service (mot clé **service**) ou un bloc (mot clé **block**). En dehors de ce cadre, une instance peut se rapporter à toute sorte d'entité. La définition d'instance fournit, en termes d'événements, une description pour la consommation et l'émission de messages, les actions, les conditions locales et partagées, les temporisations, la création d'instances et la terminaison d'instances. Mise à part l'utilisation de corégion (voir 5.1) et des expressions en ligne (voir 5.2), on a un ordre total des événements le long de chaque axe d'instance. Au sein des corégions, on ne garantit pas un ordre temporel des événements si aucune autre construction de synchronisation sous forme de relations d'ordre général n'est spécifiée.

4.3 Message

Un message dans un MSC est une relation entre un envoi et une réception. L'envoi peut provenir soit de l'environnement (à travers une porte) soit d'une instance, et une réception est destinée soit à l'environnement (une porte) soit à une instance.

Si l'envoi ou la réception est perdu, le message est incomplet. Dans ce cas il n'est représenté que par un seul événement.

Un message échangé entre deux instances peut être décomposé en deux événements: le message en entrée et le message en sortie; par exemple, le deuxième message de la Figure 4-1 a) peut être décomposé en out(m2) (sortie) et in(m2) (entrée). Dans la représentation textuelle, le message en entrée est représenté par le mot clé **in**, le message en sortie par le mot clé **out**, tous les deux suivis par le nom du message et éventuellement par un nom d'instance de message. Un message peut véhiculer des paramètres; ceux-ci sont représentés entre parenthèses.

La correspondance entre les messages en sortie et les messages en entrée doit être biunivoque. Dans la représentation textuelle, la correspondance entre les messages en entrée et les messages en sortie, se fait, en principe, par identification du nom de message et de l'adresse spécifiée. Dans la représentation graphique, un message est représenté par une flèche.

La perte d'un message, c'est-à-dire si un message est envoyé mais n'est pas consommé, est indiquée par le mot clé **lost** dans la représentation textuelle et par un "trou noir" dans la représentation graphique.

Inversement, un message spontané, c'est-à-dire un message dont l'origine est inconnue, est défini par le mot clé **found** dans la représentation textuelle et par un "trou blanc" dans la représentation graphique.

Dans la représentation textuelle, le mot clé **before** permet de définir un ordre temporel pour les messages sur les instances différentes. Dans la représentation graphique, ces concepts d'ordre généralisé sont définis à l'aide des constructions de synchronisation sous forme de lignes de connexion.

Grammaire textuelle concrète

<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>)]

<parameter list> ::=
<parameter name> [**,** <parameter list>]

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

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

Le nom <gate name> fait référence à une définition <def in gate>. Si seul le mot clé **env** est utilisé alors l'adresse <output address> désigne une définition <def in gate> avec un nom implicite (construit à l'aide de l'identification du message <msg identification> correspondante et de la direction **in**).

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

Le nom <gate name> fait référence à une définition <def out gate>. Si seul le mot clé **env** est utilisé alors l'adresse <input address> désigne une définition <def out gate> avec un nom implicite (construit à l'aide de l'identification du message <msg identification> correspondante et de la direction **out**).

Pour les messages échangés entre les instances, les règles suivantes doivent être appliquées: pour chaque <message output> un <message input> correspondant doit être spécifié et vice versa. Si le nom du message <message name> et l'adresse <address> ne suffisent pas pour une correspondance biunivoque, le nom d'instance de message <message instance name> doit être précisé.

Le <message output> ne doit pas dépendre de son <message input> par le biais d'autres messages ou de constructions d'ordre général. Ceci est le cas lorsque le graphe de connectivité (voir 4.1) contient des boucles. Si une liste de paramètres <parameter list> est spécifiée pour un <message input>, cette liste doit également être spécifiée pour le <message output> correspondant et vice versa. Les listes de paramètres <parameter list> doivent être identiques.

Grammaire graphique concrète

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

Les messages peuvent être ordonnés à l'aide de plusieurs relations d'ordre général différentes. Les messages apparaissent sur l'un des côtés de la relation d'ordre.

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

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

<void symbol> ::= .

Le symbole <void symbol> est un point géométrique fixe.

NOTE 1 – Le symbole <message out symbol> n'est en fait qu'un point sur l'axe de l'instance. La fin du symbole du message ne possédant pas la pointe de la flèche est aussi sur ce point de l'axe.

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

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

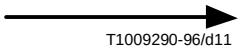
NOTE 2 – Le symbole <message in symbol> n'est en fait qu'un point sur l'axe de l'instance. La fin du symbole du message possédant la pointe de la flèche est aussi dirigée sur ce point de l'axe de l'instance.

<message area> ::= <message symbol> ***is associated with*** <msg identification>
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> ::=



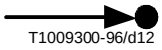
NOTE 3 – L'image miroir du <message symbol> est autorisée. La pointe de la flèche devra être sur l'axe de l'instance.

NOTE 4 – Dans la représentation graphique, le nom d'instance de message n'est pas nécessaire pour une description biunivoque.

<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> ::=

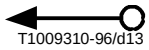


NOTE 5 – Le symbole représentant la perte de message <lost message symbol> décrit la partie correspondant à l'envoi: le trait plein part de la zone <message start area> correspondant au début de l'événement. La cible d'un message est optionnelle et peut être désignée par un identificateur associé au symbole. L'identification de la cible pourrait être écrite près du cercle noir, et l'identification du message près de la flèche.

NOTE 6 – L'image miroir du symbole est autorisée.

<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> ::=



NOTE 7 – Le symbole représentant le message spontané <found message symbol> décrit l'événement correspondant à la partie réception (la tête de la flèche) qui devrait être sur une zone <message end area>. L'instance ou la porte supposée être à l'origine du message est indiquée par l'identification associée au cercle du symbole; cette identification est optionnelle. L'identification du message pourrait être écrite près de la flèche.

NOTE 8 – L'image miroir du symbole est autorisée.

Sémantique

Pour un MSC, le message en sortie indique un envoi de message (correspondant à une sortie SDL) et le message en entrée indique une consommation de message (correspondant à une entrée SDL). Aucune construction spéciale n'est fournie pour la réception de message (entrée dans la file). Aucune définition de type n'est associée aux paramètres de la liste des paramètres.

Un message incomplet est un message qui est soit un envoi (pour lequel la réception est perdue) soit une réception (pour laquelle l'origine est inconnue).

Si des messages coïncident avec d'autres événements, se reporter aux règles de dessin du sous-paragraphe 2.4.

4.4 Environnement et les portes

Les portes représentent l'interface entre le MSC et son environnement. Un message ou une relation d'ordre attaché au cadre du MSC constitue une porte.

Une "porte de message" a toujours un nom. Celui-ci peut être défini explicitement par un nom associé à la porte sur le cadre. Sinon il est implicite et est construit à l'aide de la direction que le message prend à travers la porte et son nom; par exemple "in_X" représente une porte recevant un message X de son environnement.

Les "portes de message" sont utilisées lorsque les références à un MSC étendent le contexte à un autre MSC. Les portes réelles sur la référence au MSC sont alors connectées à d'autres "portes de message" ou à d'autres instances. De même que pour leurs définitions, les portes réelles peuvent avoir des noms explicites ou implicites.

Les "portes d'ordre" représentent des relations d'ordre incomplètes où un événement du MSC sera ordonné par rapport à un événement de l'environnement. Les "portes d'ordre" sont toujours nommées explicitement. Les "portes d'ordre" ont une direction allant de l'événement ordonné à l'événement lui succédant.

Les "portes d'ordre" sont également utilisées sur les références de MSC dans d'autres MSC. Les "portes d'ordre" réelles d'une référence MSC sont connectées à d'autres portes d'ordre ou à des événements.

Les portes dans les expressions en ligne sont identiques aux portes sur les cadres des MSC et des références MSC. La seule différence est que le cadre de l'expression en ligne est à la fois le cadre de la définition de la porte (à l'intérieur) et le symbole de l'utilisation de la porte réelle (à l'extérieur).

Les portes qui ne sont pas connectées à une référence MSC sont implicitement définies pour se propager au prochain cadre dans le diagramme (soit un cadre MSC soit un cadre d'une expression en ligne). Une porte propagée aura le même nom que la porte origine. Ceci est un raccourci pour éviter une ligne de connexion qui encombrerait le diagramme.

Les portes dans les expressions en ligne ne sont que des points de transfert sur le cadre de l'expression en ligne. Si la porte ne se prolonge pas hors du cadre, les règles implicites suivantes sont appliquées:

1) si plusieurs portes ont le même nom dans la même expression en ligne, le prolongement spécifié pour l'une des portes est valable pour toutes les autres;
2) si aucune des portes n'a le même nom et qu'aucun prolongement n'existe, on considère qu'il existe un prolongement implicite jusqu'au prochain cadre du diagramme (soit un cadre MSC, soit un cadre d'expression en ligne). Voir Figure 4-2.

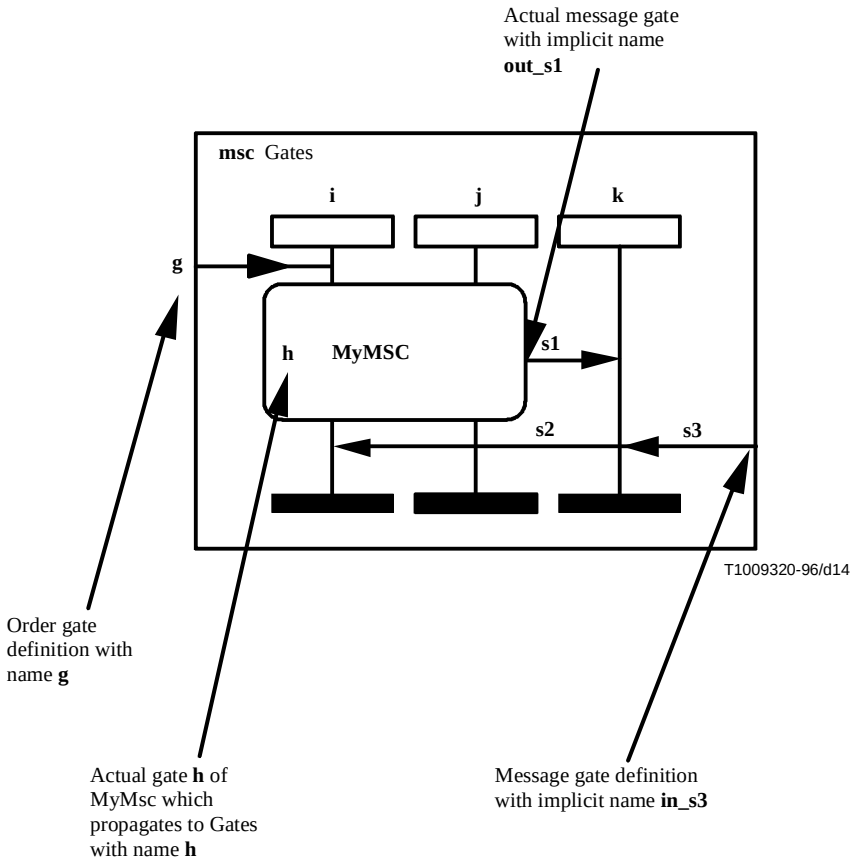


Figure 4-2/Z.120 – Exemple de portes

Grammaire textuelle concrète

<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 un nom de porte <gate name> n'est pas spécifié, un nom implicite est construit à l'aide de la direction et de l'identificateur <msg identification> correspondant.

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

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

Le nom <event name> fait référence à un événement ordonné. Le nom <gate name> fait référence à l'un des ordres suivants <def order out gate>, <actual order in gate>, <inline order out gate> ou <inline order in gate>.

<actual order in gate> ::=
<gate name>

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

Le premier nom <gate name> définit le nom de cette "porte d'ordre".

<def order out gate> ::=
<gate name>

Le nom <gate name> définit le nom de cette "porte d'ordre".

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

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

Grammaire graphique concrète

<inline gate area> ::=
{ <inline out gate area> | <inline in 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> }]

Une porte d'une expression en ligne est normalement attachée à un symbole de message à l'intérieur du cadre de l'expression en ligne, et à un symbole de message en dehors du cadre de l'expression en ligne. Lorsque aucun symbole n'est attaché à la porte, cela signifie qu'un message incomplet est associé à cette porte. Lorsque aucun message n'est attaché en dehors de la porte, ceci signifie que la porte se propage au cadre suivant.

<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>]

Le premier symbole <general order symbol> est une relation d'ordre général dans l'expression en ligne tel que l'événement dans l'expression est spécifié avant la porte. Le symbole optionnel <general order symbol> fait référence à une relation d'ordre général attachée à la porte en dehors de l'expression en ligne.

<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>]

Le premier symbole <general order symbol> est une relation d'ordre général dans l'expression en ligne tel que la porte est spécifiée avant l'événement dans l'expression. Le symbole optionnel <general order symbol> fait référence à une relation d'ordre général attachée à la porte en dehors de l'expression en ligne.

<def gate area> ::= { <def out gate area> | <def in gate area> | <def order out gate area> | <def order 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> }

NOTE 1 – La pointe de la flèche des symboles <message symbol> et <found message symbol> doit pointer sur la zone <def out gate area>.

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

NOTE 2 – Le début de la flèche des symboles <message symbol> et <lost message symbol> doit être attaché à la zone <def in 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>

<gate identification> ::=
<gate name>

<actual gate area> ::=
<actual out gate area> | <actual 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> }]

NOTE 3 – Le début de la flèche des symboles <message symbol> et <lost message symbol> part de la zone <actual out gate area>.

<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> }]

NOTE 4 – La pointe de la flèche des symboles <message symbol> et <found message> pointe sur la zone <actual in gate area>.

<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>

Sémantique statique

Bien que dans un MSC des noms de portes <gate name> ambigus soient admis, il n'est pas concevable que les références utilisent l'une de ces portes.

4.5 L'ordonnancement général

L'ordonnancement général est utilisé pour décrire deux événements ordonnés dans le temps, lorsque cet ordre ne peut pas être déduit de l'ordre imposé par les messages.

Grammaire textuelle concrète

La grammaire textuelle est décrite au 4.1.

Grammaire graphique concrète

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

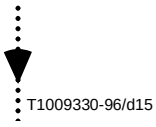
<general order symbol> ::=
<general order symbol1> | <general order symbol2>

<general order symbol1> ::=

NOTE 1 – Le symbole <general order symbol1> peut avoir une forme d'escalier, c'est-à-dire une succession de segments horizontaux et verticaux consécutifs. Les segments d'un symbole <general order symbol1> peuvent se superposer, seulement si aucune ambiguïté avec les autres symboles <general order symbol1> impliqués n'apparaît. Cela signifie qu'aucun nouvel ordre ne peut être déduit.

Le symbole <general order symbol1> peut aussi apparaître dans un symbole <instance axis symbol2> (forme de colonne). Les lignes de connection entre le symbole <general order symbol1> et la zone des événements <ordered event area> ordonnés doivent être horizontales.

<general order symbol2> ::=



NOTE 2 – Le symbole <general order symbol2> peut avoir n'importe quelle orientation et même être brisé.

<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>
| <timer area>
| <create area>
| <action symbol>

Sémantique statique

L'ordre partiel des événements défini par les constructions d'ordonnancement général et les messages n'est pas réflexif.

Sémantique

Les symboles d'ordre général décrivent l'ordre des événements qui sinon ne seraient pas ordonnés. Ceci est surtout utilisé lorsqu'on désire ordonner les événements dans une corégion pour diminuer le nombre de combinaisons possibles.

Exemples d'ordonnancement général

Les quatre exemples de la Figure 4-3 illustrent le même ordonnancement général. Les ordres suivants sont définis: a avant b, a avant d, c avant b, c avant d. a et c ne sont pas ordonnés et b et d ne sont pas ordonnés.

4.6 Condition

Une condition décrit soit un état global du système (condition globale) correspondant à toutes les instances contenues dans le MSC, soit un état correspondant à un sous-ensemble d'instances (condition non globale). Dans le second cas, la condition peut être locale, c'est-à-dire attachée à une seule instance. Dans la représentation textuelle, la condition doit être définie pour chaque instance à laquelle elle est attachée au moyen du mot clé **condition** accompagné du nom de condition. Si la condition est attachée à plusieurs instances, le mot clé **shared** ainsi que la liste des instances identifient l'ensemble des instances partageant la condition. Une condition globale se rapportant à toutes les instances est définie par le mot clé **shared all**.

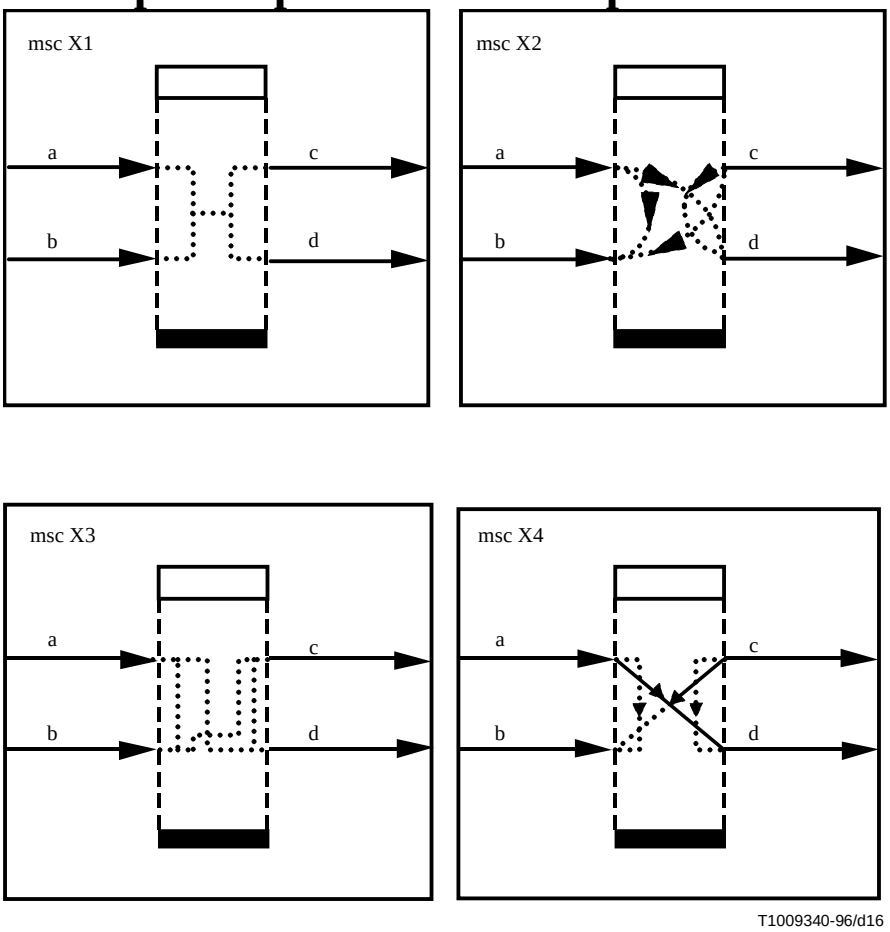


Figure 4-3/Z.120 – Exemple pour l'ordonnancement général

Les conditions globales peuvent être utilisées pour restreindre la composition des MSC dans le diagramme des MSC de haut niveau. Ce point sera développé au 5.5 "MSC de haut niveau (HMSC)".

Grammaire textuelle concrète

<shared condition> ::=
<condition identification> <shared>

<condition identification> ::=
condition <condition name list>

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

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

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

<condition> ::= <condition identification>

Pour chaque nom <instance name> contenu dans une liste <shared instance list> d'une <condition>, une instance avec une <condition> partagée correspondante doit être spécifiée. Si l'instance *b* fait partie de la liste <shared instance list> d'une <condition> partagée attachée à une instance *a*, alors l'instance *a* doit faire partie de la liste <shared instance list>

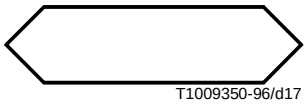
de la <condition> partagée correspondante attachée à l'instance *b*. Si les instances *a* et *b* partagent la même <condition>, alors pour chaque message échangé entre ces instances, le <message input> et le <message output> doivent être placés tous les deux, soit avant, soit après la <condition>. Si deux conditions sont ordonnées directement (parce qu'elles ont une instance en commun) ou ordonnées indirectement par des conditions sur d'autres instances, cet ordre doit être respecté sur toutes les instances partageant ces deux conditions.

Grammaire graphique concrète

<condition area> ::=

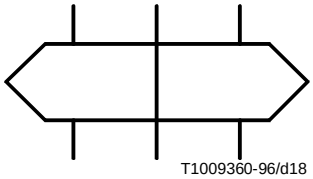
<condition symbol> ***contains*** <condition name list>
is attached to { <instance axis symbol>* } ***set***

<condition symbol> ::=



T1009350-96/d17

La zone <condition area> peut faire référence à une seule instance, ou être attachée à plusieurs instances. Si une <condition> partagée croise un symbole <instance axis symbol> d'une instance non concernée par la condition, le symbole <instance axis symbol> la traverse.



T1009360-96/d18

Sémantique

Les conditions globales qui représentent des états globaux du système, concernent toutes les instances du MSC. Pour chaque diagramme de séquence des messages:

- une condition globale initiale (état initial global);
- une condition globale finale (état final global); et
- des conditions globales intermédiaires (états intermédiaires globaux),

doivent être spécifiées en utilisant le mot clé **shared all** dans la représentation textuelle.

Les conditions globales initiales, intermédiaires et finales ne sont pas simplement introduites à des fins de documentation au sens de commentaires ou d'illustrations. Les conditions globales indiquent de possibles continuations entre diagrammes de séquence des messages contenant le même ensemble d'instances, par identification de la condition. Les conditions peuvent ainsi être utilisées pour restreindre des compositions possibles de MSC. La composition réelle des MSC est définie sous forme d'un diagramme HMSC (voir 5.5). Les compositions spécifiées doivent s'accorder avec les conditions contenues dans les MSC conformément à la définition des continuations permises.

4.7 Temporisateur

Les MSC permettent de spécifier, soit l'armement d'un temporisateur puis son expiration, soit l'armement d'un temporisateur puis son désarmement (supervision du temps). De plus, les constructions indépendantes des temporisateurs – l'armement, le désarmement et l'expiration d'un temporisateur – peuvent être utilisées séparément, par exemple dans le cas où une expiration ou une supervision d'un temporisateur est partagée entre différents MSC. Dans la représentation graphique, le symbole d'armement a la forme d'un sablier relié à l'axe de l'instance par une ligne (brisée). L'expiration est décrite par une flèche de message pointant sur l'instance attachée au symbole du sablier. Le symbole de désarmement a la forme d'une croix connectée à l'instance par une ligne (brisée).

La spécification d'un nom d'instance de temporisateur et la spécification de la durée du temporisateur sont optionnelles, aussi bien dans la représentation textuelle que graphique.

Grammaire textuelle concrète

<timer statement> ::=
<set> | <reset> | <timeout>

<set> ::= **set** <timer name> [**,** <timer instance name>] [(<duration name>)]

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

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

Au cas où le nom <timer name> n'est pas suffisant pour une correspondance bi univoque, le nom d'instance <timer instance name> doit être utilisé.

Grammaire graphique concrète

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

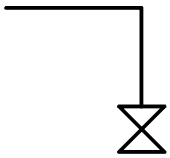
<timer set area> ::= <timer set area1> | <timer set area2>

<timer set area1> ::=
<set symbol> ***is associated with*** <timer name> [(<duration name>)]
is attached to <instance axis symbol>
[***is attached to*** {<set-reset symbol> | <reset symbol2> | <timeout symbol3> }]

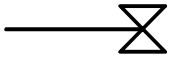
<timer set area2> ::=
<set-reset symbol> ***is associated with*** <timer name> [(<duration name>)]
is attached to <instance axis symbol>
is attached to <set symbol>
[***is attached to*** { <reset symbol2> | <timeout symbol3> }]

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

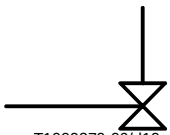
<set symbol1> ::=



<set symbol2> ::=



<set-reset symbol> ::=



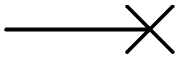
<timer reset area> ::=
<timer reset area1> | <timer reset area2>

<timer reset area1> ::=
<reset symbol1> ***is associated with*** <timer name> [(<duration name>)]
is attached to <instance axis symbol>

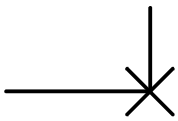
<timer reset area2> ::=
<reset symbol2> ***is associated with*** <timer name> [(<duration name>)]
is attached to <instance axis symbol>
is attached to { <set symbol> | <set-reset symbol> }

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

<reset symbol1> ::=



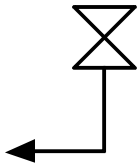
<reset symbol2> ::=



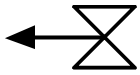
<timeout area> ::= <timeout area1> | <timeout area2>

<timeout area1> ::= <timeout symbol> ***is associated with*** <timer name> [(<duration name>)]
is attached to <instance axis symbol>

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



<timeout symbol2> ::=



<timeout area2> ::=
<timeout symbol3>
is attached to <instance axis symbol>
is attached to { <set symbol> | <set-reset symbol> }

<timeout symbol3> ::=

Sémantique

Les instructions d'armement et de désarmement sont extraites du langage SDL. L'armement désigne l'armement du temporisateur et le désarmement désigne le désarmement du temporisateur. L'expiration du temporisateur correspond à la consommation d'un signal de temporisation en SDL.

L'armement d'un temporisateur a toujours lieu avant le désarmement ou l'expiration correspondante.

Si des messages coïncident avec d'autres événements, se reporter aux règles de dessin du 2.4.

4.8 Action

En plus de l'échange de messages, des actions peuvent être spécifiées en MSC. Un texte informel est attaché à ces actions.

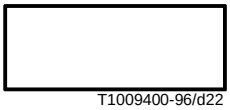
Grammaire textuelle concrète

<action> ::= **action** <action character string>

Grammaire graphique concrète

<action area> ::= <action symbol>
is attached to <instance axis symbol>
{ ***is followed by*** <general order area> }*
{ ***is attached to*** <general order area> }*
contains <action text>

<action symbol> ::=



T1009400-96/d22

Lorsque l'axe d'instance est au format colonne, la largeur du symbole <action symbol> doit coïncider avec la largeur de la colonne.

Sémantique

Une action décrit une activité interne à une instance.

4.9 Création d'instance

De même qu'en SDL, la création et la terminaison d'une instance peuvent être spécifiées en MSC. Une instance peut être créée par une autre instance. Aucun message ne doit se rapporter à une instance avant sa création.

Grammaire textuelle concrète

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

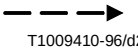
Pour chaque création <create> il doit exister une instance correspondante portant le nom spécifié. Le nom <instance name> doit se rapporter à l'instance de type processus, si ce type est spécifié. Une instance ne peut être créée qu'une seule fois, c'est-à-dire qu'au sein d'un *simple* MSC, deux <create>s ou plus ne doivent pas apparaître.

Grammaire graphique concrète

<create area> ::= <createline symbol> [***is associated with*** <parameter list>]
is attached to { <instance axis symbol> <instance head area> } ***set***
{ ***is followed by*** <general order area> }*
{ ***is attached to*** <general order area> }*

La création est représentée par l'extrémité du symbole <createline symbol> ne possédant pas la flèche. La création est attachée à un axe d'instance. Si la zone <create area> est généralement ordonnée, cet ordre s'applique à l'événement de création. La pointe de la flèche pointe sur le symbole de l'en-tête de l'instance créée.

<createline symbol> ::=



T1009410-96/d23

L'image miroir du symbole <createline symbol> est autorisée.

Sémantique

La création définit une création dynamique d'une instance par une autre.

Si la création d'instance coïncide avec d'autres événements, se reporter aux règles de dessin du 2.4.

4.10 Instance stop

La terminaison d'une instance est le pendant, dans un certain sens, de la création d'une instance. Cependant, une instance décide elle-même de sa terminaison alors qu'une instance est créée par une autre instance.

Grammaire textuelle concrète

<stop> ::= **stop**

Le symbole <stop> à la fin d'une instance n'est permis que pour les instances de type processus si son type est spécifié.

Grammaire graphique concrète

<stop symbol> ::=

T1009420-96/d24

Sémantique

L'arrêt à la fin d'une instance provoque la terminaison de cette instance.

5 Concepts structurels

Dans ce paragraphe, des concepts de haut niveau sont introduits. Ils portent sur une généralisation de l'ordre temporel (corégion) et la composition et la décomposition d'instances, les concepts de référence à un MSC et de composition des MSC basés sur le procédé algébrique.

5.1 Corégion

L'ordre total des événements le long de chaque instance (voir 4.1) n'est en général pas approprié pour les entités de plus haut niveau que les processus SDL.

De ce fait, la notion de corégion est introduite pour spécifier des événements non ordonnés sur une instance. Une telle notion de corégion couvre en particulier le cas pratique important de deux messages entrants (ou plus) pour lesquels l'ordre de consommation n'a pas d'importance. Un ordre généralisé peut être défini par des relations d'ordre général.

Grammaire textuelle concrète

<coregion> ::= **concurrent** <end> <coevent>* **endconcurrent** <coevent> ::=
<orderable event> <end>

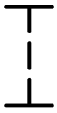
Grammaire graphique concrète

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

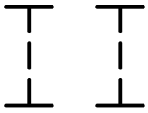
<coregion symbol> ::=
<coregion symbol1> | <coregion symbol2><coevent layer> ::=
<coevent area> | <coevent area> ***above*** <coevent layer>

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

<coregion symbol1> ::=

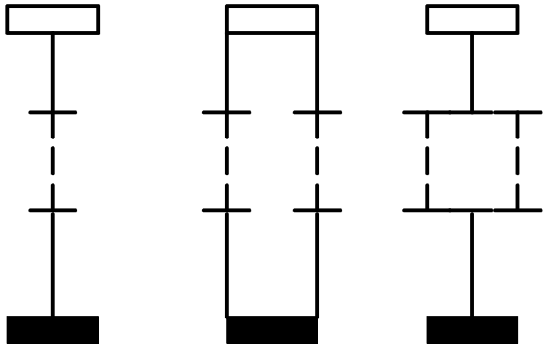


<coregion symbol2> ::=



T1009430-96/d25

Règle de dessin: Le fait que le symbole <coregion symbol> soit attaché au symbole <instance axis symbol> signifie que le symbole <coregion symbol> doit recouvrir le symbole <instance axis symbol> comme dans la Figure 5-1:



T1009440-96/d26

Figure 5-1/Z.120 – Différentes formes de corégion

Le symbole <coregion symbol1> ne doit pas être attaché au symbole <instance axis symbol2>.

Sémantique

Pour les MSC, l'ordre temporel total des événements est assuré au sein de chaque instance. En utilisant une corégion, une exception peut être faite à cette règle: les événements contenus dans une corégion ne sont pas ordonnés dans le temps si aucune construction supplémentaire de synchronisation sous forme de relations d'ordre général n'est spécifiée.

Si un armement de temporisateur suivi de son expiration ou de son réarmement apparaît dans une corégion, l'ordre entre ces événements est préservé.

5.2 Décomposition d'instance

Les instances dans les MSC peuvent faire référence aux entités à différents niveaux d'abstraction: ceci est déjà signalé à l'aide des mots clés (**system**, **block**, **service**, **process**). Les opérations de décomposition sur les instances peuvent être définies en déterminant la transition entre les différents niveaux d'abstraction. Si les instances sont composées en une instance, alors l'ordre total des événements le long de cette instance peut être répercuté pour préserver le comportement externe.

Par le mot clé **decomposed** et, éventuellement, la référence à la sous-structure <substructure reference>, un diagramme de séquence des messages plus fin peut être attaché à une instance.

Grammaire textuelle concrète

<decomposition>::=
decomposed [<substructure reference>]

<substructure reference>::=
as <message sequence chart name>

A chaque instance contenant le mot clé **decomposed**, un MSC "affiné" correspondant portant le nom spécifié doit être défini. A chaque message <message output> sur une instance décomposée, un message <message input> correspondant envoyé à l'extérieur du MSC "affiné" doit être spécifié. Cette remarque est également valable pour les messages entrants.

Les expressions en ligne et les références MSC ne doivent pas être liées à des instances décomposées.

Grammaire graphique concrète

La grammaire graphique est donnée au 4.2.

Sémantique

Par le mot clé **decomposed** un diagramme de séquence des messages plus fin (faisant référence à une sous-structure MSC) peut être attaché à une instance. Le nom de ce MSC est défini soit explicitement par la référence <substructure reference> optionnelle, soit par le même nom que l'instance décomposée. Le MSC affiné représente une décomposition de cette instance sans affecter son comportement. Dans la représentation textuelle, les messages entrants ou sortants de ces MSC sont caractérisés par l'adresse **env,** et par une connexion avec le bord du cadre de la sous-structure MSC pour la représentation graphique. Leur connexion avec les instances externes est fournie par les messages envoyés et consommés par l'instance décomposée, en utilisant l'identification du nom du message. On doit pouvoir superposer le comportement externe d'un MSC affiné sur les messages de l'instance décomposée. L'ordre des messages spécifié sur une instance décomposée doit être conservé dans le MSC affiné. Les actions et les conditions dans un MSC affiné peuvent être considérées comme un raffinement des actions et des conditions de l'instance décomposée. Cependant, contrairement aux messages il n'y a aucune superposition formelle sur les instances décomposées; il n'est pas nécessaire de suivre des règles formelles pour affiner les actions et les conditions.

5.3 Expression en ligne

Par les opérateurs des expressions en ligne, la composition de structures d'événements peut être définie dans un MSC. Ces opérateurs sont l'alternative, la composition parallèle, l'itération, l'exception et les parties optionnelles.

Grammaire textuelle concrète

<shared inline expr> ::=
{ <shared loop expr> | <shared opt expr> | <shared alt expr> |
<shared par expr> | <shared exc expr> }

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

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

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

<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

<inline expr> ::=
<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

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

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

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

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

<loop boundary> ::=
'<' <inf natural> [, <inf natural>] '>'

<inf natural> ::=
inf | <natural name>$^{+}$

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

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

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

Grammaire graphique concrète

<inline expression area> ::=
<loop area> | <opt area> | <par area> | <alt area> | <exc area>

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

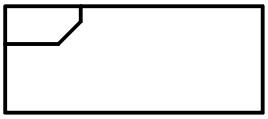
<opt area> ::=
<inline expression symbol> ***contains***
{ **opt** <operand area> }
is attached to { <instance axis symbol>* } ***set***
is attached to { <inline gate area>* | <inline order gate area>* } ***set***

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

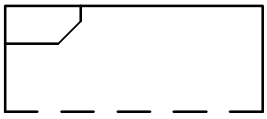
<par area> ::=
<inline expression symbol> ***contains***
{ **par** <operand area>
{ ***is followed by*** <separator area> ***is followed by*** <operand area> }* }
is attached to { <instance axis symbol>* } ***set***
is attached to { <inline gate area>* | <inline order gate area>* } ***set***

<alt area> ::=
<inline expression symbol> ***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> }*

<separator area> ::=
<separator symbol>

<separator symbol> ::=

T1009450-96/d27

La zone <inline expression area> peut faire référence à une seule instance, ou être attachée à plusieurs instances. Si une expression en ligne partagée traverse une instance qui n'est pas impliquée dans cette expression en ligne, son axe ne doit pas traverser cette zone.

Toutes les expressions d'exception doivent être partagées par toutes les instances du MSC.

Les expressions en ligne ne doivent pas être attachées à des instances décomposées.

Sémantique

Dans la représentation graphique, les mots clés représentant les opérateurs **alt**, **par**, **loop, opt** et **exc** sont placés dans le coin supérieur gauche et représentent respectivement l'alternative, le parallélisme, l'itération, la région optionnelle et l'exception. Dans la représentation graphique les opérandes sont contenus dans un cadre et sont séparés par des lignes pointillées.

L'opérateur **alt** définit une alternative dans l'exécution de différentes parties de MSC. Ceci signifie que si plusieurs sections de MSC sont censées être des alternatives, une seule d'entre elles sera exécutée. Dans le cas où les sections de MSC composant l'alternative ont un préambule commun, le choix de la section qui devra être exécutée le sera après l'exécution de ce préambule.

L'opérateur **par** définit une exécution parallèle de sections MSC. Ceci signifie que tous les événements composant ces sections MSC seront exécutés, la seule contrainte étant que l'ordre des événements à l'intérieur de chacune d'entre elles sera préservé.

La construction de boucle peut avoir plusieurs formes. La représentation de base est "**loop** <n,m>" où n et m sont des entiers naturels. Ceci signifie que l'opérande peut être répété au moins n fois et au plus m fois. Les entiers naturels peuvent être remplacés par le mot clé **inf**, tel que "**loop** <n,**inf**>". Ceci signifie que la boucle sera répétée au moins n fois. Si le second opérande est omis comme dans "**loop** <n>", ceci est équivalent à "**loop** <n,n>". Ainsi "**loop** <**inf**>" représente une boucle infinie. Si les limites de la boucle sont omises comme dans "**loop**", ceci sera interprété comme "**loop** <1,**inf**>". Si le premier opérande est plus grand que le second, la boucle sera exécutée 0 fois.

L'opérateur **opt** est une alternative où le second opérande est le MSC vide.

L'opérateur **exc** est une convention permettant de décrire les cas exceptionnels dans un MSC: soit les événements à l'intérieur du symbole <exc inline expression symbol> sont exécutés et le MSC est terminé, soit les événements suivant le symbole <exc inline expression symbol> sont exécutés. Ainsi, l'opérateur **exc** peut être vu comme une alternative où le second opérande est la fin du MSC.

Dans la représentation textuelle, le symbole optionnel <inline gate interface> définit les messages entrant ou quittant l'expression en ligne par les portes. A l'aide de l'identification du nom du message et de l'identification optionnelle du nom de la porte, la porte <inline gate interface> définit également la connexion directe du message entre deux expressions en ligne.

5.4 Référence MSC

Les références MSC sont utilisées pour spécifier un renvoi vers d'autres diagrammes MSC d'un document MSC. Les références MSC sont des objets du même type que le MSC référencé.

Les références MSC ne font pas seulement référence à un simple MSC, mais aussi à des "expressions de références MSC". Des "expressions de références MSC" sont des expressions MSC textuelles construites à partir des opérateurs **alt**, **par**, **seq, loop**, **opt**, **exc** et **subst,** et des références MSC.

Les opérateurs **alt**, **par**, **loop**, **opt** et **exc** sont décrits au 5.3. **Seq** est l'opération de séquencement faible: seuls les événements de la même instance sont ordonnés.

Subst est l'opération de substitution des concepts dans le MSC référencé. Les noms de messages sont substitués par les noms de messages, les noms d'instances par les noms d'instances et les noms de MSC par des noms de MSC.

Les portes réelles d'une référence MSC peuvent se connecter à des constructions correspondantes dans le MSC contenant cette référence. C'est-à-dire qu'une porte de message réelle se connecte à une autre porte de message réelle ou à une instance ou à une définition de porte de message du MSC contenant la référence. De plus une porte d'ordre réelle peut se connecter à une autre porte d'ordre réelle, ou à un événement ordonné ou à une définition de porte d'ordre.

Grammaire textuelle concrète

<shared msc reference> ::=
reference [<msc reference identification>**:**] <msc ref expr>
<shared> [<reference gate interface>]

<msc reference> ::=
reference [<msc reference identification>**:**] <msc ref expr>
[<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 exc expr> { **seq** <msc ref exc expr> }*

<msc ref exc expr> ::=
[**exc**] <msc ref opt expr>

<msc ref opt expr> ::=
[**opt**] <msc ref loop expr>

<msc ref loop expr> ::=
[**loop**] [<loop boundary>]
{ **empty** | <msc name> [<parameter substitution>] | (<msc ref expr>) }

<parameter substitution> ::=
subst <substitution list>

<substitution list> ::=
<substitution> [, <substitution list>]

<substitution> ::=
<replace message> | <replace instance> | <replace msc>

<replace message> ::=
[**msg**] <message name> **by** <message name>

<replace instance> ::=
[**inst**] <instance name> **by** <instance name>

<replace msc> ::=
[**msc**] {**empty** | <msc name> } **by** {**empty** | <msc name> }

<reference gate interface> ::=
{ <end> **gate** <ref gate> }*

<ref gate> ::=
<actual out gate> | <actual in gate> | <actual order out gate> | <actual order in gate>

Pour éviter les éventuelles ambiguïtés concernant les noms lors des substitutions, les mots clés optionnels **msg**, **inst** et **msc** devront être utilisés.

La substitution d'un nom de MSC doit partager la même interface de porte que l'objet remplacé.

A chaque instance définie dans le diagramme encadré, une référence MSC doit attacher l'objet référencé. Si deux diagrammes référencés par deux références MSC dans un diagramme encadré partagent les mêmes instances, ces instances doivent aussi apparaître dans le diagramme encadré.

Une référence MSC peut être attachée à des instances n'étant pas contenues dans le diagramme référencé.

L'interface d'une référence MSC doit être identique à l'interface des MSC référencés dans l'expression, c'est-à-dire que n'importe quelle porte attachée à la référence doit avoir une définition correspondante dans les MSC référencés. La correspondance est donnée par la direction et le nom du message associé à la porte ainsi que par le nom de la porte si celui-ci est présent. Ce nom de porte est unique dans l'expression référencée.

Si l'expression <msc ref expr> est une expression d'opérateurs textuels plutôt qu'un simple nom <msc name> et lorsque plusieurs références MSC font référence au même MSC, le nom optionnel <msc reference name> dans <msc reference identification> doit être utilisé pour adresser une référence MSC dans une définition de message (voir 4.3).

Grammaire graphique concrète

<msc reference area>::=

<msc reference symbol>
contains { <msc ref expr> [<actual gate area>*] } ***set***
is attached to { <instance axis symbol>* } ***set***
is attached to { <actual gate area>* } ***set***

<msc reference symbol>::=

T1009460-96/d28

La zone <msc reference area> peut être attachée à une ou plusieurs instances. Si une référence partagée <msc reference area> traverse un symbole <instance axis symbol> d'une instance non impliquée dans cette référence, le symbole <instance axis symbol> la traverse.

Les références MSC ne doivent pas être attachées à des instances décomposées.

Les références MSC ne doivent pas directement ou indirectement faire référence à elles-mêmes (récursivité).

Un MSC contenant des références et des substitutions est illégal si, après avoir développé les références et appliqué les substitutions à maintes reprises, on obtient un MSC illégal.

Sémantique

Chaque MSC peut être vu comme une définition d'un type MSC. Les types MSC peuvent être utilisés dans d'autres types MSC par des références MSC.

Un type MSC peut être attaché à son environnement par les portes de message. Si un type MSC est utilisé dans un autre type, les portes sont utilisées pour définir les points de connexion. Les identificateurs de portes peuvent être associés aux points de connexion par des noms.

En général, la référence MSC peut faire référence à une expression MSC textuelle pouvant être modifiée suite à une substitution. Dans le cas simple, où l'expression MSC est un nom MSC, la référence MSC désigne une définition de type MSC correspondante. La correspondance est donnée par le nom MSC unique dans le document MSC. Le résultat d'une référence avec substitution est que la définition du MSC référencé sera modifiée par les remplacements effectués à la suite des substitutions. **Subst** x **by** y signifie que x est remplacé par y. Toutes les substitutions dans une liste <substitution list> doivent être appliquées en parallèle. Ainsi, il n'est pas nécessaire de spécifier l'ordre des substitutions.

Si une définition MSC, dans laquelle on a appliqué une substitution, contient des références MSC, alors la substitution devrait être appliquée également aux définitions MSC correspondant à ces références MSC.

Dans la représentation textuelle, l'interface optionnelle <reference gate interface> définit les messages entrant ou quittant la référence MSC par les portes. Par l'identification du nom du message et éventuellement l'identification du nom de la porte, l'interface <reference gate interface> définit également la connexion directe d'un message entre deux références MSC.

Une référence MSC comportant le mot clé **empty** fait référence à un MSC ne contenant aucun événement et aucune instance.

5.5 MSC de haut niveau (HMSC)

Les MSC de haut niveau permettent de définir graphiquement les combinaisons possibles d'un ensemble de MSC. Un HMSC est essentiellement un graphe où chaque nœud est soit:

- un symbole de départ (il n'y a qu'un seul symbole de départ dans chaque HMSC);
- un symbole de fin;
- une référence MSC;
- une condition;
- un point de connexion; ou
- un cadre parallèle.

Les lignes de flux relient les nœuds dans le HMSC et indiquent la séquence possible entre les nœuds dans le HMSC. Les lignes de flux entrantes sont toujours reliées au bord supérieur du nœud alors que les lignes de flux sortantes sont reliées au bord inférieur. Si plus d'une ligne de flux sort d'un nœud, il s'agit d'une alternative.

Les références MSC peuvent être utilisées soit pour faire référence à un MSC simple soit à un certain nombre de MSC utilisant une expression MSC textuelle.

Les conditions dans les HMSC peuvent être utilisées pour définir des états globaux du système et imposent des restrictions sur les MSC référencés dans le HMSC.

Les cadres parallèles contiennent un ou plusieurs petits HMSC constituant les opérandes d'un opérateur parallèle, c'est-à-dire que les événements dans les différents petits HMSC peuvent être imbriqués.

Les points de connexion sont introduits pour simplifier la mise en page des HMSC et n'ont pas de signification.

Grammaire textuelle concrète

<msc expression> ::=
<start> <node expression> *

<start> ::=
<label name> {**alt** <label name> }* <end>

<node expression> ::=
<label name> **:** { <node> **seq** (<label name> { **alt** <label name> }*) | **end** }<end>

<node> ::=
(<msc ref expr>)
| **empty** | <msc name>
| <par expression>
| **condition** <condition name list>
| **connect**

<par expression> ::=
expr <msc expression> **endexpr** { **par expr** <msc expression> **endexpr** }*

Sémantique statique

Tous les noms référencés <label name> dans <start> ou <node expression> doivent être définis dans le HMSC, c'est-à-dire apparaître sous la forme "<label name>**:**" dans une expression <node expression>. Chaque nœud dans un graphe HMSC doit être accessible à partir du symbole <start>, c'est-à-dire que le graphe doit être connecté.

Grammaire graphique concrète

<mscexpr area> ::=
{ <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>$^{+}$} ***se***

<hmsc end symbol> ::=

<hmsc line symbol> ::=
<hmsc line symboll> | <hmsc line symbol2>

<hmsc line symboll> ::=

<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> ***contains*** <msc ref expr>

<connection point symbol> ::=

<hmsc condition area>::=
<condition symbol> ***contains*** <condition name list>

<par expr area>::=
<par frame symbol> ***contains*** { <mscexpr area>$^{+}$} ***set***

<par frame symbol>::=
<frame symbol>

Règles de dessin

Le symbole <hmsc line symbol> peut être une ligne brisée et avoir n'importe quelle direction.

Le fait que le symbole <hmsc start symbol> soit suivi par le symbole <alt op area> signifie que les symboles <hmsc line symbol> sont reliés au coin inférieur du symbole <hmsc start symbol>. C'est ce que montre la Figure 5-2a où deux lignes <hmsc line symbol> suivent le symbole de départ <hmsc start symbol>:

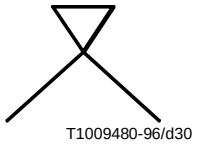


Figure 5-2a/Z.120 – Règles de dessin HMSC

Le fait que le symbole <hmsc line symbol> soit attaché à un autre symbole dans la règle de production <alt op area> signifie que les symboles <hmsc line symbol> sont reliés au bord supérieur du symbole en question. La Figure 5-2b illustre les cas où le symbole est un <msc reference symbol> et un <hmsc end symbol>:

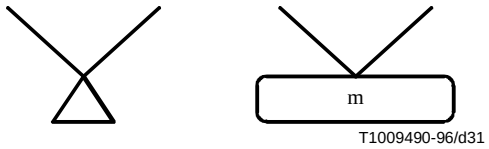


Figure 5-2b/Z.120 – Règles de dessin HMSC

Le fait que la zone <node area> soit suivie par une zone <alt op area> dans la règle de production <node expression area> signifie que les symboles <hmsc line symbol> sont reliés au bord inférieur du symbole dans la zone <node area>. La Figure 5-2c montre comment un symbole <msc reference symbol> est suivi par une zone <alt op area>:

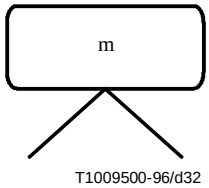


Figure 5-2c/Z.120 – Règles de dessin HMSC

Sémantique statique

La composition des MSC spécifiée à l'aide des HMSC peut être supervisée par les conditions dans les HMSC. Ces conditions peuvent être utilisées pour indiquer des états globaux du système. Notons que seules les conditions globales peuvent être utilisées pour restreindre la composition des MSC. Toutes les conditions d'un HMSC sont des conditions globales. Dans un simple MSC (un MSC décrit en utilisant les instances, les messages, etc.) seules les conditions partagées par toutes les instances du MSC sont globales et peuvent être utilisées pour restreindre la composition dans les HMSC. Les conditions non globales dans les MSC simples sont considérées comme des commentaires.

Chaque MSC simple, HMSC, cadre parallèle (dans un HMSC) et expression MSC (dans une référence MSC) a un ensemble de conditions initiales et un ensemble de conditions finales observant les règles suivantes:

L'ensemble des conditions initiales pour un MSC simple (c'est-à-dire un MSC non décrit par un diagramme HMSC) est défini comme suit:

- si le MSC possède une condition globale placée immédiatement après les en-têtes d'instances, alors l'ensemble des conditions initiales est composé des noms <condition name> définis par cette condition;
- si le MSC ne possède pas une telle condition globale, alors l'ensemble des conditions initiales est défini comme étant l'ensemble de tous les noms possibles <condition name>. Cette règle s'applique également à un MSC vide.

La définition de l'ensemble des conditions finales pour un MSC simple est la suivante:

- si le MSC possède une condition globale suivie uniquement par les fins d'instance, alors l'ensemble des conditions finales est composé des noms <condition name> définis par cette condition;
- si le MSC ne possède pas une telle condition globale, alors l'ensemble des conditions finales est défini comme étant l'ensemble de tous les noms possibles <condition name>. Cette règle s'applique également à un MSC vide.

La définition de l'ensemble des conditions initiales d'un HMSC est la suivante:

- si le symbole <hmsc start symbol> est immédiatement suivi par un ou plusieurs symboles <condition symbol>, alors l'ensemble des conditions initiales est défini comme étant l'intersection des ensembles des noms <condition name> définis par chacun des symboles <condition symbol>. Notons que si une branche sortante du symbole <start symbol> commence avec plus d'un symbole <condition symbol>, alors seul le premier est considéré pour faire partie des conditions initiales du HMSC;
- si le symbole <start symbol> n'est pas immédiatement suivi par un symbole <condition symbol>, alors l'ensemble des conditions initiales est défini comme étant l'ensemble complet de tous les noms <condition name> possibles.

La définition de l'ensemble des conditions finales d'un HMSC est la suivante:

- si le HMSC possède un ou plusieurs symboles <hmsc end symbol> immédiatement précédés par les symboles <condition symbol>, alors l'ensemble des conditions finales est l'intersection des ensembles des noms <condition name> définis par chaque symbole <condition symbol>;
- si aucun des symboles <hmsc end symbol> dans le HMSC n'est immédiatement précédé par un symbole <condition symbol>, alors l'ensemble des conditions finales est l'ensemble complet de tous les noms <condition name> possibles.

La définition de l'ensemble des conditions initiales d'une expression <msc ref expression> (c'est-à-dire l'expression associée à une référence <msc reference>) est la suivante (où m1 est le nom d'un MSC et e1 et e2 sont des expressions MSC):

- l'ensemble des conditions initiales pour une expression 'm1' est l'ensemble des conditions initiales du MSC m1;
- l'ensemble des conditions initiales pour 'e1 seq e2' est l'ensemble des conditions initiales pour e1;
- l'ensemble des conditions initiales pour 'e1 alt e2' est l'intersection de l'ensemble des conditions initiales pour e1 et de l'ensemble des conditions initiales pour e2;
- l'ensemble des conditions initiales pour 'e1 par e2' est l'intersection de l'ensemble des conditions initiales pour e1 et de l'ensemble des conditions initiales pour e2;
- l'ensemble des conditions initiales pour 'opt e1' est l'ensemble des conditions initiales pour e1;
- l'ensemble des conditions initiales pour 'loop e1' est l'ensemble des conditions initiales pour e1.

La définition de l'ensemble des conditions finales d'une expression MSC dans une référence MSC est la suivante (où m1 est le nom d'un MSC et e1 et e2 sont des expressions MSC):

- l'ensemble des conditions finales pour l'expression 'm1' est l'ensemble des conditions finales du MSC m1;
- l'ensemble des conditions finales pour 'e1 seq e2' est l'ensemble des conditions finales pour e2;
- l'ensemble des conditions finales pour 'e1 alt e2' est l'intersection de l'ensemble des conditions finales pour e1 et de l'ensemble des conditions finales pour e2;
- l'ensemble des conditions finales pour 'e1 par e2' est l'intersection de l'ensemble des conditions finales pour e1 et de l'ensemble des conditions finales pour e2;
- l'ensemble des conditions finales pour 'opt e1' est l'ensemble des conditions finales pour e1;
- l'ensemble des conditions finales pour 'loop e1' est l'ensemble des conditions finales pour e1.

Les conditions initiales et finales pour <par expr area> dans les HMSC sont définies de la même manière que pour les expressions parallèles (**par**):

- l'ensemble des conditions initiales pour <par expr area> est l'intersection de l'ensemble des conditions initiales des opérandes;
- l'ensemble des conditions finales pour <par expr area> est l'intersection de l'ensemble des conditions finales des opérandes.

Quatre restrictions statiques sont liées aux conditions dans les HMSC:

- si une référence <msc reference> est immédiatement précédée par un symbole <condition symbol>, et son ensemble associé de noms <condition name>, alors cet ensemble doit être un sous-ensemble de l'ensemble des conditions initiales des expressions <msc ref expression> associées à la référence <msc reference>;
- si une référence <msc reference> est immédiatement suivie par un symbole <condition symbol>, et son ensemble associé de noms <condition name>, alors cet ensemble doit être un sous-ensemble de

l'ensemble des conditions finales de l'expression <msc ref expression> associée à la référence <msc reference>;

- si une zone <par expr area> est immédiatement précédée par un symbole <condition symbol>, et son ensemble associé de noms <condition name>, alors cet ensemble doit être un sous-ensemble de l'ensemble des conditions initiales de <par expr area>;
- si une zone <par expr area> est immédiatement suivie par un symbole <condition symbol>, et son ensemble associé de noms <condition name>, alors cet ensemble doit être un sous-ensemble de l'ensemble des conditions finales de la zone <par expr area>.

Sémantique

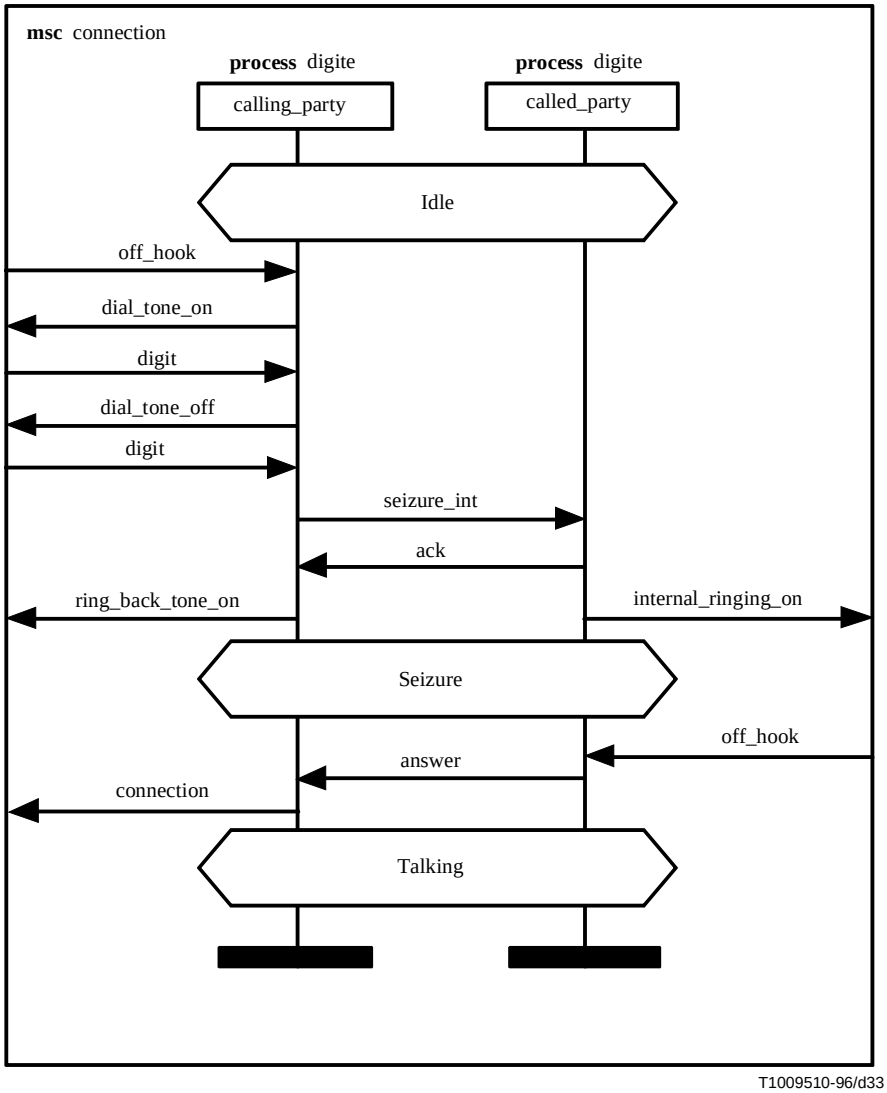
Quant à l'exécution, le graphe décrivant la composition de MSC dans un HMSC est interprété de la façon suivante. L'exécution commence au symbole <hmsc start symbol>. Puis elle continue avec un nœud suivant l'un des arcs sortant de ce symbole. Ces nœuds sont considérés comme étant des opérandes de l'opérateur **alt** (voir 5.3). Après exécution du nœud sélectionné, le processus de sélection et d'exécution est répété pour les arcs sortant de ce nœud. L'exécution d'un nœud terminal signifie la fin de l'exécution du HMSC. Une référence MSC s'exécute conformément à la description faite au 5.4. L'exécution d'une condition ou d'un point de connexion est une opération vide. L'exécution d'un cadre parallèle consiste en l'exécution des opérandes de ce cadre parallèle, comme le décrit le 5.3 pour l'opérateur **par**. Une exécution séquentielle de deux nœuds reliés par un arc est décrite par l'opérateur **seq** (voir 5.4).

6 Exemples de diagrammes de séquence des messages

Ce paragraphe est donné à titre indicatif plus que normatif.

6.1 Diagramme standard de flux de messages

Cet exemple illustre une connexion simplifiée réalisée dans un système de commutation. L'exemple contient les constructions MSC les plus basiques: instances (de processus), environnement, messages, conditions globales.

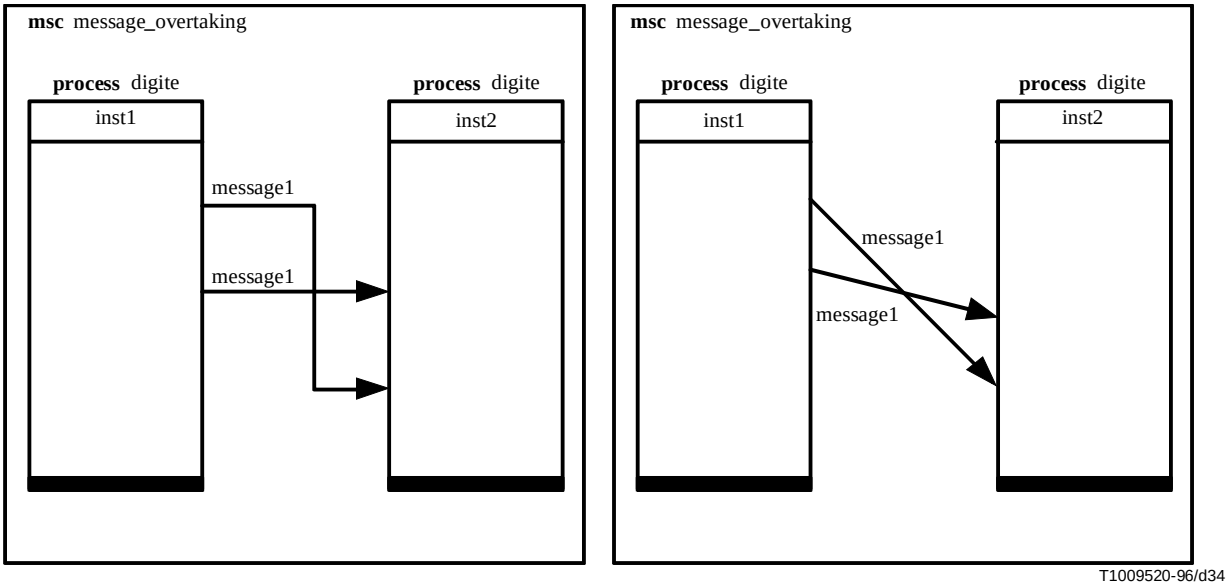


T1009510-96/d33

```
msc connection; inst calling_party: process digite, called_party: process digite;
        instance calling_party: process digite;
            condition 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;
        instance called_party: process digite;
            condition 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;
```

6.2 Dépassement de message

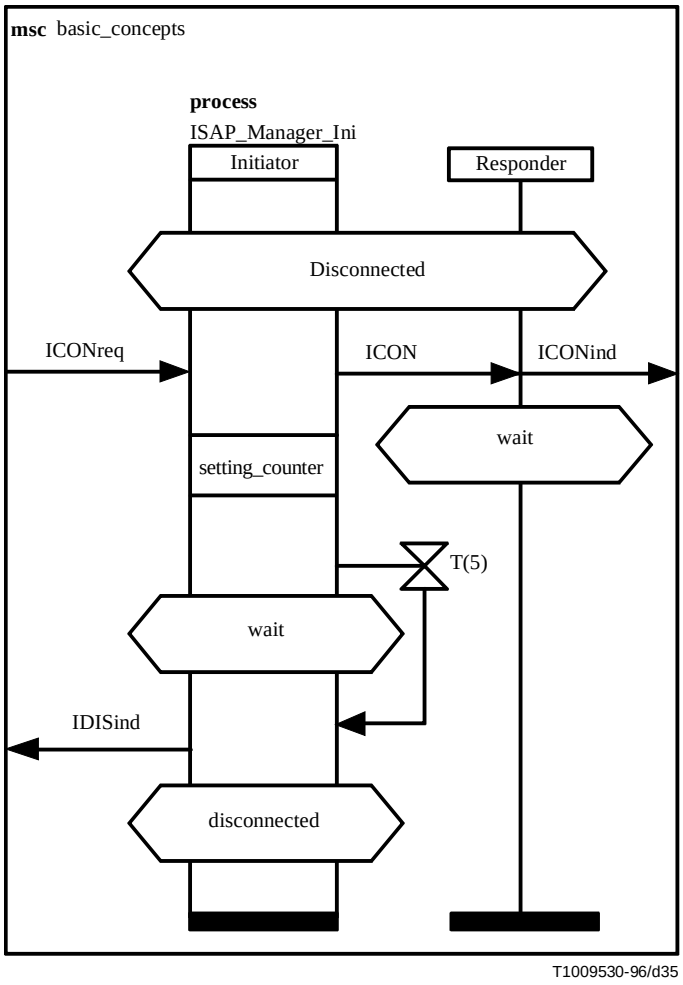
Cet exemple illustre le dépassement d'un message par un autre, les deux messages ayant le même nom: "message 1". Dans la représentation textuelle, les noms des instances de messages (a, b) sont utilisés afin d'avoir une correspondance biunivoque entre le message en entrée et le message en sortie. Dans la représentation graphique, les messages sont soit représentés par des flèches horizontales, l'une avec une portion de ligne brisée afin d'illustrer le dépassement, soit par des croisements de flèches obliques.




```
msc message_overtaking; inst inst1, inst2;
        instance inst1: process digite;
            out message1, a to inst2;
            out message1, b to inst2;
        endinstance;
        instance inst2: process digite;
            in message1, b from inst1;
            in message1, a from inst1;
        endinstance;
endmsc;
```

6.3 Concepts de base du MSC

Cet exemple contient les constructions MSC de base: instances, environnement, messages, conditions, actions et expiration de temporisateur. Dans la représentation graphique, les deux types de symboles d'instance sont utilisés: la forme ligne simple et la forme colonne.



Syntaxe textuelle orientée instance

```
msc basic_concepts; inst Initiator: process ISAP_Manager_Ini, Responder;
        instance Initiator: process ISAP_Manager_Ini;
            condition Disconnected shared all;
            in ICONreq from env;
            out ICON to Responder;
            action setting_counter;
            set T(5);
            condition wait shared;
            timeout T;
            out IDISind to env;
            condition disconnected shared;
        endinstance;
        instance Responder;
            condition Disconnected shared all;
            in ICON from Initiator;
            out ICONind to env;
            condition wait shared;
        endinstance;
endmsc;
```

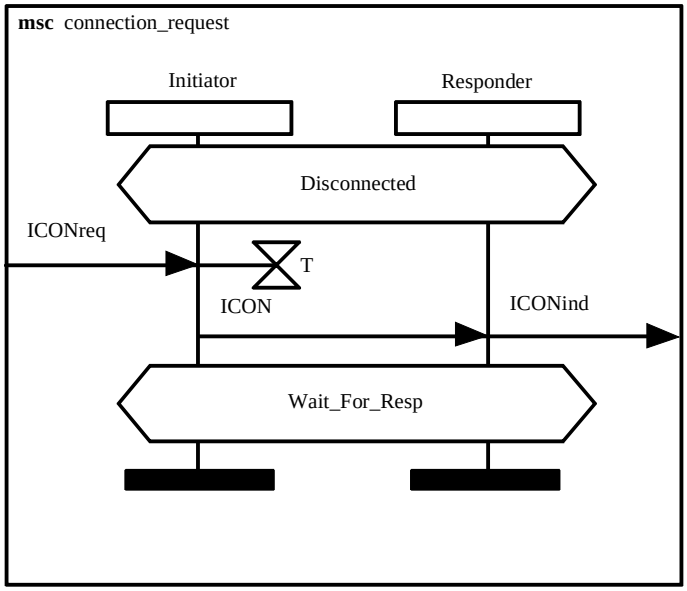
Syntaxe textuelle orientée événement

```
msc basic_concepts; inst Initiator: process ISAP_Manager_Ini, Responder;
Initiator:      instance process ISAP_Manager_Ini;
Responder:      instance;
all:            condition 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;
                set T(5);
                condition wait;
                timeout T;
                out IDISind to env;
                condition disconnected;
                endinstance;
endmsc;
```

6.4 Composition de MSC/décomposition de MSC

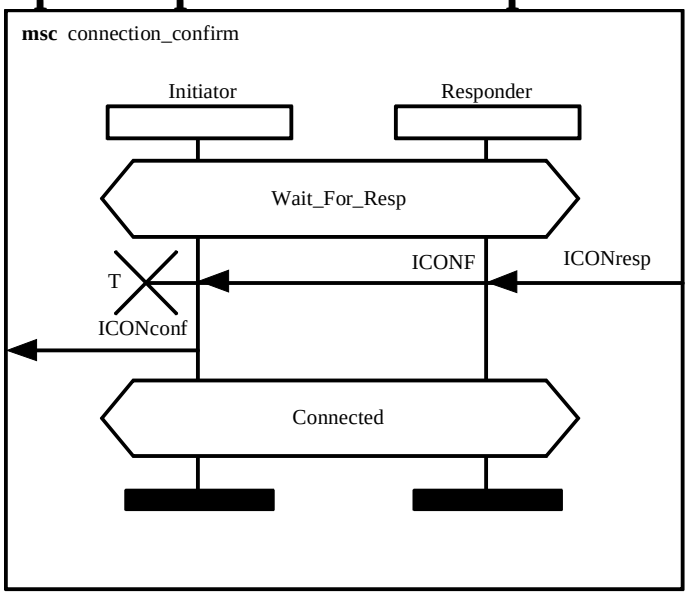
Cet exemple illustre la composition de MSC au moyen de conditions globales. La condition globale finale "Wait For Resp" du MSC "connection request" est identique à la condition globale initiale du MSC "connection confirm". Par conséquent, les deux MSC peuvent être composés pour donner le MSC résultant "composed" (voir exemple au 6.5).

La composition est définie au moyen du diagramme HMSC "con_setup".

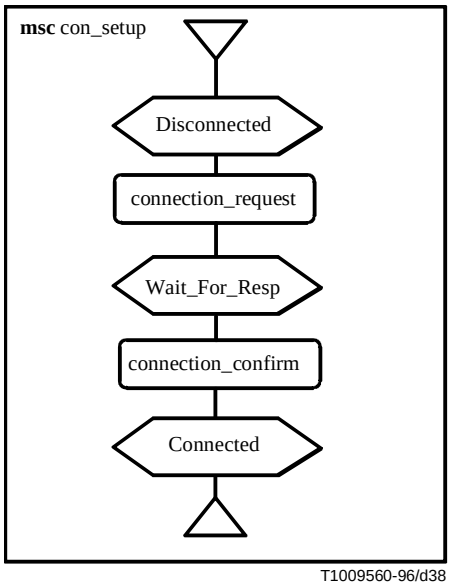


T1009540-96/d36

```
msc connection_request; inst Initiator, Responder;
        instance Initiator;
            condition Disconnected shared all;
            in ICONreq from env;
            set T;
            out ICON to Responder;
            condition Wait_For_Resp shared all;
        endinstance;
        instance Responder;
            condition Disconnected shared all;
            in ICON from Initiator;
            out ICONind to env;
            condition Wait_For_Resp shared all;
        endinstance;
endmsc;
```



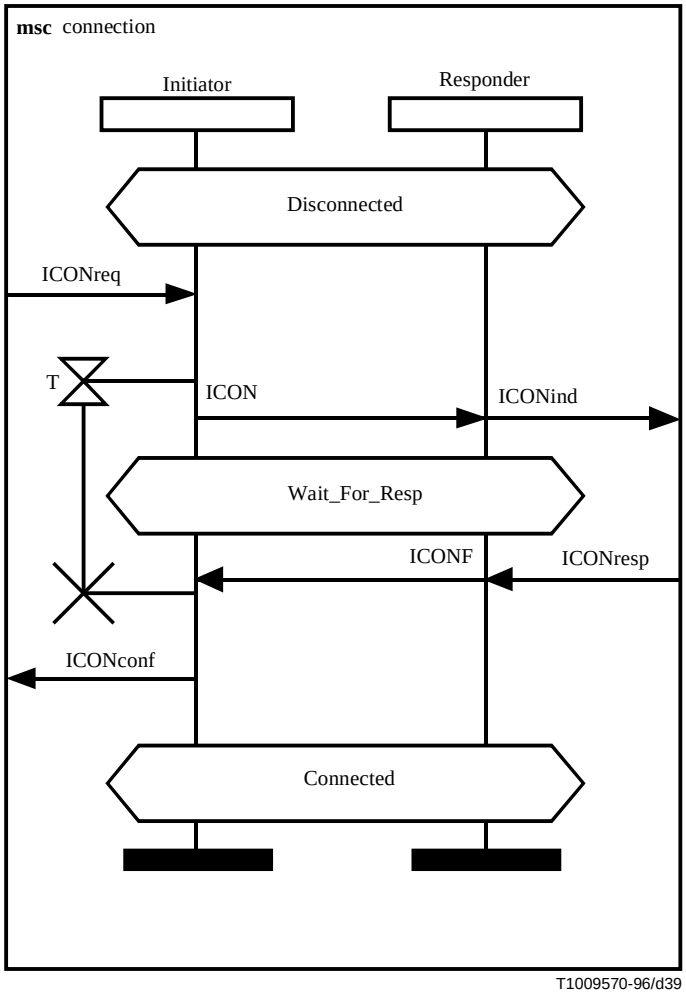
msc connection_confirm; **inst** Initiator, Responder;
instance Inititator;
condition Wait_For_Resp **shared all**;
in ICONF **from** Responder;
reset T;
out ICONconf **to env**;
condition Connected **shared all**;
endinstance;
instance Responder;
condition Wait_For_Resp **shared all**;
in ICONresp **from env**;
out ICONF **to** Initiator;
condition Connected **shared all**;
endinstance;
endmsc;



msc con_setup;
expr L1;
L1: **condition** Disconnected **seq** (L2);
L2: connection_request **seq** L3;
L3: **condition** Wait_For_Resp **seq** (L4);
L4: connection_confirm **seq** (L5);
L5: **condition** Connected **seq** (L6);
L6: **end**;
endmsc;

6.5 MSC avec supervision de temps

Cet exemple "MSC connection" contient un réarmement de temporisateur.

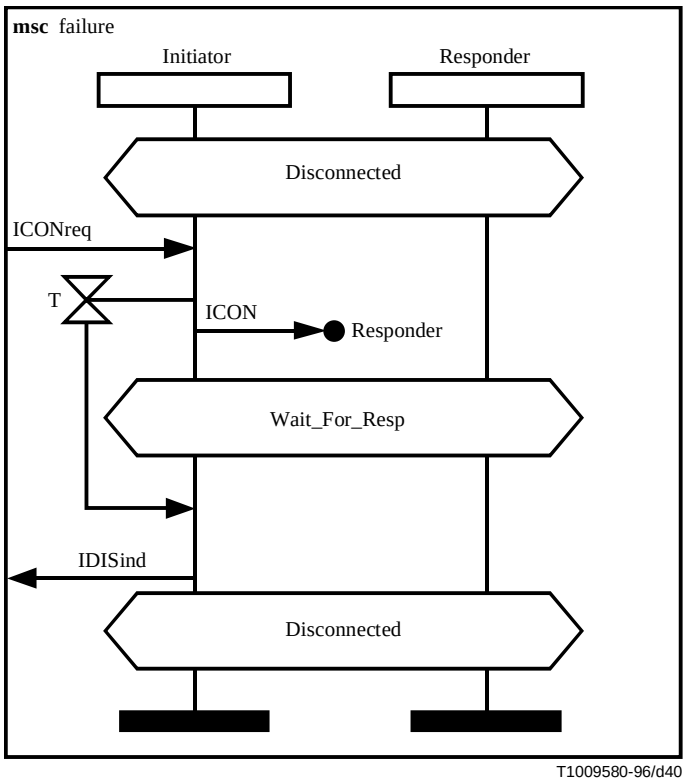


msc connection; **inst** Initiator, Responder;
instance Initiator;
condition Disconnected **shared all**;
in ICONreq **from env**;
set T;
out ICON **to** Responder;
condition Wait_For_Resp **shared all**;
in ICONF **from** Responder;
reset T;
out ICONconf **to env**;
condition Connected **shared all**;

endinstance;
instance Responder;
condition 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;

6.6 MSC avec un message perdu

Dans cet exemple, le "MSC failure" contient une expiration de temporisateur suite à une perte de message.

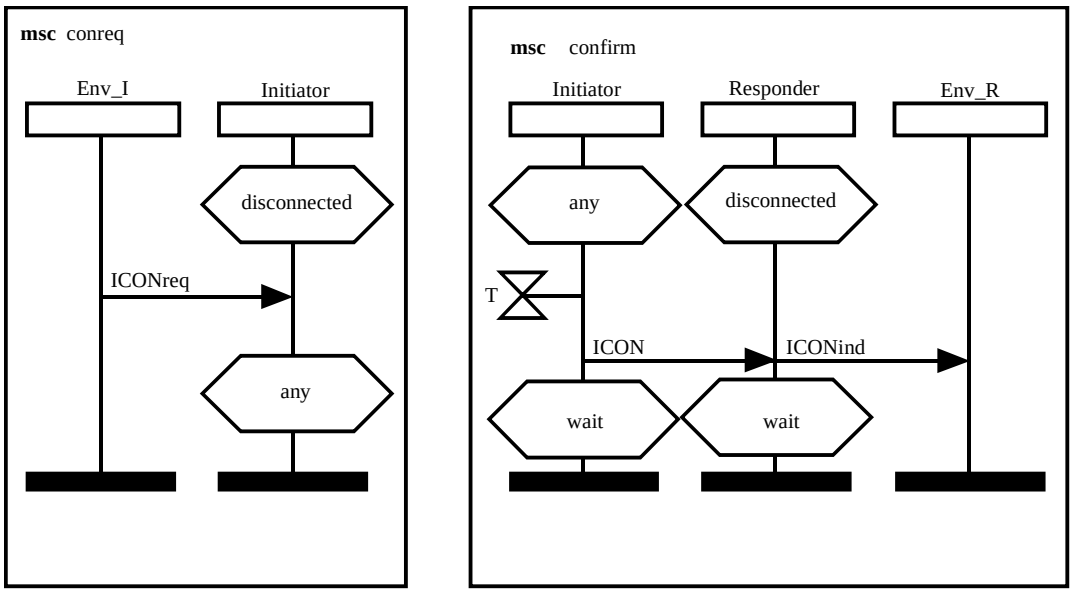


T1009580-96/d40

msc failure; **inst** Initiator, Responder;
instance Initiator;
condition Disconnected **shared all**;
in ICONreq **from env**;
set T;
out ICON **to lost** Responder;
condition Wait_For_Resp **shared all**;
timeout T;
out IDISind **to env**;
condition Disconnected **shared all**;
endinstance;
instance Responder;
condition Disconnected **shared all**;
in ICON **from** Initiator;
condition Wait_For_Resp **shared all**;
condition Disconnected **shared all**;
endinstance;
endmsc;

6.7 Conditions locales

Dans cet exemple, les conditions locales d'une instance sont utilisées pour indiquer les états locaux à cette instance.

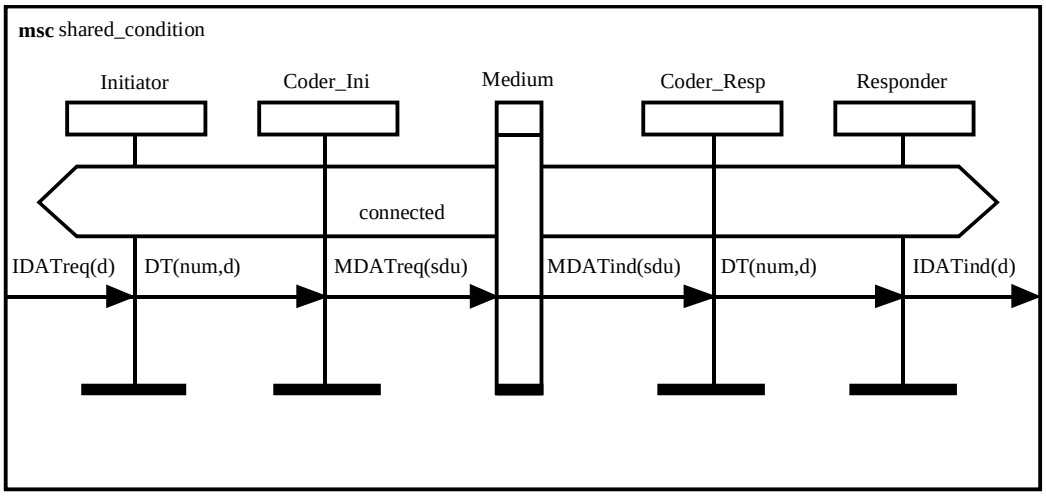



```
msc conreq; inst Env_I, Initiator;
        instance Env_I;
            out ICONreq to Initiator;
        endinstance;
        instance Initiator;
            condition disconnected shared;
            in ICONreq from Env_I;
            condition any shared;
        endinstance;
endmsc;


msc confirm; inst Initiator, Responder, Env_R;
        instance Initiator;
            condition any shared;
            set T;
            out ICON to Responder;
            condition wait shared;
        endinstance;
        instance Responder;
            condition disconnected;
            in ICON from Initiator;
            out ICONind to Env_R;
            condition wait shared;
        endinstance;
        instance Env_R;
            in ICONind from Responder;
        endinstance;
endmsc;
```

6.8 Condition partagée et messages avec paramètres

Cet exemple contient la condition partagée "connected". Cette condition est partagée par les instances "Initiator" et "Responder". Les instances "Coder Ini", "Medium", "Coder Resp" ne sont pas concernées par cette condition. Dans la représentation textuelle, le mot clé **shared** et la liste d'instances indiquent les instances auxquelles la condition est attachée.

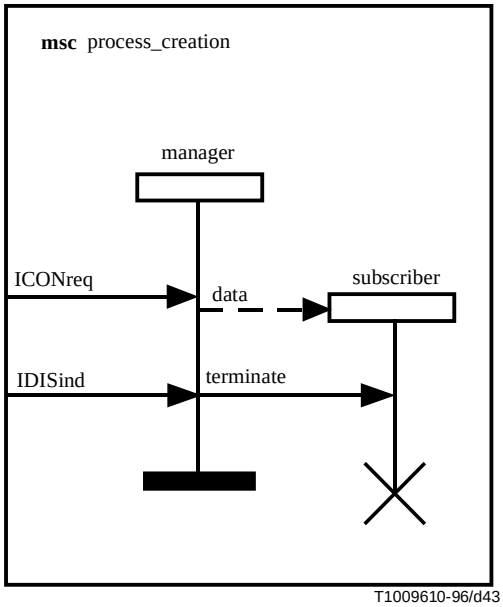


T1009600-96/d42

msc shared_condition; **inst** Initiator, Coder_Ini, Medium, Coder_Resp, Responder;
instance Initiator;
condition connected **shared** Responder;
in IDATreq(d) **from env**;
out DT(num,d) **to** Coder_Ini;
endinstance;
instance Coder_Ini;
in DT(num,d) **from** Initiator;
out MDATreq(sdu) **to** Medium;
endinstance;
instance Medium;
in MDATreq(sdu) **from** Coder_Ini;
out MDATind(sdu) **to** Coder_Resp;
endinstance;
instance Coder_Resp;
in MDATind(sdu) **from** Medium;
out DT(num,d) **to** Responder;
endinstance;
instance Responder;
condition connected **shared** Initiator;
in DT(num,d) **from** Coder_Resp;
out IDATind(d) **to env**;
endinstance;
endmsc;

6.9 Création et terminaison de processus

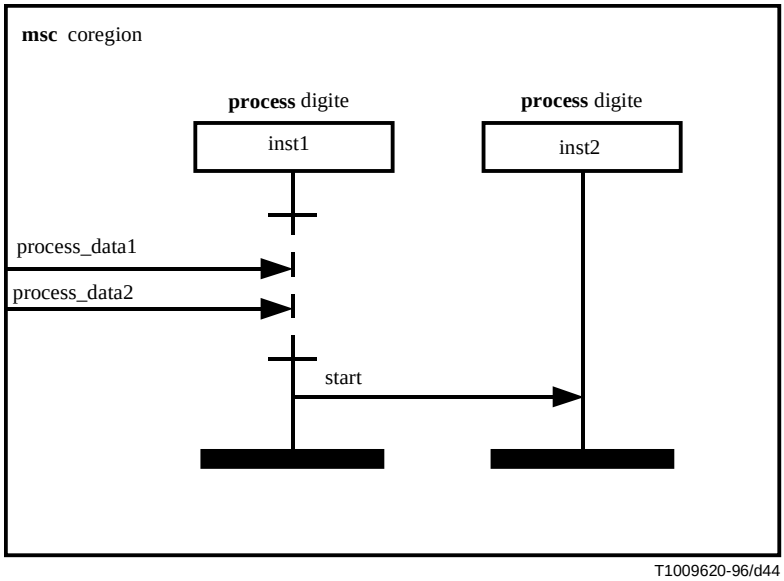
Cet exemple illustre la création dynamique de l'instance "subscriber" engendrée par une demande de connexion (connection request) et sa terminaison engendrée par une demande de déconnexion (disconnection request).




```
msc process_creation; inst manager, subscriber;
        instance manager;
            in ICONreq from env;
            create subscriber(data);
            in IDISind from env;
            out terminate to subscriber;
        endinstance;
        instance subscriber;
            in terminate from manager;
            stop;
        endinstance;
endmsc;
```

6.10 Corégion

Cet exemple montre une région de concurrence qui indique que la consommation de "process data1" et celle de "process data2" ne sont pas ordonnées dans le temps: "process data1" peut être consommé avant "process data2" ou inversement.

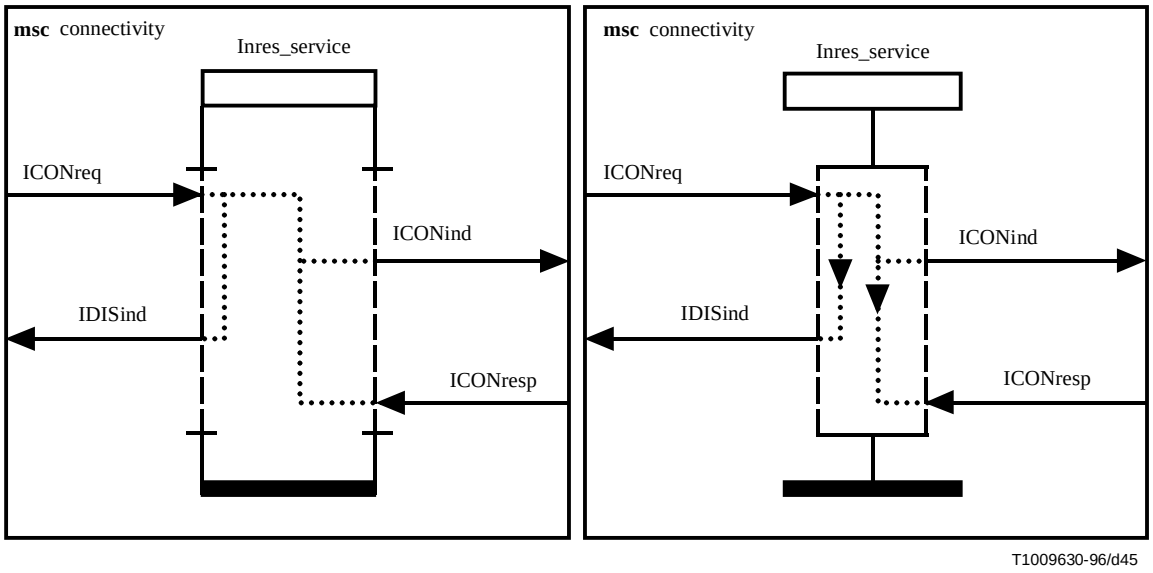


```
msc coregion; inst inst1, inst2;
        instance inst1: process digite;
            concurrent;
                in process_data1 from env;
                in process_data2 from env;
            endconcurrent;
            out start to inst2;
        endinstance;
        instance inst2: process digite;
            in start from inst1;
        endinstance;
endmsc;
```

6.11 Ordre généralisé dans une corégion

Cet exemple montre un ordre généralisé dans une zone de corégion au moyen de "connexions", c'est-à-dire que l'ordre est décrit à l'aide de connexions reliant les événements dans la corégion. Dans le MSC "connectivity" l'ordre suivant est défini: ICONreq < ICONind < ICONresp, ICONreq < IDISind.

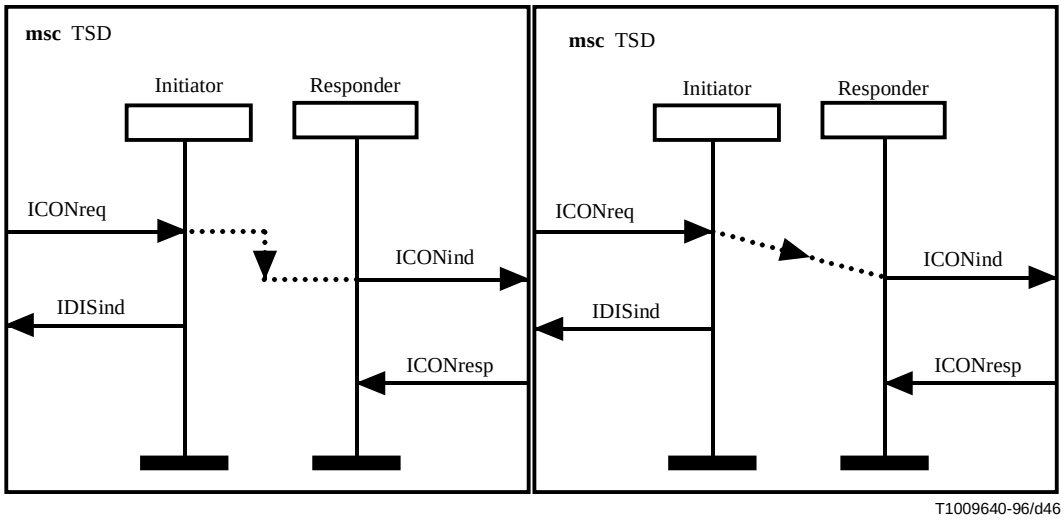
Dans ce cas, IDISind n'est pas ordonné par rapport à ICONind et ICONresp.




```
msc connectivity; inst Inres_service;
        instance Inres_service;
            concurrent;
                in ICONreq from env before Label1, Label2;
                Label1 out ICONind to env before Label3;
                Label2 out IDISind to env;
                Label3 in ICONresp from env;
            endconcurrent;
        endinstance;
endmsc;
```

6.12 Ordre généralisé entre différentes instances

Cet exemple montre l'utilisation des constructions de synchronisation issues des diagrammes de séquence du temps pour décrire l'ordre généralisé entre les différentes instances. La ligne (brisée ou oblique) entre le message entrant "ICONreq" et le message sortant "ICONind" définit l'ordre suivant: ICONreq < ICONind.

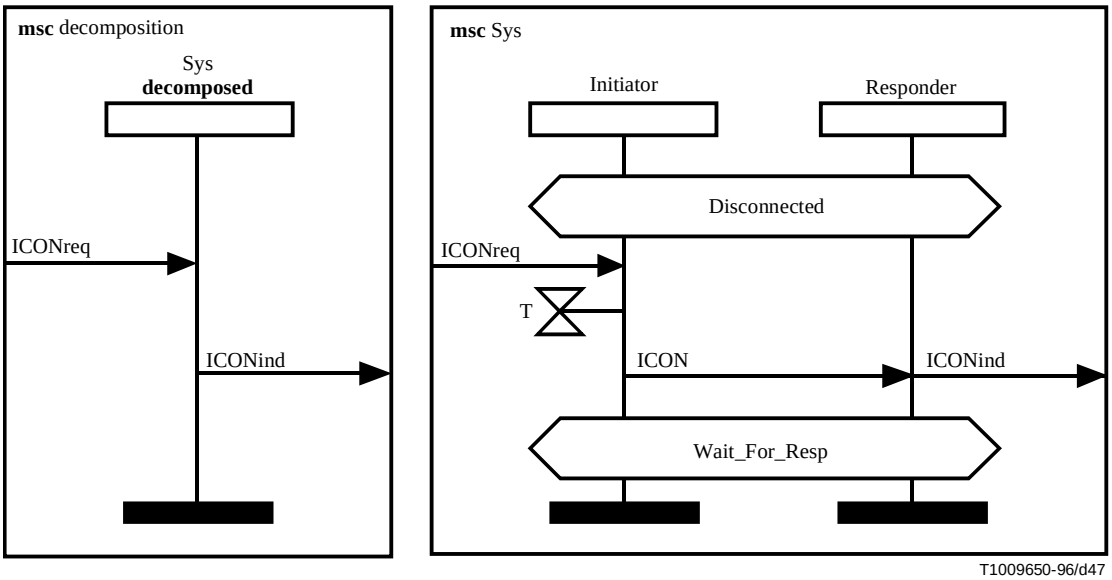


T1009640-96/d46

msc TSD; **inst** Initiator, Responder;
instance Initiator;
in ICONreq **from env before** Label1;
out IDISind **to env**;
endinstance;
instance Responder;
Label1 **out** ICONind **to env**;
in ICONresp **from env**;
endinstance;
endmsc;

6.13 Décomposition d'instance

Cet exemple contient le MSC affiné "Sys". Ce MSC est attaché à l'instance "Sys" représentant ainsi une décomposition de cette instance.



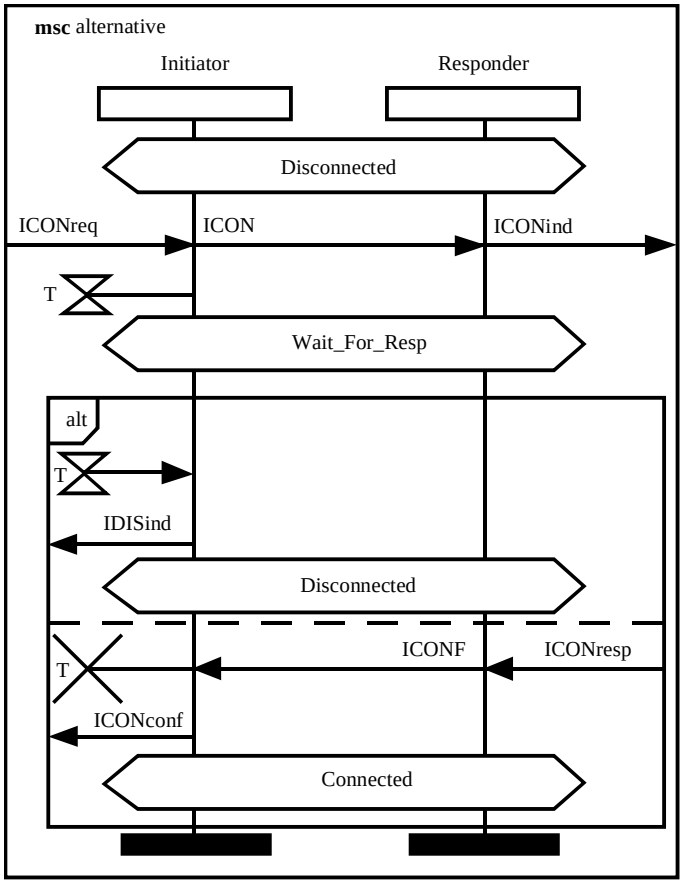
T1009650-96/d47

```
msc decomposition; inst Sys;
        instance Sys decomposed;
            in ICONreq from env;
            out ICONind to env;
        endinstance;
endmsc;

msc Sys; inst Initiator, Responder;
        instance Initiator;
            condition Disconnected shared all;
            in ICONreq from env;
            set T;
            out ICON to Responder;
            condition Wait_For_Resp shared all;
        endinstance;
        instance Responder;
            condition Disconnected shared all;
            in ICON from Initiator;
            out ICONind to env;
            condition Wait_For_Resp shared all;
        endinstance;
endmsc;
```

6.14 Expression en ligne contenant une alternative

Dans cet exemple, le cas d'une connexion réussie est combiné avec le cas d'une connexion échouant dans un MSC à l'aide de l'expression en ligne utilisant l'opérateur de l'alternative (MSC "alternative"). Dans le MSC "exception" la même situation est décrite à l'aide de l'expression en ligne équivalente utilisant l'opérateur de l'exception.

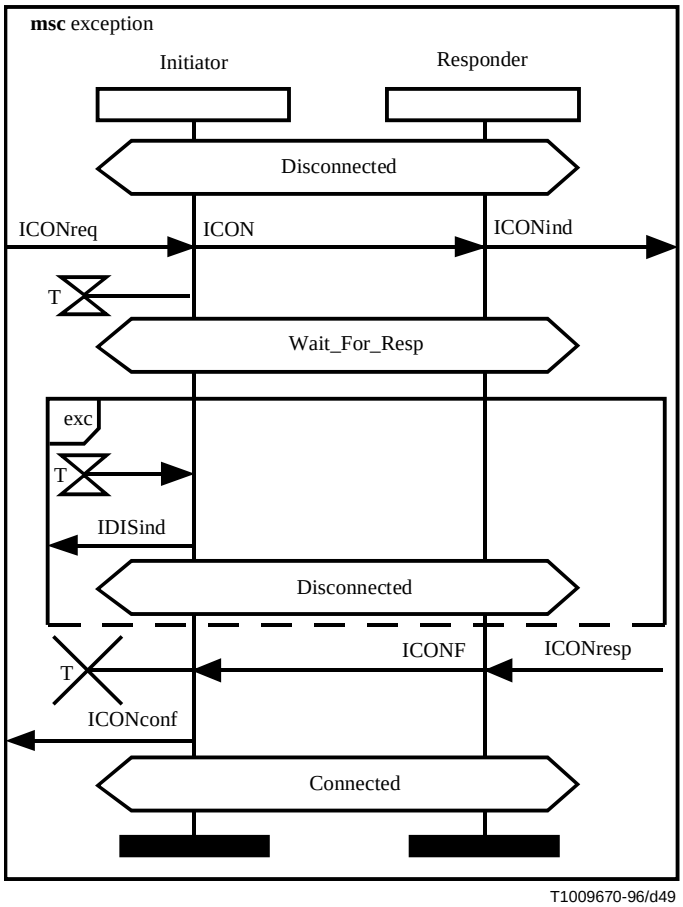


T1009660-96/d48

```
msc alternative; inst Initiator, Responder;
        Initiator:      instance;
        Responder:      instance;
        all:            condition Disconnected;
        Initiator:      in ICONreq from env;
                        out ICON to Responder;
                        set 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;
                            reset T;
                            out ICONconf to env;
            all:            condition Connected;
                        alt end;
        Initiator:      endinstance;
        Responder:      endinstance;
endmsc;
```

L'opérateur **exc** est une alternative où le second opérande est la fin du MSC comme le montre l'exemple suivant:

MSC "alternative" a exactement la même signification que le MSC "exception":



T1009670-96/d49

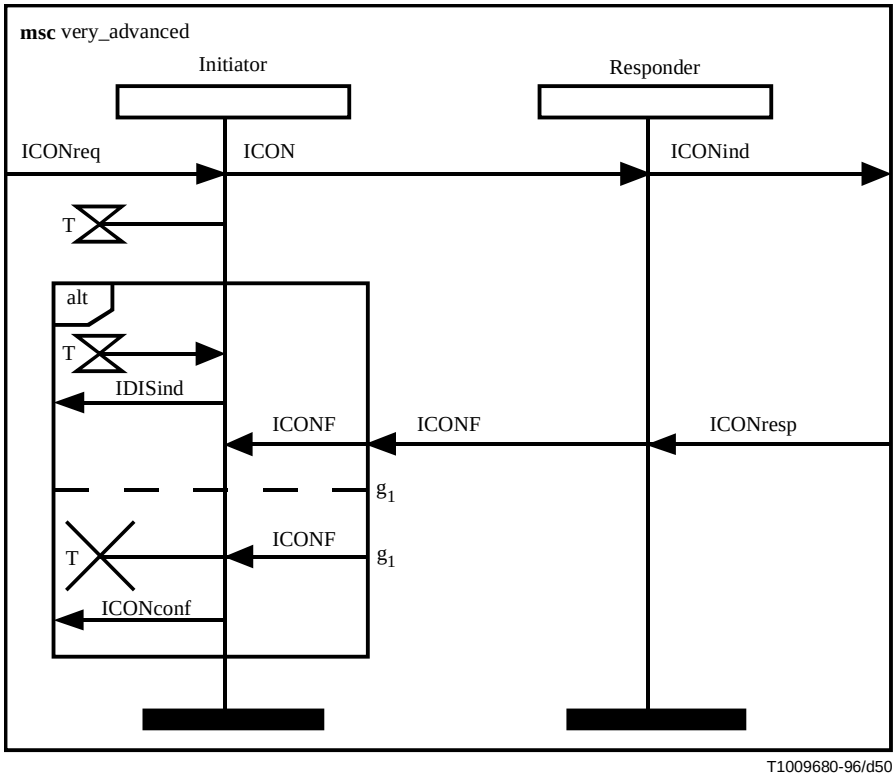
```
msc exception; inst Initiator, Responder;
        Initiator:      instance;
        Responder:      instance;
        all:            condition Disconnected;
        Initiator:      in ICONreq from env;
                        out ICON to Responder;
                        set T;
        Responder:      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;
                        reset T;
                        out ICONconf to env;
        all:            condition Connected;
        Initiator:      endinstance;
        Responder:      endinstance;
endmsc;
```

6.15 Expression en ligne avec des portes

Cet exemple décrit le même scénario que l'exemple 6.14 de façon légèrement différente: le message "ICONF" issu de l'instance "Responder" est connecté par les portes à l'expression en ligne utilisant l'opérateur de l'alternative attachée à l'instance "Initiator". Le message "ICONF" issu de l'instance "Responder" est propagé par la porte g1 (sur les deux alternatives) à l'instance "Initiator". Ceci décrit le fait que l'instance "Initiator" est en attente d'une réponse de la part de l'instance "Responder". Deux cas sont combinés dans le MSC "very_advanced":

a) l'instance "Responder" ne répond pas dans les temps: le message entrant "ICONF" est perdu après expiration du temporisateur et déconnexion de l'instance "Initiator";

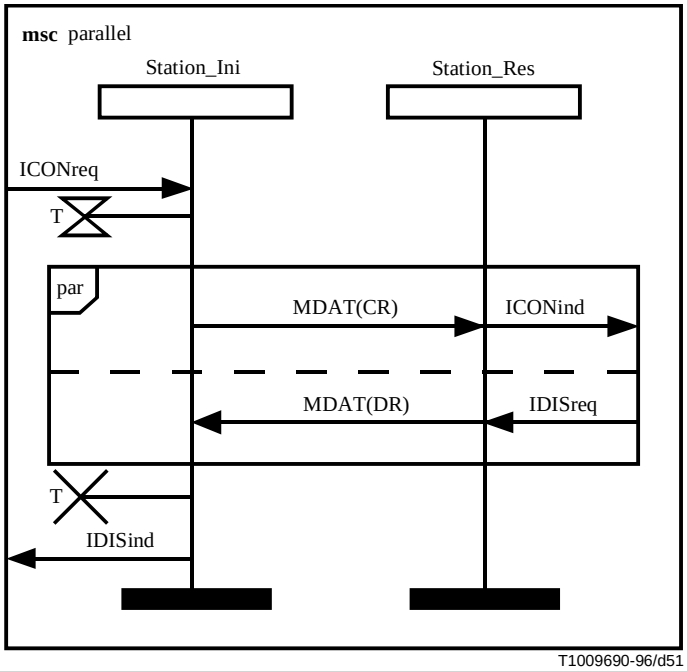
b) l'instance "Responder" répond en temps voulu conduisant à une connexion réussie.



```
msc very_advanced; inst Initiator, Responder; gate out ICONreq to Initiator;
gate in ICONind from Responder; gate out ICONresp to Responder;
        Initiator:      instance;
        Responder:      instance;
        Initiator:      in ICONreq from env;
                        out ICON to Responder;
                        set 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 in IDISind from Initiator;
                            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;
                            gate in ICONconf from Initiator;
                            in ICONF from env via g1;
                            reset T;
                            out ICONconf to env;
                        alt end;
        Initiator:      endinstance;
        Responder:      endinstance;
endmsc;
```

6.16 Expression en ligne utilisant une composition parallèle

Cet exemple illustre comment une expression en ligne parallèle décrit l'imbrication d'une demande de connexion de la part de l'instance "Station_Res", c'est-à-dire "MDAT(CR)" suivi de "ICONind", avec une demande de déconnexion de la part de l'environnement, c'est-à-dire "IDISreq" suivi de "MDAT(DR)".

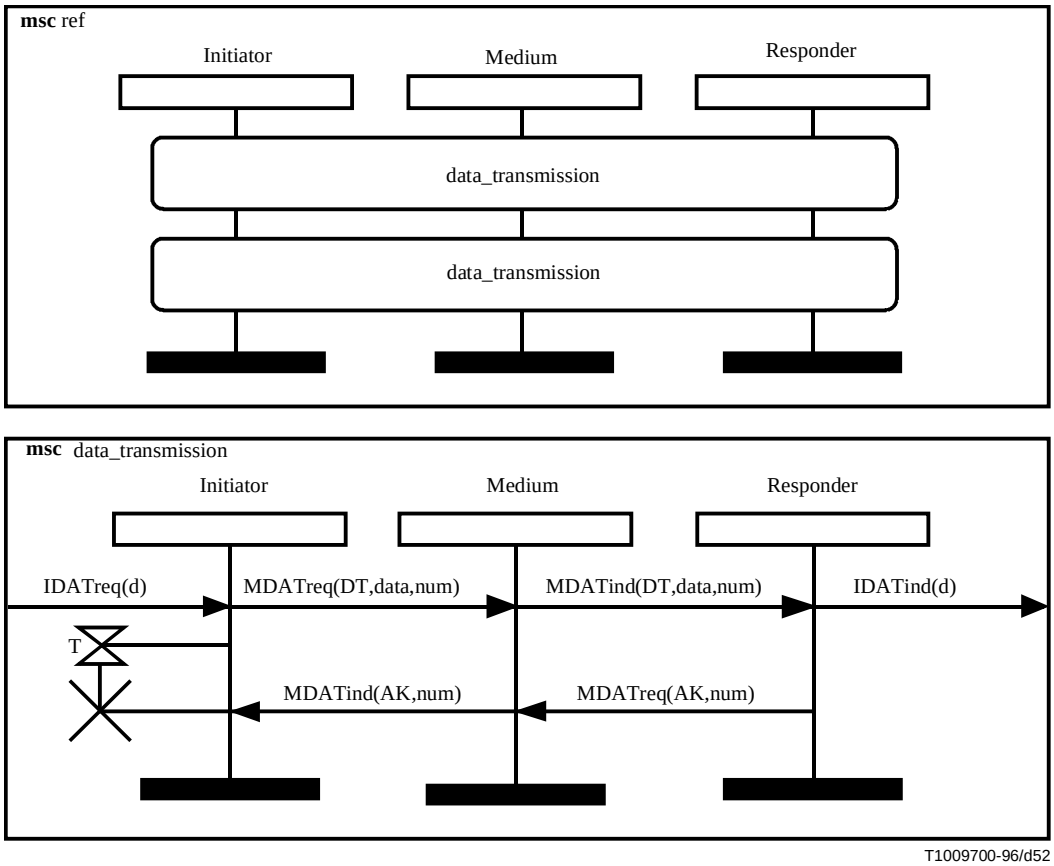


T1009690-96/d51

```
msc parallel; inst Station_Ini, Station_Res;
        Station_Ini:    instance;
                        in ICONreq from env;
                        set 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: reset T;
                        out IDISind to env;
                        endinstance;
endmsc;
```

6.17 Référence MSC

Dans cet exemple, la référence MSC "data_transmission" est utilisée pour décrire deux transmissions de données aboutissant à des succès. Cette référence correspond à la même définition MSC.

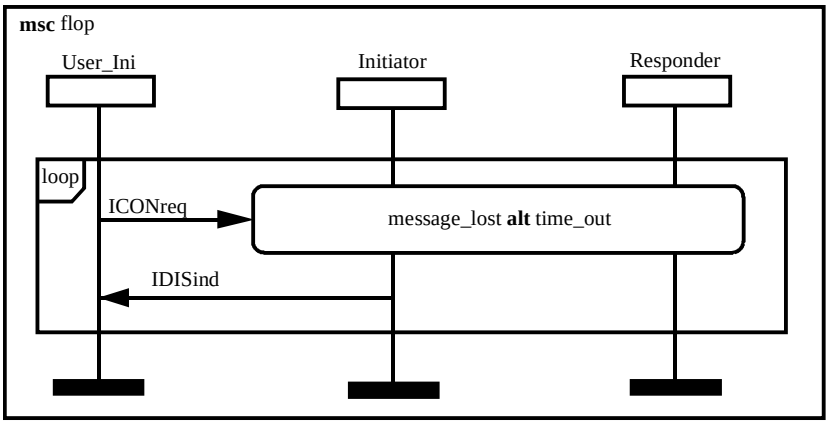



```
msc ref; inst Initiator, Medium, Responder;
        Initiator:      instance;
        Medium:         instance;
        Responder:      instance;
        all:            reference data_transmission;
                        reference data_transmission;
        Initiator:      endinstance;
        Medium:         endinstance;
        Responder:      endinstance;
endmsc;
```

```
msc data_transmission; inst Initiator, Medium, Responder;
        Initiator:      instance;
        Medium:         instance;
        Responder:      instance;
        Initiator:      in IDATreq(d) from env;
                        out MDATreq(DT,data,num) to Medium;
                        set 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;
                        reset T;
        Initiator:      endinstance;
        Medium:         endinstance;
        Responder:      endinstance;
endmsc;
```

6.18 Référence de MSC avec une porte

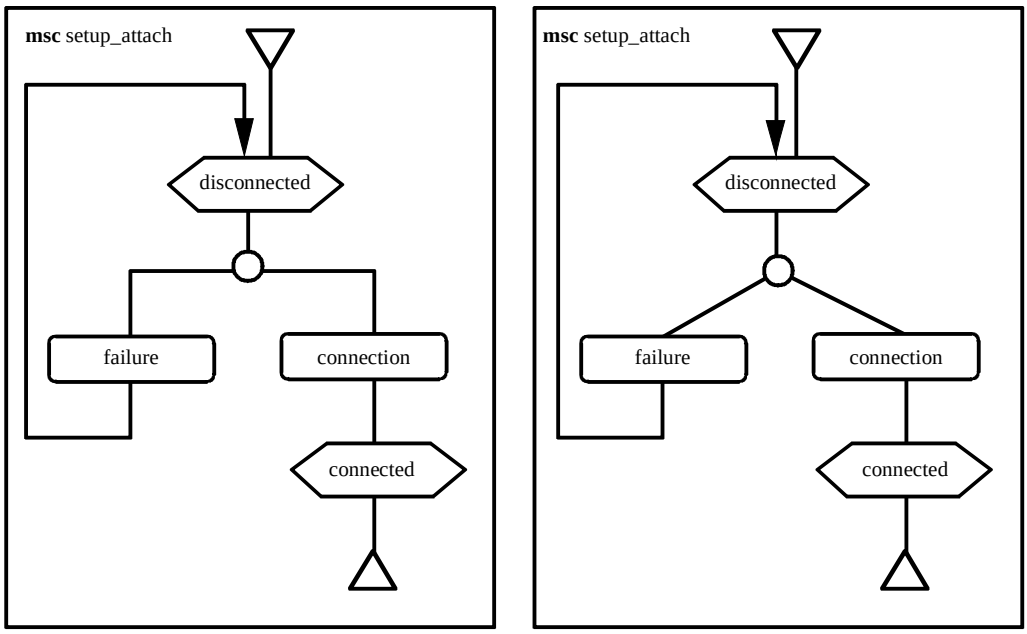
Cet exemple montre une référence MSC reliée avec l'extérieur par une porte. Les MSC référencés "message_lost" et "time_out" sont décrits dans l'exemple 6.21.




```
msc flop; inst User_Ini, Initiator, Responder;
        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;
            Initiator:      out IDISind to User_Ini;
            User_Ini:       in IDISind from Initiator;
                        loop end;
        User_Ini:       endinstance;
        Initiator:      endinstance;
        Responder:      endinstance;
endmsc;
```

6.19 MSC de haut niveau avec une boucle libre

Dans cet exemple le MSC "setup_attach" décrit le modèle des connexions à l'aide d'une boucle libre.

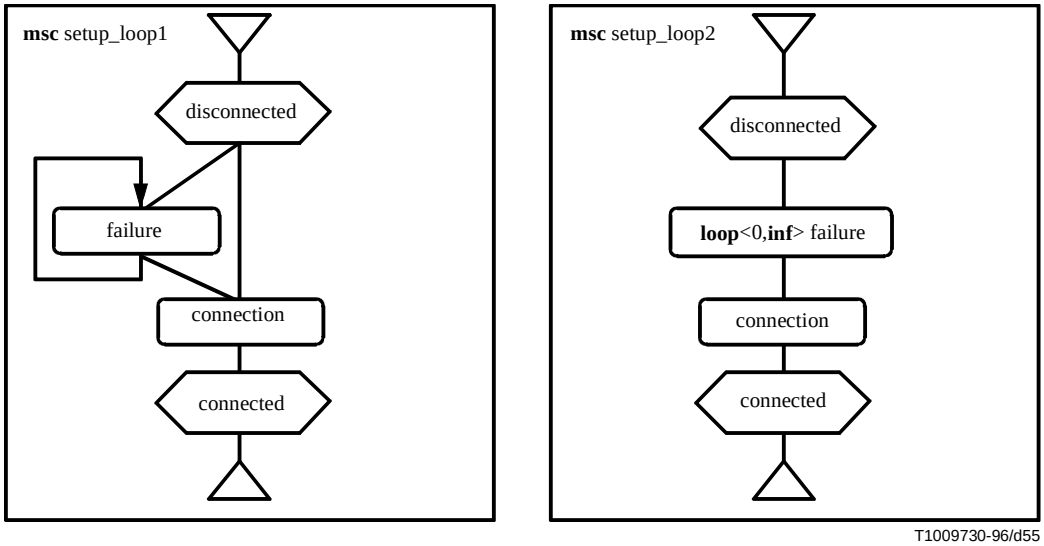


T1009720-96/d54

msc setup_attach;
expr L1;
L1: **condition** 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;

6.20 MSC de haut niveau avec une boucle

Dans cet exemple le MSC "setup_loop" décrit le modèle des connexions à l'aide d'une boucle attachée à la référence MSC "failure".



T1009730-96/d55

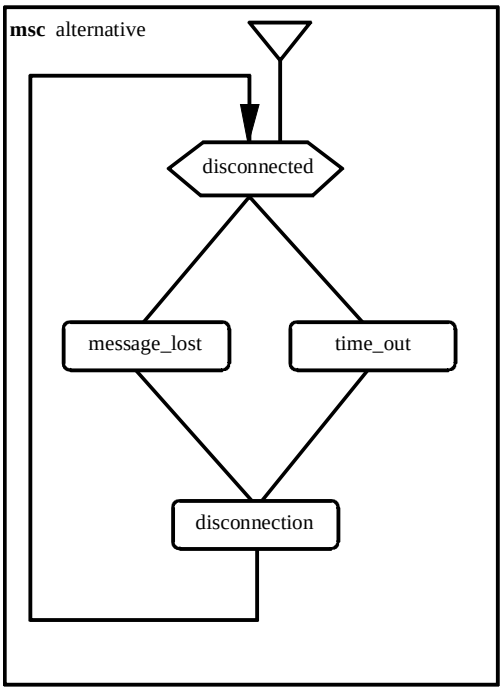
msc setup_loop1;
expr L1;
L1: **condition** disconnected **seq** (L2 **alt** L3);
L2: failure **seq** (L2 **alt** L3);
L3: connection **seq** (L4);

L4: **condition** connected **seq** (L5);
L5: **end**;
endmsc;

msc setup_loop2;
expr L1;
L1: **condition** disconnected **seq** (L2);
L2: (**loop** <0,**inf**> failure) **seq** (L3);
L3: connection **seq** (L4);
L4: **condition** connected **seq** (L5);
L5: **end**;
endmsc;

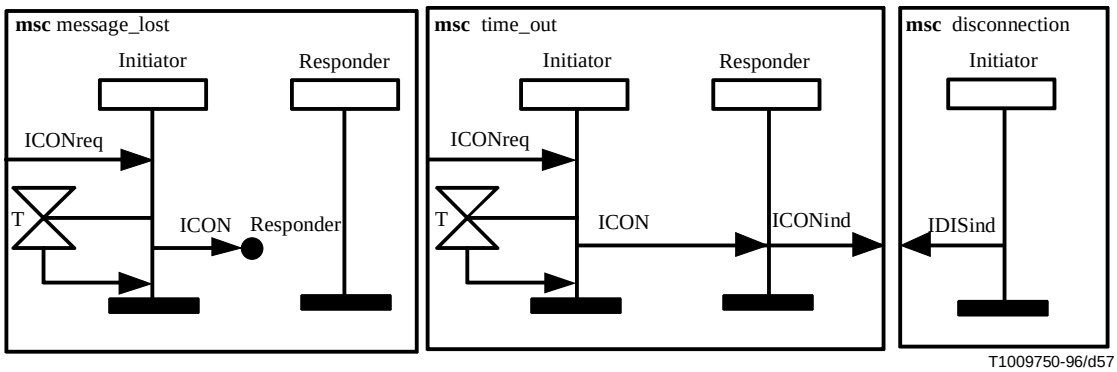
6.21 MSC de haut niveau avec une alternative

Dans cet exemple le MSC "alternative" montre une construction d'alternative contenant un bon "parenthésage": la construction de branchement possède sa propre construction d'union.



T1009740-96/d56

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;



T1009750-96/d57

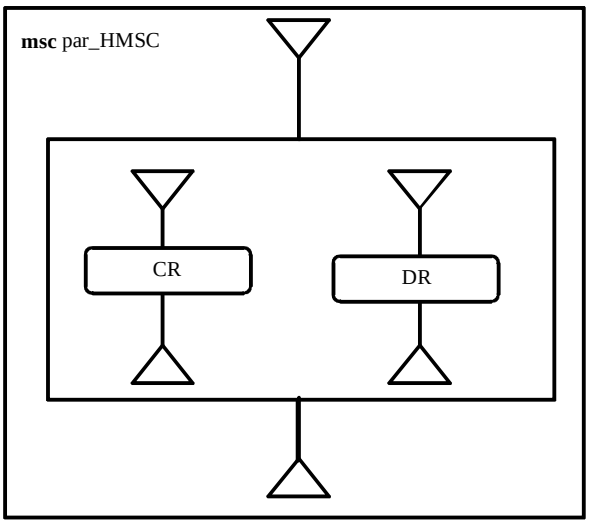
```
msc message_lost; inst Initiator, Responder;
        instance Initiator;
            in ICONreq from env;
            set T;
            out ICON to lost Responder;
            timeout T;
        endinstance;
        instance Responder;
            in ICON Initiator;
        endinstance;
endmsc;

msc time_out; inst Initiator, Responder;
        instance Initiator;
            in ICONreq from env;
            set T;
            out ICON to Responder;
            timeout T;
        endinstance;
        instance Responder;
            in ICON from Initiator;
            out ICONind to env;
        endinstance;
endmsc;

msc disconnection; inst Initiator, Responder;
        instance Initiator;
            out IDISind to env;
        endinstance;
endmsc;
```

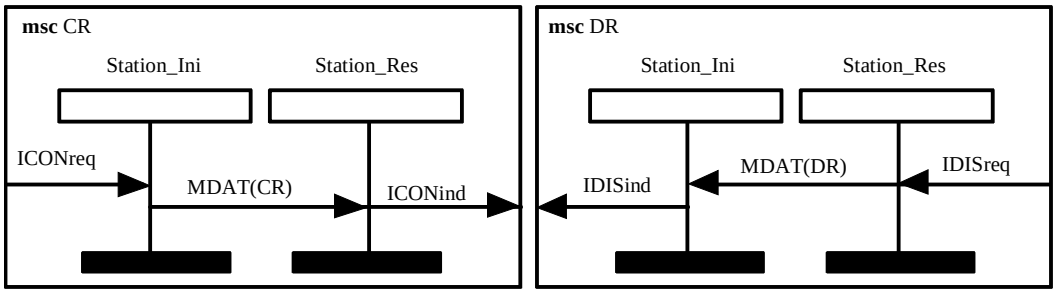
6.22 MSC de haut niveau avec une composition parallèle

Dans cet exemple, le parallélisme dans le diagramme HMSC est utilisé pour exprimer le fait que la demande de connexion de la part de l'"Initiator" et la demande de déconnexion de la part du "Responder" sont exécutées en parallèle.



T1009760-96/d58

```
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;
```



T1009770-96/d59

```
msc CR; inst Station_Ini, 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, Station_Res;
        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;
```

Annexe A

Index

Cet index n'est publié que dans la version anglaise de la présente Recommandation.

Remplacée par une version plus récente

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	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	Maintenance: systèmes de transmission, de télégraphie, de télécopie, circuits téléphoniques 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 pour données et communication entre systèmes ouverts
Série Z	**Langages de programmation**