Titre: Title:	L'analyse des services téléphoniques dans le cadre de la théorie de la commande des systèmes à événements discrets
Auteur: Author:	Anka Munteanu
Date:	1998
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Munteanu, A. (1998). L'analyse des services téléphoniques dans le cadre de la théorie de la commande des systèmes à événements discrets [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6914/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6914/
Directeurs de recherche: Advisors:	John G. Thistle
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

L'ANALYSE DES SERVICES TÉLÉPHONIQUES DANS LE CADRE DE LA THÉORIE DE LA COMMANDE DES SYSTÈMES À ÉVÉNEMENTS DISCRETS

ANKA MUNTEANU

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE ÉLECTRIQUE)

AVRIL 1998

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

L'ANALYSE DES SERVICES TÉLÉPHONIQUES DANS LE CADRE DE LA THÉORIE DE LA COMMANDE DES SYSTÈMES À ÉVÉNEMENTS DISCRETS

présenté par: MUNTEANU Anka

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

M. BOIS Guy, Ph.D., président

M. THISTLE John, Ph.D., membre et directeur de recherche

Mme BERGERON Anne, Ph.D., membre

DÉDICACE

À ma fille Alexandra

REMERCIEMENTS

Je tiens à remercier en premier lieu le professeur John Thistle pour son support scientifique à toutes les étapes de cette recherche. C'est grâce à ses conseils que j'ai réussi à comprendre ce domaine et à trouver des solutions aux problèmes générés par la nouveauté du sujet abordé.

J'étais inspiré aussi par les suggestions du professeur Roland Malhamé qui a toujours souligné l'avenir de cette recherche.

Mes travaux ont été achevés avec la collaboration de Mme Nathalie Rico de Nortel, qui nous a fourni en tous temps les documents et les informations nécessaires pour pouvoir analyser des services téléphoniques réels.

RÉSUMÉ

L'objectif principal de ce mémoire est de modéliser des services téléphoniques réels et leurs superviseurs dans le cadre de la théorie des systèmes à événements discrets (SED). En raison de leur grand nombre et leur complexité, les services téléphoniques sont conçus de façon répartie et modulaire. Par conséquence, ces services peuvent interagir de manière imprévue et indésirable. C'est pour cette raison qu'on développe actuellement des méthodes formelles pour les décrire et pour régler les conflits éventuels qui peuvent se présenter parmi un ensemble donné de services. Les travaux présentés dans ce mémoire s'inscrivent dans un domaine où il y a déjà beaucoup d'approches mais peu d'exemples d'implantations.

Le service téléphonique étudié en détail est CSMI (Call Screening, Monitor and Intercept), récemment offert par la compagnie Nortel. La formalisation de la spécification CSMI sera suivie par la synthèse de son superviseur et une analyse des résultats.

Une grande partie du travail est centrée sur l'élaboration de l'automate de spécification CSMI. Dans l'esprit de la théorie SED, cet automate devrait retenir toute l'information contenue dans le document du service tout en restant simple. La complexité de CSMI est due aux deux aspects de ce service: 1) la messagerie et 2) l'écoute et interception de messages. On met aussi en évidence les caractéristiques de modélisation des SED. Toutes les informations concernant le mini-réseau téléphonique sont traduites en termes de

langages générés à partir d'un alphabet spécifique. L'étape suivante est le calcul du plus grand sous-langage commandable qui représente la solution du problème de contrôle du commutateur téléphonique, en présence de ce service. On fait ensuite la comparaison avec une autre procédure de modélisation qui suit les étapes de déroulement du service indiquées dans le document Nortel. La comparaison de ces deux approches donne des informations utiles pour le développement d'une méthodologie de modélisation.

On se consacre ensuite à la présentation d'un exemple d'interférence entre CSMI et CW (Call Waiting). En effet, nous analysons quel est l'état critique où l'interférence arrive. Nous synthétisons par la suite deux "filtres", un pour chaque superviseur, qui ont le rôle de cacher certains événements qui mènent à l'interférence, sans les empêcher ou les annuler. En même temps, on essaie de mettre en évidence certaines règles pour ce processus de filtrage. Les derniers calculs démontrent la fonctionnalité du commutateur téléphonique sous le contrôle des superviseurs modifiés par les filtres.

ABSTRACT

The main goal of this thesis is to model real telephone features in the Discrete Event Systems theory (DES) framework. Due to their increasing number, features are designed in a modular way and their implementation is distributed, which leads to unexpected and undesirable interactions. It is widely accepted today that the resolution of feature interactions requires the use of formal methods.

In this thesis we study a new feature offered by NORTEL , called CSMI (Call Screening, Monitor and Intercept). We formalize the specification of this feature and analyse a supervisor computed for it. We give detailed explanations of the synthesis of the CSMI specification as well as the role and the meaning of the states and the events that appear in this automaton. The complexity of the service is dueto its two aspects: the voice mailbox function and the monitor and interception of the call. A model of a small telephone network is represented in terms of a formal language, whose elements are strings of symbols of a specific alphabet. The synthesis of the feature specification is followed by a computation of the greatest controllable sublanguage which is a solution to the control problem at the telephone switch level. The next step is the comparison of this synthesis procedure with another procedure based on the specification document provided by Nortel, which follows all the steps in the monitoring and interception of a call.

We continue with the presentation of an interaction example between CSMI and CW. In

our approach the priority given to a feature depends on the event that takes the system to a conflict situation. We identify this critical event and then synthesize two filters ("reporter maps"), one for each supervisor, which mediate the observation and control performed by the respective supervisors. Meanwhile, we identify certain rules for the filtering process. The final computations show the functionality of the telephone switch under the control of the supervisors in the presence of filters.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

Options de service téléphonique:

AC - Answer Call

ARC - Automatic ReCall

CSMI - le service téléphonique Call Screening, Monitor and Intercept

CW - le service téléphonique Call Waiting

CFDA - Call Forward Don't Answer

DMS - Digital Multiplexing Switch

DN - Directory Number

FSM - Finite State Machine

NBAS - Network-Based Answering System

OCS - le service téléphonique Originating Call Screening

POTS - Plain Old Telephone Service

PSTN - Public Switched Telephone Network

R - W - Ramadge - Wonham

SED - systèmes à événements discrets

SIM - Service Interaction Management

TCS - le service téléphonique Terminating Call Screening

VMNSS - Voice Message Network Support Services

Événements dans le modèle du réseau téléphonique:

on_hx - on hook x : raccrochage de la part de l'abonné "x" (x = i, j, k)

off_hx - off hook x : décrochage de la part de l'abonné "x" (x = i, j, k)

con_xy - connexion "x" - "y" : connexion entre les abonnés "x" et "y"

(x, y = i, j, k, x≠ y)

no_con xy - tonalité d'occupation entre les abonnés "x" - "y" (x, y = i, j, k)

fwd_xyz - appel généré par l'abonné "x" vers l'abonné "y" et renvoyé vers l'abonné "z"

LISTE DES FICHIERS

CHAPITRE II

FIL: la modélisation du renvoi de l'appel au commutateur "i"

FIM: la version marquée du fichier FIL

FJL: la modélisation du renvoi de l'appel au commutateur "j"

FJM: la version marquée du fichier FJM

FKL: la modélisation du renvoi de l'appel au commutateur "k"

FKM: la version marquée du fichier FKM

FLAL: la spécification "flash_i"

FLALM: la version marquée de FLAL

G: le générateur (le modèle du mini-réseau téléphonique)

G = sync (SW_I, SW_J, SW_K)

G_I: la projection locale (au commutateur "i") du générateur G (le générateur local); G_I = project (G, LIST)

GIM: le générateur local enrichi avec la fonction "flash"

GIM = meet (G_I, FLALM)

GM: la version marquée du fichier G

GM = sync (MI, MJ, MK)

GIM_M: la projection locale au commutateur "i" du GM

GIM_M = project (GM, LIST)

où LIST comprend tous les événements qui ne font pas partie de l'alphabet Σ_i.

MI: la version marquée du fichier SW_I

MI = meet (SWIM, FIM)

MJ: la version marquée du fichier SW_J

MJ = meet (SWJM, FJM)

MK: la version marquée du fichier SW_K

MK = meet (SWKM, FKM)

SWI: le modèle local simple du commutateur "i"

SW_I: le modèle local complet du commutateur "i"

SW_I = meet (SWI, FIL)

SWIM: la version marquée du fichier SWI

SWJ: le modèle local simple du commutateur "j"

SW_J: le modèle local complet du commutateur "j"

SW_J = meet (SWJ, FJL)

SWJM: la version marquée du fichier SWJ

SWK: le modèle local simple du commutateur "k"

SW_K: le modèle local complet du commutateur "k"

SW_K = meet (SWK, FKL)

où LIST comprend tous les événements qui ne font pas partie de l'alphabet Σ_i.

SWKM: la version marquée du fichier SWK

CHAPITRE III

CSMI: la spécification simple du service (figure 3.2)

CSMI2: la spécification complexe du service (figure 3.3)

CSMI3: la spécification complexe du service, version finale (figure 3.4)

CSMI4: la spécification simple du service, version finale (figure 3.5)

SUP_CSMI: le superviseur correspondant à la spécification CSMI

SUPCSMI4: le superviseur correspondant à la spécification CSMI4

CHAPITRE IV

CSMI41R: spécification CSMI4 avec des transitions en majuscule dans un état supplémentaire

CSMI41RM: la version marquée de CSMI41R

CSMI_R: l'automate SYNCSMI avec tous les événements en notation normale (sans majuscules)

CSMI_R2: la version reétiquettée de SYCSMI2

CWS: la version simple du superviseur CW

CWSR: l'automate CWS avec les événements en majuscule bouclés dans un état supplémentaire

CWSRM: la version marquée de CWSR

CW_R: l'automate SYNCCW avec tous les événements en notation normale

CW_R2: la version reétiquettée de SYNCCW2

RCSMI: automate RP avec les événements du filtre CSMI en majuscule

RCW: l'automate RP avec les événements du filtre CW en majuscule

RP: l'ensemble des filtres CSMI et CW (reporter maps)

SYNCSMI: le produit synchrone de RCSMI et CSMI41R

SYNCCW: le produit synchrone des automates RCW et CWSR

SYCSMI2: le produit synchrone de RCSMI et CSMI41RM

SYNCCW2: le produit synchrone de RCW et CWSRM

Première étape

S_CSMI1: le nouveau superviseur CSMI

S_CW1: le nouveau superviseur CW

Deuxième étape

S_CSMI2: le nouveau superviseur CSMI

S_CW2: le nouveau superviseur CW

Troisième étape

S_CSMI3: le nouveau superviseur CSMI

S_CW3: le nouveau superviseur CW

CHAPITRE I

INTRODUCTION

En raison de leur grand nombre et de leur complexité, les options de service téléphoniques posent des problèmes spéciaux d'implantation. L'un des effets de l'accumulation des options de service est le phénomène "d'interférence" ou d'interactions indésirables. L'interférence entre options de service est un phénomène général des systèmes répartis qui se présente dans des contextes autres que le traitement des appels téléphoniques. À cause de l'ampleur du problème, des efforts substantiels ont été déjà faits pour définir un cadre théorique adéquat pour la modélisation des services et la détection de leurs interférences. Toutes les études cherchent une solution théorique pour éviter la propagation des interférences à l'étape d'implantation, où elles causent des perturbations importantes.
On doit mentionner que la définition même de ces concepts est difficile et en évolution. Les descriptions ambiguës ou incomplètes des services sont parfois des sources d'interférence. Ce chapitre présente quelques approches nouvelles à ce sujet.

1.1 La problématique des interférences dans le domaine des services téléphoniques: types d'interférences et exemples

L'interférence est définie comme une interaction entre un ensemble de règles de fonctionnement implantées dans un commutateur téléphonique. Elle est étudiée et traitée dans la littérature de plusieurs points de vue, comme la spécification du service, la

synthèse du protocole, l'implantation du service, etc. Une autre distinction serait celle entre les interférences des services pour clients (customer features) et interférences des fonctions du système téléphonique (system features), (Cameron, E.J. et al, 1994). En se limitant à la première catégorie, on peut classifier ces interférences d'après le nombre d'utilisateurs qui participent dans l'appel, comme suit: interférences "Single-User-Single-Component" , interférences "Single-User-Multiple-Componen"t, interférences "Multiple-User-Single-Component" et interférences "Multiple-User-Multiple-Component". Les interférences "Single-User" arrivent quand un seul abonné utilise plusieurs services en même temps. Les interférences "Single-Component" se produisent quand un seul composant du réseau (commutateur, noeud de service) s'occupe de plusieurs services en même temps. Comme les chapitres suivants traiteront un exemple "Single-User-Single-Component", nous énumérons ci-dessous quelques exemples.

Une interférence de type "Single-User-Single-Component" serait celle entre les services CSMI (Call Screening, Monitor and Intercept) et CW (Call Waiting) analysés au chapitre 4. Nous observons que l'interprétation de l'événement "flash" est différente pour les deux services: pendant la période d'écoute de CSMI, le "flash_i" executé par l'abonné qui écoute son message génère une interception de l'appel en question. Cependant, le même événement pourrait être interprété par CW comme une demande de commutation de la connexion vers le nouvl appel.

Les services TWC (Three Way Calling) et "911" peuvent être eux aussi en conflit de la

manière suivante: l'abonné qui utilise TWC doit mettre son appel en attente pour pouvoir être capable de se connecter à un deuxième appelant. En même temps, le service "911" ne permet pas cette opération. Si un abonné A veut aider B en lui faisant la connexion à "911", il pourrait le faire seulement s'il appelle premièrement B, le met en attente et ensuite il appelle "911".

Les services CW et TWC (ou MWC: Multi Way Calling) interfèrent de la façon suivante: un "flash-hook" généré par un abonné déjà engagé dans une connexion est interprété par TWC comme une tentative de se connecter à un troisième participant; cependant CW l'interprète comme une mise en attente pour la connexion courante et l'acceptation d'un nouvel appel qui était en attente. Supposons que, pendant une conversation entre A et B, il arrive un appel de la part de C qui veut parler à A. Si A est abonné au CW, il sera avisé par la tonalité CW de cette situation. Il est possible quand même que A a fait un "flash". Dans cette situation, il n'est pas évident lequel des deux services interprétera cet événement en premier et comment évoluerent en conséquence les connexions respectives de A avec B et C.

Enfin, les services CW et AC (Answer Call) interfèrent au moment où un nouvel appelant essaie de rejoindre un abonné occupé. À ce moment, AC fait la connexion de ce dernier appelant à une messagerie et CW ne peut plus générer sa tonalité spécifique pour signaler qu'il y a un appel en attente. Supposons que l'abonné A possède les deux services. Si A parle à quelqu'un au moment où un deuxième appel arrive, il n'est pas clair si A devrait

être avisé par une tonalité CW ou bien si cet appel devrait être renvoyé vers sa messagerie.

1.2 Théories et essais de formalisation.

Nous nous intéressons principalement aux approches qui modélisent le niveau d'abstraction de la spécification du service téléphonique. On constate qu'une bonne partie de la recherche est dédiée à l'étude de cette première étape pour garantir des spécifications cohérentes. Une tendance générale de ce point de vue est de modéliser le réseau téléphonique par des systèmes à événements discrets (SED). On considère qu'une interférence se manifeste quand un événement généré dans le commutateur téléphonique déclenche des transitions dans deux ou plusieurs services, avec des effets différents (Cheng, 1995).

Dans les sections suivantes nous présentons quelques approches à la modélisation des services téléphoniques comme celle basée sur le langage de spécification Delphi, la modélisation avec règles de production dans le langage LOTOS et enfin le modèle "évitant-l'interférences et les "négociateurs".

1.2.1 La modélisation et l'interférence des services téléphoniques dans le langage de spécification Delphi

Le langage Delphi (Bostrom, 1995) a été conçu pour être utilisé dans les premières étapes

de la synthèse des services téléphoniques. Une spécification Delphi consiste en une partie statique qui est le modèle conceptuel (MC) et une partie dynamique qui représente les règles de fonctionnement du système. Le modèle est basé sur des "entités", des "proprietés d'entités" et des "relations" entre entités. Un exemple d'une entité serait "pots_subscriber" qui représente un abonné quelconque. Cette entité est associée à des "déclarations d'états" (ou propriétés) telles que "start", "idle", etc. ainsi qu'à deux relations: "en appel"("calling") et "en conversation"("speaking"). Dans l'exemple de spécification Delphi qui suit, nous représentons les règles du service POTS (Plain Old Telephone Service) qui est la fonction de base du commutateur téléphonique:

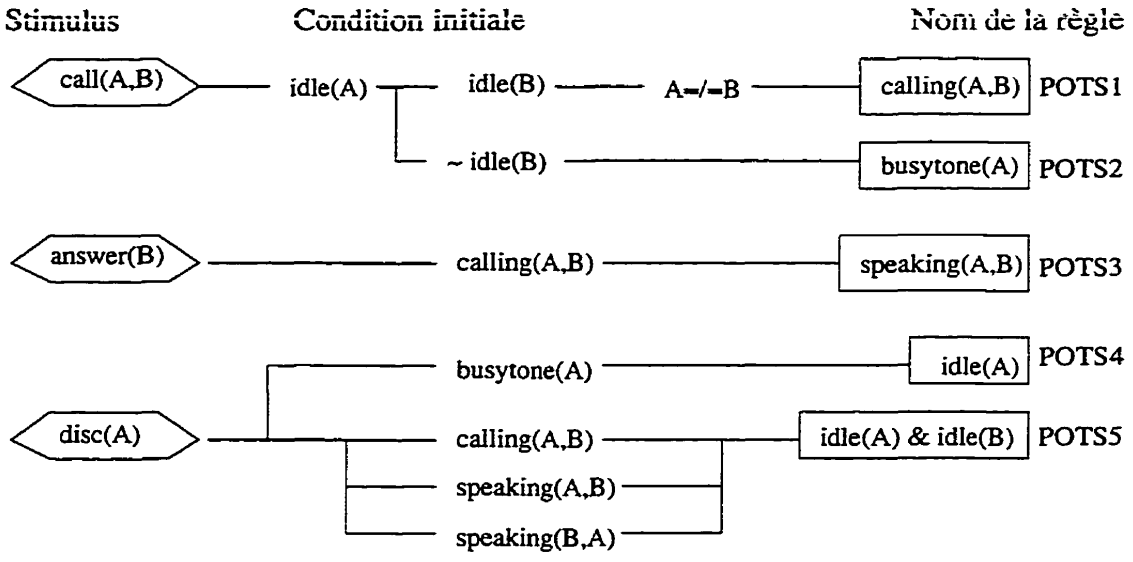


Figure 1.1 Le service téléphonique de base (POTS)

Nous voulons mentionner les cinq règles, nommées POTS1,...,POTS5. Les deux premières, POTS1 et POTS2, sont déclenchées par le même stimulus, "call(A, B)", qui représente l'appel de A vers B. Elle partagent aussi la condition initiale "idle(A)" qui

signifie que A est accroché. POTS1 finit par la conclusion "calling(A, B)" et POTS2 qui représente le cas où B est occupé, finit par "busytoneA" (tonalité d'occupation envoyé vers A). L'ajout de services (features) est représenté par des nouvelles branches qui peuvent être ajoutées à n'importe quel point dans la règle.

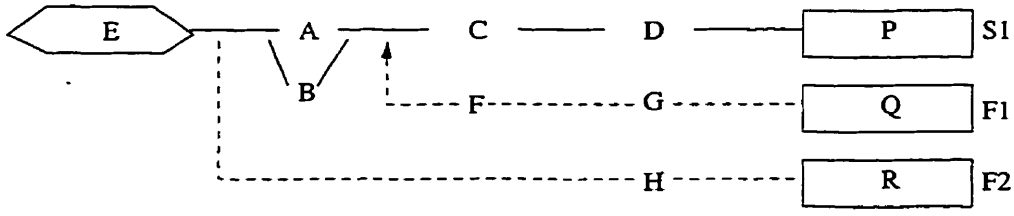


Figure 1.2 L'extension du service de base S1 avec des nouvelles branches F1, F2

Ces nouvelles branches sont appelées "non-monotones" si elles modifient la conclusion de la branche principale (comme par exemple F1). Modifier la conclusion d'une branche veut dire dans ce contexte qu'une branche du service de base POTS (S1) est influencée par la nouvelle branche et réciproquement, la branche de base influence le fonctionnement de la nouvelle branche. Une branche "non-monotone" restreint la fonctionnalité de la branche de base où elle est attachée. La branche "non-monotone" sera en relation d'exclusion mutuelle avec la branche principale et ses fonctions seront prioritaires par rapport aux celles de cette branche. La notion de "monotonié" indique l'absence d'interférence entre les nouvelles branches et les branches de base.

La spécification du service de base enrichi avec quelques autres services s'appelle "spécification conjointe" (joint spécification) qui est utilisée pour la détection des interférences. On défini aussi une relation de priorité ">" entre les branches d'une même

règle. Les interférences seront résolues en accordant la priorité à l'un des deux services en conflit, à un moment particulier dans le déroulement de l'appel.

Si l'on souhaite à ajouter des services, on complète le graphe ci-dessus avec des nouvelles règles, représentées par des nouvelles branches. L'interférence se manifeste formellement par l'activation de deux branches à l'application d'un seul stimulus, ce qui implique l'obtention de deux conclusions différentes. Les auteurs Bonstrom et Engstedt (1995) soulignent que la représentation des règles est difficile pour une collection de services, même si le modèle conceptuel est assez facile à représenter. Dans ce cas, on opère avec des "règles jointes" et le traitement est fait en attribuant des priorités aux règles.

1.2.2 Méthodes nouvelles de détection et d'élimination des interférences

Les auteurs Kawarasaki et Otha (1995) proposent la représentation des services téléphoniques et la résolution des interférences par des "règles de production" (production rules) et ils choisissent comme outil de calculs le langage formel des réseaux de Petri. Ils expliquent ce choix par l'impossibilité d'exprimer avec des SED certaines causes de l'apparition des interactions et certains types d'interactions entre services. Ils donnent quelques exemples de telles interférences: selon eux, la modélisation par SED crée les soit-disantes "transitions illégales". Une transition illégale est une nouvelle transition obtenue par la combinaison des états des plusieurs services. Ils utilisent aussi la notion d'"état perdu" qui serait un état atteignable dans un des services avant la composition des services et non atteignable après cette opération. Par exemple, si on applique une règle

prioritaire par rapport à une autre, on perd la chaîne de transitions de la règle de basse priorité. Évidemment, les priorités des règles de ce système sont fixes.

Enfin, un troisième problème serait les "transitions perdues". Elle sont des transitions qui existaient avant la conjonction des services et qui sont perdues après cette opération. Après avoir mis en évidence ces problèmes, les auteurs présentent quelques autres situations génératrices d'interférences liées cette fois au fait que la transmission de l'information dans le réseau n'est pas instantanée. Donc un événement peut être considéré antérieur ou postérieur à l'autre dans un état du système, ce qui engendre différentes suites de transitions, selon le cas. Ils nomment les SED par FSMs (Finite State Machines) et ils proposent une représentation des services par des "règles de production" qui prennent la forme suivante:

état actuel -> événement -> état suivant

La règle est appliqué au système au moment où l'événement que l'on veut déclencher coïncide avec l'événement produit dans le système. La méthode de détection consiste en un algorithme proposé au concepteurs de spécifications qui devraient examiner une liste possible de combinaisons d'événements pouvant éventuellement conduire aux interférences crées par les états transitoires du système. Pour ce faire, les auteurs ont utilisé le langage formel de réseaux de Petri et ils ont converti les règles de production dans ce langage. Les places et les transitions des réseaux de Petri correspondent aux états et aux transitions des FSMs. Les jetons des réseaux de Petri sont assimilés aux terminaux du service téléphonique. Pour identifier les terminaux, les jetons seront "colorés". Même si les règles de productions ont été reformulées pour ces réseaux Petri "colorés", on manque

pour l'instant de méthode mathématique pour le calcul de l'ensemble des suites d'événements générées, dit le "T-invariant". Pour faire ce dernier calcul, on efface l'information de couleur et en appliquant l'algorithme proposé on obtient les résultats suivants: le nombre d'états atteignables n'est pas réduit par cette technique mais le nombre d'applications de règles nécessaires pour les atteindre est diminué. On remarque aussi que les "T-invariants" pour les réseaux "non colorés" ne sont pas toujours les mêmes que ceux pour les réseaux "colorés" ce qui donne des fois des "T-invariants" redondants par rapport aux spécifications des services.

1.2.3 La spécification formelle des services téléphoniques avec LOTOS

Une autre approche basée sur l'algèbre de processus propose la représentation formelle des services téléphoniques dans le langage LOTOS. En effet, plusieurs auteurs comme Stepien et Logrippo (1995) essaient d'exprimer les phénomènes rencontrés dans la modélisation des services téléphoniques de cette manière. Les principes de modélisation exposés par Cheng (1995) sont les mêmes que ceux déjà présentés dans les sections précédentes, c'est à dire qu'il modélise chaque service par une machine, il compose par la suite ces machines et il calcule enfin toutes les configurations atteignables. Ces machines sont synthétisées en couches, où la couche de bas niveau représente le service téléphonique de base et la couche de haut niveau représente les services. Les processus générés par les services sont représentés par des graphes qui ont comme événements des "points de l'appel" (Points in Call, PICs) et "des points de détection"(Detection Points, DPs). Les événements sont aussi

utilisés pour la synchronisation avec d'autres processus qui représentent des nouveaux services. Cette représentation permet de formuler individuellement les spécifications des services et offre un mécanisme de compositions appelé "composition parallèle".

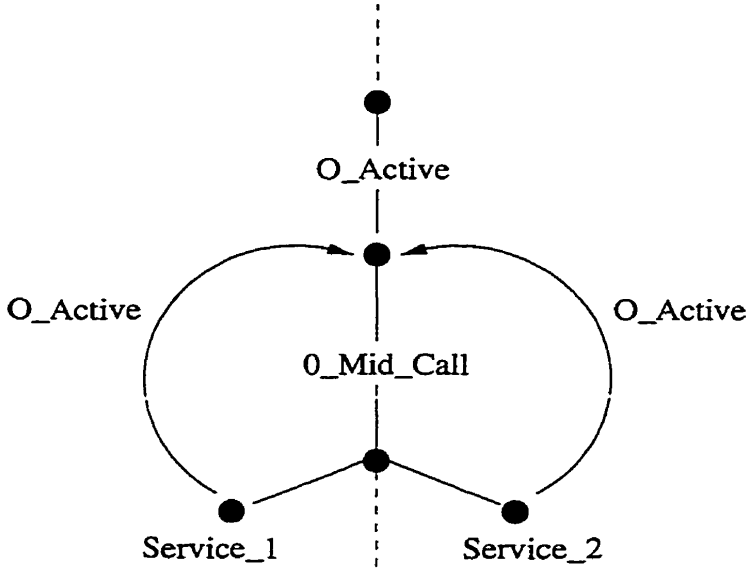


Figure 1.3 La composition parallèle des services 1 et 2

De plus, pour chaque service on introduit un processus "link" qui fait la connexion entre ce processus et le modèle d'appel standard. À ce moment là, il se peut que deux services soient invoqués en même temps dans un point du graphe, nommé "point d'invocation" ou "point de synchronisation". Les techniques de traitement de ces points sont de deux types: l'introduction d'un nouveau processus qui contrôle les priorités des services invoqués simultanément et l'établissement des priorités et/ou le blocage de certains services, ce qui est considéré comme une contrainte de composition. Toutes ces informations seront exprimées dans un processus appelé SIM (Service Interaction Management). Les

interférences peuvent être analysées en étudiant ces processus "link" entre services qui sont le moyen d'interaction pour les processus qui expriment les différentes fonctions téléphoniques.

Les auteurs Faci et Logrippo (1994) proposent aussi une représentation formelle des services basée sur le langage de spécification LOTOS. Ils définissent trois types de contraintes imposées dans la synthèse des spécifications: des contraintes locales qui établissent les séquences d'événements pour chaque abonné, qu'il soit appelé ou appelant, des contraintes "bout-en-bout" (end-to-end) qui établissent les séquences d'événements pour chaque connexion entre deux abonnés et finalement des contraintes globales telles que celle qui associe un seul numéro d'abonné à une connexion donnée.

Ils ont constaté que chaque service agit de la part de l'appelant ou de la part de l'appelé et qu'il y a peu de services avec un double rôle. Par exemple, le service CW agit de la part du destinataire et le service TWC de la part de l'appelant.

Les spécifications des services sont formalisées de façon indépendante et sont par la suite réunies, si nécessaire, dans une spécification complexe. Pour détecter des interactions au niveau des spécifications, ils ont adapté le modèle "orienté-connaisance" ("knowledge-oriented") de Halpern et Moses, comme suit: pour analyser par exemple si deux services interfèrent, le concepteur définit un ensemble de "buts" (knowledge goals) à atteindre pour le système en question; si l'un de ces "buts" n'est pas atteignable, on décide qu'il y a une interférence ou une erreur de conception. Les "buts" sont exprimés comme des processus LOTOS qui sont composés en parallèle avec la spécification complexe. Un exemple de but serait "l'état de conversation" (talking state). Si on n'est pas capable de détecter une

interférence en utilisant un but quelconque, il est recommandé de choisir d'autres buts et de simuler de nouveau. C'est donc la responsabilité du concepteur de définir des buts pertinents pour être capable de faire cette détection.

1.2.4 Autres approches

Quelques auteurs ont développé une modélisation du réseau téléphonique nommé "évitant-l'interférence" (interaction avoiding) qui fonctionne avec des concepts nouveaux pour mieux décrire les phénomènes qui apparaissent dans ces réseaux. Les auteurs soulignent qu'une source d'interférence est la terminologie de télécommunication qui décrit par le même mot des éléments différents du réseau. Par exemple "numéro d'abonné" peut signifier un abonné "i", son "rôle" dans une connexion téléphonique ou encore l'équipement qui est utilisé pour acheminer son appel. Le "rôle" dénote un abonné ou un groupe d'abonnés; il possède une identification qui permet la reconnaissance des participants dans un appel. Les services sont considérés comme des ensembles d'opérations de télécommunications qui sont disponibles pour un "rôle". Quand un abonné sollicite un service, il aura, du début, un rôle déterminé. Ce modèle est dit un modèle informationnel et il y a quelques modélisations pour des services tels que: l'appel vocal de base (basic voice call), appel vidéo de base (basic video call), etc. Les auteurs envisagent de consolider ce modèle pour pouvoir analyser des services plus complexes et investiguer des méthodes de conception de services.

Avec la même idée d'éviter la confusion de terminologie, d'autres auteurs (Zibman, 1995)

essaient de définir une nouvelle architecture basée sur la séparation des tâches (separation of concerns), par exemple: on sépare les utilisateurs des terminaux, les appels de connexions, etc.

Griffeth et Velthuijsen (1994) proposent l'introduction des "agents" et des "négociateurs" avec les rôles suivants: les agents sont des entités du système qui essaient de réaliser leurs objectifs respectifs, les négotiateurs sont ceux qui les aident à réussir. La structure d'un système de télécommunication fonctionnant sur ces principes sera celle donnée à la figure 1.4.

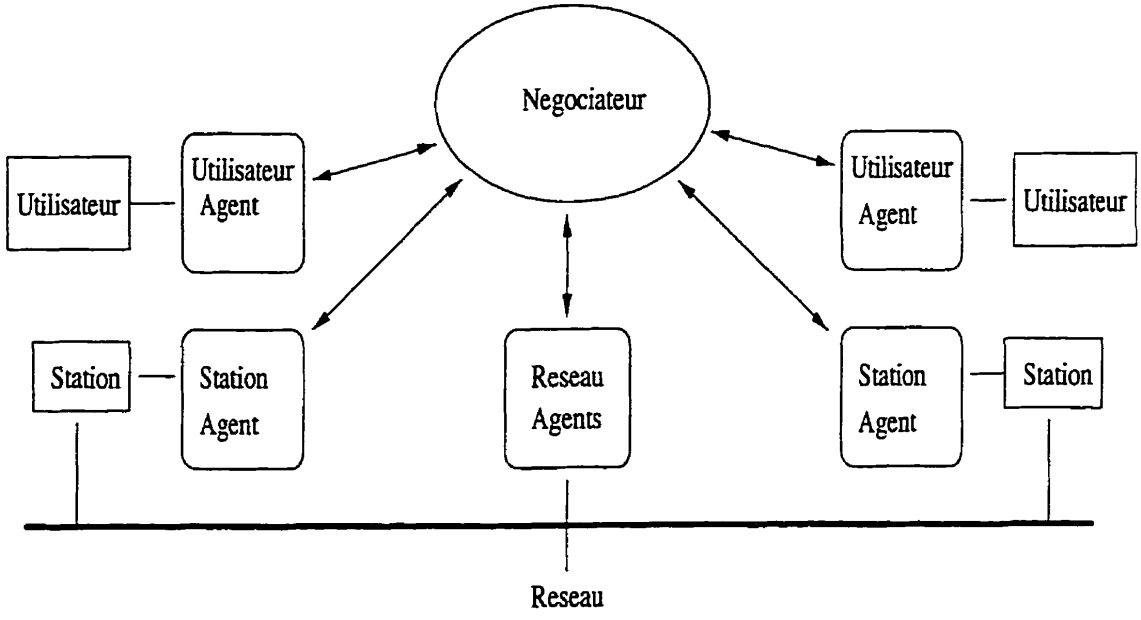


Figure 1.4 Le modèle du réseau à négociateur

La négociation est assurée par des «arbitres». Un processus de négociation se déroule

comme suit:

- un abonné ou un autre utilisateur du réseau téléphonique crée une proposition pour initier ou modifier un appel. Cette proposition consiste en un ensemble ordonné d'opérations qui seront exécutées par le commutateur téléphonique.
- le système de négociation répondra à cette proposition par le même ou par un autre ensemble d'opérations.

La conclusion des auteurs est que la méthode de négociation est plus pratique que les méthodes conventionnelles appliquées pour résoudre les interférences, parce que tous les problèmes qui arrivent sont résolus de façon dynamique pendant le fonctionnement du système (run-time resolution). Cette méthode aurait l'avantage de pouvoir enregistrer les intentions courantes des abonnés et de les négocier.

1.3 Conclusion

Toutes les approches présentées sont adéquates pour exprimer l'information contenue dans les spécifications des services téléphoniques. On pourrait les juger selon leur capacité d'exprimer cette information d'une façon simple, condensée et non ambiguë. De ce point de vue on remarque les progrès faits dans les dernières années pour définir les règles de fonctionnement des services. Ces règles essaient de spécifier les connexions qui existent à un moment donné entre les parties qui communiquent. Une classification des interférences s'est avérée utile pour trouver des solutions à ce problème. En effet, on s'est mis d'accord

à l'effet que les nouveaux services ne devraient pas introduire des définitions qui sont en contradiction avec le modèle de base du réseau (ou du commutateur) téléphonique. Quant à la méthode de formalisation, la plupart des approches sont orientées vers les SEDs. Les interférences sont résolues en attribuant des priorités aux services selon des règles qui essaient de maintenir la fonctionnalité de deux services. Ces méthodes sont plus ou moins faciles à utiliser en fonction des méthodes de calcul utilisées. Par exemple, dans la modélisation Delphi (section 1.2.1), les auteurs déclarent qu'on peut faire la synthèse des services et l'analyse des interférences pour un nombre de services qui peut être assez grand (sans donner un ordre de grandeur). Dans l'approche de Kawarasaki (section 1.2.2), on montre que la taille des réseaux de Petri obtenus est proportionnelle à la complexité des services analysés et on essaie de contrôler le nombre des états en évitant la redondance. La méthode basée sur le langage LOTOS (section 1.2.3) ne pose pas de problèmes de taille parce qu'on opère avec des processus LOTOS.

Après avoir présenté les recherches les plus récentes dans ce domaine, une introduction théorique en SED introduit les éléments de base du processus de modélisation. Nous présentons ici le modèle d'un commutateur téléphonique dans un mini-réseau de trois abonnés. Cette présentation fait l'objet du chapitre 3. On y analyse en détail la signification des états et des transitions de l'automate et on souligne les éléments caractéristiques de ce service. La formalisation de la spécification est suivie par la synthèse d'un superviseur qui assure le contrôle dans le commutateur où on implante le service CSMI.
Nous présentons au chapitre 4 un exemple d'interférence entre ce service et CW, suivie

par une suggestion de résolution basée sur l'approche des priorités flexibles, proposée (Thistle, Malhamé, Hoang, Lafortune, 1996). Cette approche propose comme solution aux interférences d'établir les priorités entre les services conflictuels en fonction de l'événement qui fait évoluer le système de l'état critique où le conflit arrive, vers l'état suivant.

Au chapitre 5, les conclusions font une synthèse des résultats, soulignent l'importance de ces modèles pour l'implantation des services et proposent des suggestions sur les méthodologies de modélisation des services téléphoniques et de leurs interactions

CHAPITRE II

LA THÉORIE DE RAMADGE-WONHAM DANS LA MODÉLISATION D'UN RÉSEAU TÉLÉPHONIQUE L'APPROCHE MODULAIRE

2.1 Notions de base

Une approche initiée par Ramadge-Wonham (Ramadge, 1989) propose des techniques systématiques pour la synthèse de commande pour des systèmes à événements discrets (SED). Les SED sont des systèmes qui évoluent avec l'occurrence d'événements instantanés, qui surviennent à des moments aléatoires. Dans le cadre de la théorie Ramadage-Wonham, ils sont modélisés comme des "générateurs" de langages formels, munis de mécanismes de commande. Des "contrôleurs", ou "superviseurs", exercent de manière dynamique un contrôle sur l'évolution du système. La théorie vise principalement le développement de systèmes de commande complexes. Pour cette raison, elle insiste sur des méthodes "modulaires" de synthèse de superviseurs.

2.1.1 Alphabets et langages

Un "alphabet" Σ est un ensemble fini de symboles qui représentent les événements respectifs qui sont susceptibles de se produire dans un système donné. La notation $\Sigma+$ représente toutes les suites finies de symboles (ou "mots") $\sigma_1\sigma_2...\sigma_k$ où $k \geq 1$ et $\sigma_i \in \Sigma$.

Définissons:

$$\Sigma^* = \Sigma+ \cup \{\varepsilon\} \qquad \{2.1\}$$

où $\{\varepsilon\}$ est le mot vide.

Chaque application peut avoir son propre alphabet. Les éléments de Σ^* sont appelés des "mots".

2.1.1.1 L'alphabet local du commutateur téléphonique

La première étape dans la formalisation d'un problème de synthèse à l'intérieur du cadre formel de cette théorie est la définition de l'alphabet qui semble caractériser le mieux possible le problème en question. L'alphabet doit être composé d'événements qui apparaissent dans le processus de contrôle du système à modéliser, événements qui ne peuvent pas être décomposés et qui sont assez significatifs dans les processus engendrés par le système. Il faut faire dès le début une distinction entre des "événements cause" et des "événements effet", par exemple les tonalités générées par un commutateur téléphonique sont d'habitude des effets d'un événement produit dans la système: la tonalité de manoeuvre est l'effet d'un décrochage, la sonnerie est l'effet d'une composition de numéro, etc. Cette analyse est importante pour déterminer le type de chaque événement du point de vue de l'action de contrôle qu'on souhaite à exercer dans le système. Le résultat de cette analyse pour un commutateur téléphonique, donne l'alphabet présenté au tableau 2.1. Cet alphabet est dit "local" parce qu'il ne contient que d'événements qui concernent seulement le commutateur "i". Nous ne considérons qu'un "mini-réseau" téléphonique

comprenant seulement trois commutateurs, "i", "j", "k", avec un seul abonné("i", "j" ou "k" respectivement) branché directement à chacun. Cette option a été choisie parce que la plupart des services téléphoniques engagent seulement deux ou trois participant à la fois: sinon, nous pouvons toujours attribuer un double rôle à un des trois membres du mini-réseau choisi comme modèle. En plus un réseau de petite taille permet une analyse plus précise parce que les modèles générés sont plus petits sans perdre la qualité ou la quantité de l'information contenue par rapport au plus grands systèmes. En permutant les indices, nous obtenons les deux autres alphabets locaux (tableaux 2.2 et 2.3, pages 36 à 38). Un alphabet local est appelé Σ_i, Σ_j ou Σ_k et contient 29 événements pour chaque commutateur.

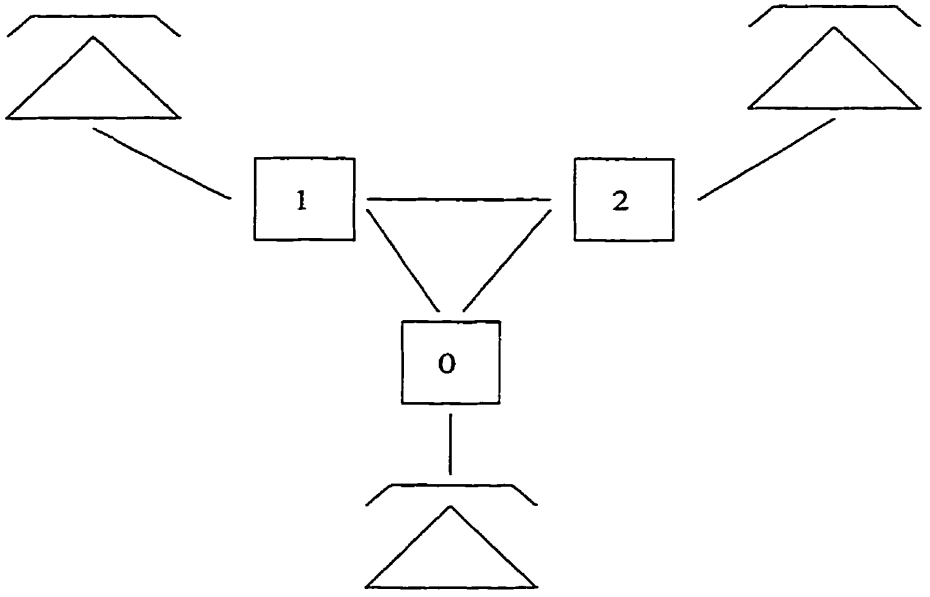


Figure 2.1 Un mini-réseau téléphonique à 3 abonnés

Le mécanisme standard de commande dans l'approche R-W est basé sur une répartition de l'alphabet (local en l'occurrence) en sous-alphabets d'événements "commandables" (Σ_c) et "non commandables" (Σ_n):

$$\Sigma = \Sigma_c \cup \Sigma_n \qquad \{2.2\}$$

Un événement est considéré commandable si il peut être empêché par un contrôleur externe et non commandable au cas contraire.

À la section suivante nous énumérons la liste des événements pour notre modèle du mini-réseau et nous discutons de leur commandabilité. Dans cette application, la commandabilité d'un événement dépend du commutateur où le superviseur en question sera implanté. Dans chaque modèle, la commandabilité est définie par rapport au commutateur "i".

2.1.1.2 Les événements de l'alphabet Σ_i du commutateur "i"

Les événements "on_hx" (on hook x, x = i, j, k) représentent l'accrochage de l'appareil téléphonique branché au commutateur "i", "j" ou "k", respectivement. Tous les trois sont non commandables. La non commandabilité exprime le fait qu'ils peuvent survenir en tout temps et donc ne sont pas sous le contrôle du commutateur. Nous considérons donc les "on_hj" et "on_hk", respectivement, comme étant observables par le commutateur "i".

L'événement "flash_i" représente un accrochage de moins de 1.5 secondes (conformément aux spécifications Bellcore). Si l'accrochage dure plus long temps que cela, il sera

considéré comme un "on_hi". Beaucoup de services utilisent une signalisation de ce type.

Les événements "req_ix" (request ix, x = i, j, k) représentent la composition du numéro de l'abonné "x", par "i". Ces événements sont générés par le commutateur "i" et ils sont en conséquence commandables par le commutateur "i". Ils peuvent être considérés comme des demandes de connexion envoyées vers le commutateur "x".

Les événements "req_xi" (x = j, k) représentent des demandes de connexion du commutateur "x" vers le commutateur "i". Ils sont considérés non commandables dans l'alphabet du commutateur "i" parce qu'ils sont générés au niveau du commutateur "x". Les événements "con_ix" (connect ix, x=j, k) sont générés par le commutateur "i" après un "req_ix". Ils indiquent que l'étape de la réalisation de la connexion est achevée. Si la ligne de l'appelé est libre, un tel événement donnerait lieu typiquement à une sonnerie sur son appareil. Ces événements sont considérés non commandables par le commutateur "i", parce que ce sera le commutateur "x" qui décidera si la connexion sera faite ou non.

Les événements "no_con ix" (x = j, k) peuvent être générés à la suite d'un "req_ix" si le commutateur "x" empêche l'établissement de la connexion de "i" à "x". L'abonné "i" entendra typiquement dans un tel cas une tonalité d'occupation. Ces événements sont considérés non commandables par le commutateur "i", car ils sont générés au niveau du commutateur "x". Les événements "no_con xi" (x = i, j, k) représentent la réponse du commutateur "i" à un "req_xi". Ce sont des événements commandables et le résultat est la

tonalité d'occupation envoyée par le commutateur "i" vers "x".

Les événements "fwd_xyz" (forward x, y, z = i, j, k):

Cette catégorie d'événements nécessite une discussion spéciale. Ils signifient le renvoi automatique de l'appel destiné à "y", vers "z". Ce type d'événement peut suivre à un "req_xy" (voir les figures 2.2 et 2.3) et il donne au commutateur qui a initié l'appel une possibilité de décision concernant le renvoi de cet appel au cas où il ne peut pas être achevé vers la destination initiale. Dans cette modélisation, le "fwd_xyz" signifie seulement l'initiation du renvoi et il est équivalent à une demande faite par "y" vers "x" de renvoyer son appel vers "z". L'accomplissement de l'appel initial sera donc sous la responsabilité de "x" qui va générer un "req_xz" ou bien un "no_con xx". Cette deuxième étape est sous le contrôle du commutateur "x". On constate que les événements amènent le système à l'état 4. Nous souhaitons quand même que le "fwd_xyz" soit suivi par un "req_xz" ou par un "no_con xx". Ces contraintes sont retrouvées dans l'automate à la figure 2.3.

Les événements "fwd_iij", "fwd_iik", "fwd_jij", "fwd_kik", "fwd_jik" et "fwd_kij" sont considérés comme étant commandables au niveau du commutateur "i" parce que c'est lui qui les génère. Les événements "fwd_iki", "fwd_ikj", "fwd_iji" et "fwd_ijk" sont considérés non commandables.

2.2 Automates et générateurs

Un automate (déterministe) "A" est un quintuple:

$$A = (\Sigma, Q, \delta, q0, Q_t) \qquad \{2.3\}$$

où Σ est un "alphabet" fini.

Q est un ensemble d'"' états"

δ est une "fonction de transition" définie comme suit:

$$\delta : \Sigma \times Q \dashrightarrow Q \qquad \{2.4\}$$

q0 est l'"état initial"

Q_t est l'ensemble des "états terminaux", $Q_t \subseteq Q$.

La fonction δ peut être étendue, par récurrence, à la fonction:

$$\delta : \Sigma^* \times Q \dashrightarrow Q \qquad \{2.5\}$$

Elle peut être une fonction partielle.

Le langage $L \subseteq \Sigma^*$ reconnu par A est:

$$L = \{s \in \Sigma^* \mid \delta(s,q0) \cap Q_t \neq \varnothing\} \qquad \{2.6\}$$

On peut obtenir ainsi une représentation concrète d'un langage par une structure de transitions.

Un "générateur "est lui aussi un quintuple:

$$G = (\Sigma, Q, \delta, q0, Qm) \qquad \{2.7\}$$

où Qm est un ensemble d'états buts tels que définis à la section 2.2.2. La fonction de transition est définie seulement pour un sous-ensemble d'éléments σ ($\sigma \in \Sigma$). La notation

$\delta(s,q)!$ veut dire que $\delta(s,q)$ est définie. Comme pour la définition précédente, on étend δ à une fonction partielle:

$$\delta : \Sigma^* \times Q \dashrightarrow Q \qquad \{2.8\}$$

d'après les règles suivantes (Ramadge, 1989):

$$\delta(\varepsilon,q) = q \qquad \{2.9\}$$

$$\delta(s\sigma,q) = \delta(\sigma, \delta(s,q)) \qquad \{2.10\}$$

si la partie droite est définie.

Le générateur est donc une machine qui démarre à l'état 0 et génère des mots en exécutant seulement les transitions qui sont définies par sa fonction partielle de transition δ. Le sous-ensemble de mots $s \in \Sigma^*$ tel que $\delta(s,q0)!$ représente le comportement en boucle fermée du générateur G ou le langage "généré" par G, dénoté par L(G).

2.2.1 Le modèle local simple du commutateur téléphonique

Étant donné l'alphabet local de l'application (section 2.1.1.), nous procédons à la synthèse du modèle local du commutateur téléphonique (Thistle, J.G., 1996).

Le modèle sera synthétisé pour un mini-réseau à trois commutateurs qui peut simuler la plupart des services et des interférences. Les calculs sur les langages qui décrivent les automates sont faits à l'aide du logiciel TCT (annexe I). Le modèle local simple du commutateur "i" est représenté à la figure 2.2. Ce modèle est considéré simple parce qu'on ne tient compte ici que des événement de renvoi de l'appel "fwd_xyz". Il s'agit d'un automate à quatre états, symétrique par rapport aux trois membres du réseau. Ce modèle

représente le commutateur en l'absence du contrôle et comprend les séquences d'événements qui peuvent être générés par un seul abonné. L'état initial de cet automate est l'état 0 dans lequel l'abonné est accroché. S'il décroche, l'automate évolue vers l'état 4 où il peut engendrer des appels et générer des mots tels que:

req_ix con_ix

req_ix no_con ix

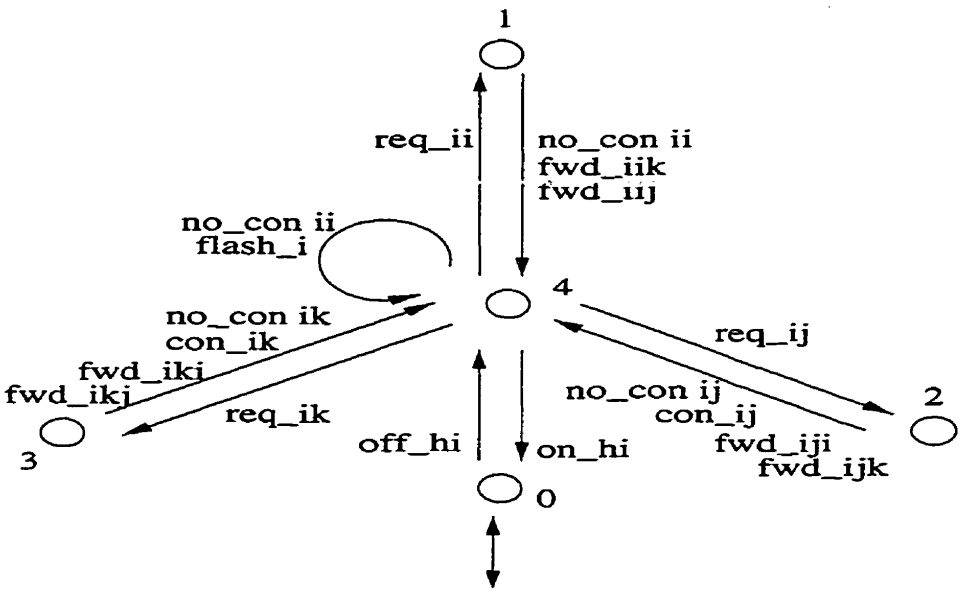


Figure 2.2 Le modèle local simple du commutateur téléphonique

2.2.1.1 Commentaires sur le modèle

Le modèle montré ci-dessus présente quelques particularités:

1. La structure de transitions impose, évidemment, des contraintes sur les séquences d'événements qui appartiennent à L(G), par exemple, chaque "req_ix" (x= i, j, k) doit être

suivi par l'un des événements suivants: "con_ix", "no_con ix". On ne peut pas engendrer par exemple une séquence comme:

req_ix -------> on_hi

2. Les événements qui ne paraissent pas dans cette représentation mais qui font partie de l'alphabet local, sont "bouclés" dans tous les états de l'automate. Le terme "bouclé" signifie ici un événement qui se produit sans changer l'état du système, donc on le représente comme une boucle attaché à l'état respectif. On ajoute ces boucles avec la procédure "selfloop" de TCT.

3. Cet automate ne modélise pas le renvoi de l'appel vers un troisième commutateur. Cette approche est traitée séparément à la section 2.2.4.

Comme le mini-réseau contient trois commutateurs identiques, on calcule de la même façon les deux autres générateurs pour "j" et "k", en permutant les indices. Nous obtenons ainsi les automates SWJ et SWK.

Chacun des trois automates a 8 états et 24 transitions.

2.2.2 Langages achevés

Pour pouvoir introduire une notion de blocage, nous introduisons d'abord la notion de "langage achevé".

Le sous-ensemble de mots $s \in L(G)$ tel que $\delta(s,q0) \in Qm$ représente le "langage achevé" Lm(G) du générateur G et ses éléments sont les mots marqués. L'interprétation de cette

définition est la suivante: tous les mots générés à partir de l'état initial et qui appartiennent à L(G) représentent une action achevée ou un cycle complet de l'automate. On a les relations suivantes entre les langages L(G) et Lm(G):

$$\varnothing \subseteq Lm(G) \subseteq L(G) \subseteq \Sigma^* \qquad \{2.11\}$$

Il se peut quand même que le système génère des mots qui ne peuvent pas être étendus à des mots marqués.

G est considéré "non blocant" si:

$$L = \overline{Lm} \qquad \{2.12\}$$

ce qui veut dire qu'on peut étendre chaque mot généré par G à une tâche achevée.

La notation $\overline{L}$ dénote partout la "fermeture" de L définie comme suit:

$$\overline{L} = \{s \in \Sigma^* \mid n \in \Sigma^* \text{ tel que } sn \in L\}$$

Dans l'exemple du générateur du commutateur téléphonique le seul état achevé est l'état 4 et on remarque qu'on peut l'atteindre si seulement si on a généré les séquences:

req_xy -------> con_xy

req_xy --------> no_con xy

qui représentent des appels complets.

2.2.3 Produit synchrone de générateurs

Les systèmes complexes sont d'habitude modélisés en plusieurs étapes. Pour simplifier la

synthèse, on décompose généralement une tâche complexe en plusieures tâches plus simples pour lesquelles on synthétise des automates. Les automates partiels ainsi obtenus doivent fonctionner concurremment pour exécuter la tâche complexe et se synchroniser pour assurer le fonctionnement souhaité. L'opération TCT qui permet le calcul de l'automate complexe qui représente la combinaison des automates simples est le produit synchrone, "sync".

En termes de langages, on définit une fonction Pi comme suit:

- soit $L1 \subseteq \Sigma_1^*$, $L2 \subseteq \Sigma_2^*$, $\Sigma = \Sigma_1 \cup \Sigma_2$ et

$$P_i : \Sigma^* \rightarrow \Sigma_i^* \ (i = 1, 2) \text{ tel que} \qquad \{2.13\}$$

$$P_i(\varepsilon) = \varepsilon \qquad \{2.14\}$$

$$P_i(\sigma) = \varepsilon \quad \text{si } \sigma \notin \Sigma_i \qquad \{2.15\}$$

$$P_i(\sigma) = \sigma \quad \text{si } \sigma \in \Sigma_i \qquad \{2.16\}$$

$$P_i(s\sigma) = P_i(s)P_i(\sigma),\ s \in \Sigma^*,\ \sigma \in \Sigma_i \qquad \{2.17\}$$

L'effet de P_i sur un mot "s" est d'éliminer tous les événements σ du mot "s" tels que $\sigma \notin \Sigma_i$. P_i est dite la "projection naturelle" du Σ^* dans Σ_i. Le produit synchrone est:

$$L1 \| L2 = P_1^{-1}(L1) \cap P_2^{-1}(L2) \qquad \{2.18\}$$

Un mot $s \in L1 \| L2$ si seulement si:

$$P_1(s) \in L1 \text{ et } P_2(s) \in L2 \qquad \{2.19\}$$

Si $L1 = Lm(G1)$ et $L2 = Lm(G2)$, on interprète le produit synchrone comme représentant l'évolution simultanée de deux sous-systèmes modélisés pour G1 et G2 qui se

synchronisent sur l'occurrence d'événements communs. La procédure "sync" de TCT calcule un générateur:

$$G = \text{sync}(G1, G2) \qquad \{2.20\}$$

où

$$Lm(G) = Lm(G1) \parallel Lm(G2) \qquad \{2.21\}$$

$$L(G) = L(G1) \parallel L(G2) \qquad \{2.22\}$$

Si $\Sigma_1 \cap \Sigma_2 = \varnothing$, on appelle ce produit "entrelâcé" ("shuffle"). Dans ce cas, le langage achevé du générateur:

$$G = \text{sync}\ (G1, G2)$$

est obtenu tout simplement entrelaçant les mots de L1 avec les mots de L2. Ceci modélise l'opération indépendante et asynchrone des sous-systèmes modélisés par G1 et G2.

Pour modéliser le mini-réseau téléphonique nous faisons le produit entrelacé simultané des trois générateurs SWI, SWJ et SWK:

$$G = \text{sync}\ (SWI, SWJ, SWK) \qquad \{2.23\}$$

2.2.4 La modélisation du renvoi "fwd_xyz"

Pour que le modèle présenté à la section 2.2.1 soit plus réaliste, nous l'enrichissons avec des événements "fwd_xyz" (Thistle, 1996) qui représentent une sorte de communication entre deux commutateurs impliqués dans une connexion, de la façon suivante: si un commutateur génère un "req_xy", il donne la possibilité à "x" de transférer son appel vers

un autre commutateur. L'automate qui décrit cette fonction est montré à la figure 2.3.

Le fonctionnement du commutateur "i" sera donc décrit par l'accomplissement simultané des deux fonctions: celle de base décrite à la figure 2.2 et celle mentionnée ci-dessus.

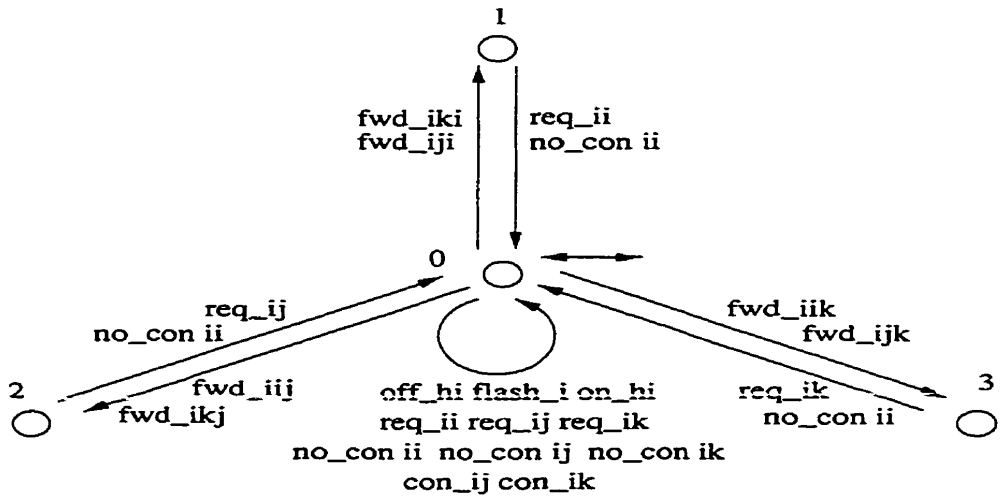


Figure 2.3 La modélisation du renvoi "fwd_xyz"

Pour que le générateur englobe, dans sa forme finale, la modélisation à la figure 2.2 et aussi celle à la figure 2.3, nous calculons la "conjonction" de ces deux automates (comme définie à la section 2.3.3). En terme des langages, si on note par G1 et G2 les deux automates et par G leur conjonction, nous obtenons:

$$Lm(G) = Lm(G1) \cap Lm(G2) \qquad \{2.24\}$$

$$L(G) = L(G1) \cap L(G2) \qquad \{2.25\}$$

La fonction TCT "meet" nous permet de calculer cette conjonction. Pour notre exemple nous obtenons:

SWI_M = meet (SWI, FWD_I) {2.26}

SWJ_M = meet (SWJ, FWD_J) {2.27}

SWK_M = meet (SWK, FWD_K) {2.28}

Chacun des trois automates a 8 états (état marqué: 0) et 24 transitions.

Le générateur sera:

G = sync (SWI_M, SWJ_M, SWK_M) {2.29}

Ce générateur aura 512 états dont l'état initial marqué et 4608 transitions. Le modèle local du générateur sera obtenu en calculant la projection du langage de cet automate G, dans l'alphabet Σ_i:

G_I = project (G, LIST) {2.30}

où LIST = $(\Sigma_j \setminus (\Sigma_i \cap \Sigma_j)) \cup (\Sigma_k \setminus (\Sigma_i \cap \Sigma_k))$

G_I aura 72 états et 600 transitions. Dans ce modèle local, on retrouvera comme marqués, les projections des mots marqués du langage L(G).

2.3 Superviseurs et superviseurs non blocants

Nous enrichissons les SEDs d'un mécanisme de commande dans le but de contrôler leur évolution. En particulier, nous considérons que les événements qui composent l'alphabet sont soit commandables, soit non commandables. De cette façon, nous obtenons une partition de l'alphabet Σ, comme suit:

$\Sigma_c \subseteq \Sigma$ qui représente le sous-ensemble d'événements commandables de Σ et

$\Sigma_n \subseteq \Sigma$ qui représente le sous-ensemble d'événements non commandables de Σ

Nous considérons que les événements commandables peuvent être empêchés par un contrôleur ou "superviseur" et que les autres ne sont pas sous son contrôle.

Le rôle d'un superviseur sera de restreindre le langage du générateur de façon à le garder à l'intérieur de limites acceptables.

On souhaite respecter ces spécifications en synthétisant le superviseur le moins restrictif possible pour le fonctionnement du système. Dans les termes de la théorie des SED, les superviseurs sont des applications:

$$f : L(G) \rightarrow \Gamma$$

où $\Gamma = \{\gamma \subseteq \Sigma \; ; \; \Gamma \supseteq \Sigma_n\}$ représente "l'ensemble d'entrées de commande" et f(s) représente l'ensemble des événements qui sont activés à la suite de l'occurrence du mot "s". Le langage $L_f(G)$ engendré par G sous la supervision de "f" est défini par récurrence sur la longueur des mots, comme suit:

$$\varepsilon \in L_f(G) \qquad \{2.31\}$$

et $$s\sigma \in L_f(G) \Leftrightarrow s \in L_f(G),\ \sigma \in f(s) \text{ et } s\sigma \in L(G) \qquad \{2.32\}$$

pour tout $s \in \Sigma^*$, $\sigma \in \Sigma$. L'interprétation de {2.32} est la suivante: tout mot "sσ" obtenu par la concaténation du mot "s" avec l'événement, est généré sous la supervision de "f" si seulement si "s" est généré sous la supervision de "f", "σ" est permis par "f" à la suite de la génération de "s" et le mot "s" est généré par le SED sans supervision. Le rôle de la supervision est de contrôler l'ensemble de mots générés, en procédant à la désactivation de certains événements commandables selon l'état où il se trouve.

Le langage achevé par le système sous supervision est, par définition:

$$Lm_f(G) := L_f(G) \cap Lm(G) \qquad \{2.33\}$$

et il représente l'ensemble des mots achevés, générés par le SED et qui survivent sous la supervision de "f". Le superviseur est dit "non blocant" pour G si:

$$L_f(G) = \overline{Lm_f(G)} \qquad \{2.34\}$$

La condition de non-blocage signifie que tout mot généré par le SED sous la supervision fait partie de la fermeture du langage achevé généré sous la supervision, donc un superviseur non blocant n'empêche pas le générateur d'atteindre ses états marqués.

2.3.1 Commandabilité des langages

Un langage quelconque $K \subseteq \Sigma^*$ est dit commandable par rapport à un générateur G si:

$$\overline{K}\Sigma_n \cap L(G) \subseteq \overline{K}$$

Cette propriété nous dit que le langage K est commandable si seulement si tout mot "s" qui appartient à $\overline{K}$ complété avec un événement non commandable "σ_n"devient un mot "$s\sigma_n$" qui appartient à L(G).

La théorie SED donne les conditions sous lesquelles on peut faire la synthèse d'un superviseur non blocant pour un générateur quelconque G (Ramadge, 1989).

Soit:

$K \subseteq Lm(G)$ non vide

alors il existe un superviseur "f" non blocant pour G tel que $Lm_f(G) = K$ si et seulement si: (i) K est commandable par rapport à G:

$$\overline{K}\Sigma_n \cap L(G) \subseteq \overline{K} \qquad \{2.35\}$$

(ii) K est fermé par rapport à Lm(G):

$$K = \overline{K} \cap Lm(G) \qquad \{2.36\}$$

Comme un des buts de la synthèse des superviseurs est de trouver une solution optimale, nous faisons appel à la définition de la classe de sous-langages commandables d'un langage E quelconque, *C(E)*:

$$C(E) = \{K \subseteq E: K \text{ est commandable par rapport à } G\} \qquad \{2.37\}$$

qui contient une borne supérieure pour le sup-demi-treillis (*C(E)*) sous la relation d'inclusion "$\subseteq$" , notée "supC(E)", qui sera la solution du problème de supervision. Le langage E sera le langage de spécification (discuté au chapitre 2.4).

Les superviseurs synthétisés doivent être non blocants, c'est à dire:

$$L_f(G) \subseteq \overline{Lm_f(G)} \qquad \{2.38\}$$

2.3.2 Superviseurs non conflictuels

Pour qu'on puisse cumuler le contrôle de deux ou plusieurs superviseurs dans un même commutateur téléphonique, il est utile de savoir sous quelles conditions les superviseurs peuvent fonctionner ensemble. En termes de langages, on introduit la notion de "non conflictualité" (Ramadge, P.J., 1989) et on dit que deux langages K et L sont non conflictuels si:

$$\overline{(K \cap L)} = \overline{K} \cap \overline{L} \qquad \{2.39\}$$

c'est-à-dire chaque mot qui est un préfixe de K et de L en même temps, pourra être prolongé à un mot qui appartient à K et L en même temps.

Soit A un automate fini, il est "co-accessible"si:

$s \in \Sigma^*$ tel que $\delta(s,q0)!$ il existe $s' \in \Sigma^* \mid \delta(ss',q0) \cap Q_t \neq \varnothing$

Nous définissons aussi le superviseur "propre" pour G comme un automate accessible et "co-accessible" As si:

(i) L(As) est commandable

(ii) L(As) est Lm(G) fermé

(iii) le système en boucle fermée As/G est non blocant:

$$\overline{Lm_{As}}(G) = \overline{L(As) \cap Lm(G)} = \overline{L(As)} \cap L(G) = L_{As}(G)$$

2.3.3 Conjonction de superviseurs

Si nous obtenons pour plusieurs services des superviseurs propres, qui devraient opérer simultanément, nous pouvons réaliser une conjonction de superviseurs pour le commutateur en question qui serait une solution du problème original de contrôle. Étant donnés deux superviseurs propres (par rapport à G) S1 et S2, dont Lm(S1), Lm(S2) sont commandables par rapport à G et S1/G et S2/G sont non blocants, c'est à dire:

$$\overline{Lm(S1/G)} = \overline{L(S1/G)} \qquad \{2.40\}$$

$$\overline{Lm(S2/G)} = \overline{L(S2/G)} \qquad \{2.41\}$$

on définit la conjonction de S1 et S2 comme:

$$S1 \wedge S2 = Ac(S1xS2) \qquad \{2.42\}$$

où Ac(S1xS2) dénote le composant accessible du produit des deux automates (Ramadge, 1989). Soit un automate "A":

$$A = (\Sigma, Q, \delta, q0, Q_t)$$

Le composant accessible de l'automate A est:

$$Ac\,(A) = (\Sigma, Q_{ac}, \delta_{ac}, q0, Q_{tac}) \qquad \text{où}$$

$$Q_{ac} = \{\, q \in Q \mid \text{il existe } m \in \Sigma^* \text{ tel que } q \in \delta(m, x0) \,\}$$

$$\delta_{ac} = \delta \mid (\Sigma \times Q_{ac})$$

$$Q_{tac} = Q_t \cap Q_{ac}$$

2.4 Langages de spécification

Les langages de spécifications sont des représentation formelles des chaînes d'événements permises dans le système sous contrôle.

Une spécification prend typiquement la forme suivante:

$$Lm_S(G) \subseteq L(E) \qquad \{2.43\}$$

où S - le superviseur qui implante la spécification

G - le générateur

E - la spécification

2.4.1 Exemples simples de spécifications de services téléphoniques.

Prenons comme exemples deux services téléphoniques simples, le **OCS** (Originating Call Screening) et le **TCS** (Terminating Call Screening)

2.4.1.1 Le service OCS

Ce service consiste à bloquer les appels provenant de l'appareil d'un certain abonné vers une liste préétablie de numéros. Dans cet exemple, modélisé pour le mini-réseau à trois abonné "i", "j" et "k", on considère qu'on bloque l'accès de "i" vers "j" et on permet la connexion "i" - "k". Un automate qui exprime cette contrainte n'a qu'un seul état où on permet tous les événements de l'alphabet local Σi, sauf le "req_ij".

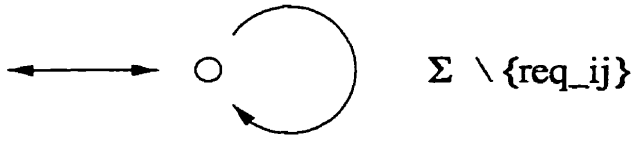


Figure 2.4 La spécification du service OCS

L'événement "fwd_iij" qui est commandable au commutateur "i" sera empêché par le superviseur en l'absence de "req_ij". Cette spécification (figure 2.4) avec un état marqué

et 28 transitions, conduira à la synthèse d'un superviseur le moins restrictif possible, SUP_OCS, par le calcul du plus grand sous-langage commandable du langage de la spécification.

2.4.1.2 Le service TCS

Ce service est conçu pour empêcher l'accès inverse, d'un (ou plusieurs) abonné(s) "j" vers un abonné "i", alors il doit interdire l'événement "con_ji" (commandable par le commutateur "i") de se produire. En même temps, il doit bloquer tout autre acheminement de cet appel ("req_ji"), en empêchant aussi le "fwd_jik". L'automate TCS (figure 2.5) a un état marqué et 27 transitions et sera utilisé pour le calcul du superviseur SUP_TCS qui sera le moins restrictif possible pour le commutateur "i".

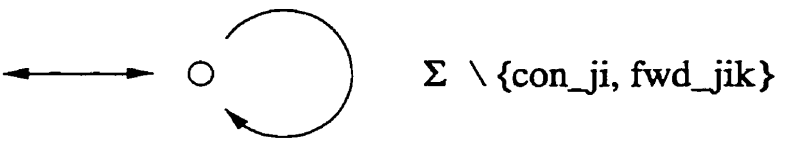


Figure 2.5 La spécification du service TCS

Tableau 2.1 L'alphabet local du commutateur "i"

No.	Événement	Code TCT	Commandabilité locale
1.	off_hi	2	N
2.	flash_i	8	N
3.	on_hi	14	N
4.	on_hj	16	N
5.	on_hk	18	N
6.	req_ij	1	O
7.	req_ik	3	O
8.	req_ii	5	O
9.	req_ji	20	N
10.	req_ki	22	N
11.	con_ij	32	N
12.	con_ik	34	N
13.	con_ji	7	O
14.	con_ki	9	O
15.	no_con ji	11	O
16.	no_con ki	15	O
17.	no_con ii	17	O
18.	no_con ij	40	N
19.	no_con ik	42	N
20.	fwd_iij	19	O
21.	fwd_iik	21	O
22.	fwd_iki	52	N
23.	fwd_ikj	54	N
24.	fwd_jij	23	O
25.	fwd_jik	25	O
26.	fwd_kik	27	O
27.	fwd_kij	29	O
28.	fwd_iji	72	N
29.	fwd_ijk	74	N

Tableau 2.2. L'alphabet local du commutateur "j"

No.	Événement	Code TCT	Commandabilité par rapp. à "i"
1.	off_hj	4	N
2.	flash_j	10	N
3.	on_hi	14	N
4.	on_hj	16	N
5.	on_hk	18	N
6.	req_ij	1	O
7.	req_jk	24	N
8.	req_jj	28	N
9.	req_ji	20	N
10.	req_kj	26	N
11.	con_ij	32	N
12.	con_jk	36	N
13.	con_ji	7	O
14.	con_kj	38	N
15.	no_con ji	11	O
16.	no_con kj	46	N
17.	no_con jj	48	N
18.	no_con ij	40	N
19.	no_con jk	44	N
20.	fwd_jji	56	N
21.	fwd_jjk	58	N
22.	fwd_jkj	60	N
23.	fwd_jki	62	N
24.	fwd_jij	23	O
25.	fwd_jik	25	O
26.	fwd_kjk	68	N
27.	fwd_kji	70	N
28.	fwd_iji	72	N
29.	fwd_ijk	74	N

Tableau 2.3 L'alphabet local du commutateur "k"

No.	Événement	Code TCT	Commandabilité par rapp. à "i"
1.	off_hk	6	N
2.	flash_k	10	N
3.	on_hi	14	N
4.	on_hj	16	N
5.	on_hk	18	N
6.	req_ik	3	O
7.	req_jk	24	N
8.	req_kk	30	N
9.	req_ki	22	N
10.	req_kj	26	N
11.	con_ik	34	N
12.	con_jk	36	N
13.	con_ki	9	O
14.	con_kj	38	N
15.	no_con ki	15	O
16.	no_con kj	46	N
17.	no_con kk	50	N
18.	no_con ik	42	N
19.	no_con jk	44	N
20.	fwd_iki	52	N
21.	fwd_ikj	54	N
22.	fwd_jkj	60	N
23.	fwd_jki	62	N
24.	fwd_kki	64	N
25.	fwd_kkj	66	N
26.	fwd_kjk	68	N
27.	fwd_kik	27	O
28.	fwd_kij	29	O
29.	fwd_kji	70	N

CHAPITRE III

LE SERVICE CSMI (Call Screening, Monitor and Intercept)

3.1 La spécification du service CSMI

Le service CSMI, récemment offert par NORTEL, étend la gamme de services connus sous le nom de VMNSS (Voice Message Network Support Services). Le service devrait améliorer la façon dont on traite les appels qui sont manipulés par un NBAS (Network-Based Answering Service).

En effet, la messagerie actuelle du réseau (NBAS) qui fournit le service d'enregistrement des messages se distingue d'un répondeur téléphonique par le fait qu'elle n'offre pas la possibilité d'identifier les appelants. Le nouveau service permet, au choix, l'identification de l'appelant ainsi que l'interception de l'appel. Il est offert aux clients qui ont une ligne simple (single-line) et qui sont déjà abonnés au NBAS.

Ce service prévoit l'envoi d'un signal d'avertissement (ring splash) de NBAS vers l'abonné, qui est avisé ainsi de la possibilité d'écoute et même d'interception de l'appel. Les processus générés par le service sont représentés à la figure 3.1. Dans la description et la modélisation du service qui suivent, nous utiliserons la terminologie suivante:

- "abonné" dénotera l'utilisateur du réseau téléphonique qui bénéficie du service CSMI.
- "appelant" dénotera celui qui appelle l'abonné et qui sera éventuellement son interlocuteur.

- le NBAS dénotera le troisième participant à ce service, la messagerie du réseau.

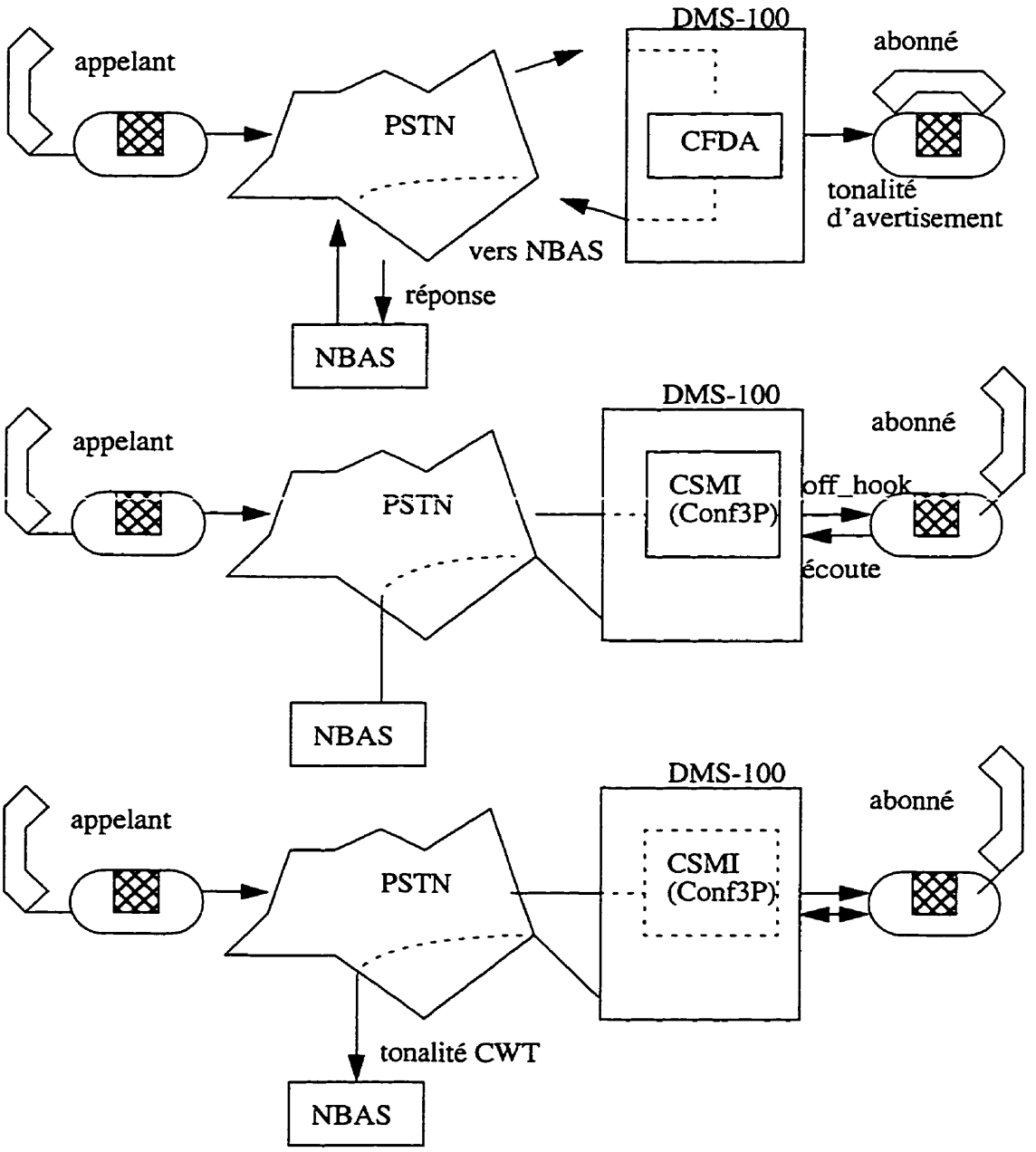


Figure 3.1 Les étapes du service CSMI

Dans le fonctionnement CSMI on distingue deux aspects:

A. **l'utilisation partielle du service, dans le seul but d'écouter le message au moment où il est déposé à la messagerie.** Dans ce cas, les étapes du service sont les suivantes:

- l'abonné décroche pendant "la période d'écoute" (screening period) et compose le code d'accès de CSMI; "période d'écoute" signifie la période pendant laquelle il est possible d'écouter le message qui vient d'arriver.
- pour un certain type d'abonné, il suffit de décrocher pendant la période d'écoute afin d'être capable d'écouter ce dernier message.

B. **l'utilisation complète du service,** c'est-à-dire l'écoute et l'interception du message se déroulent selon les étapes suivantes:

- un appellant essaie de rejoindre l'abonné.
- l'appel est renvoyé vers le NBAS.
- une fois que le NBAS répond, ce dernier envoie une "tonalité d'avertissement" (ring-splash) vers l'abonné. La tonalité d'avertissement est une courte sonnerie d'une durée de 0.5 secondes.
- l'abonné déclenche le service en décrochant avant que "la période d'écoute" ne soit écoulée (et il compose un code, si nécessaire). La période d'écoute est comptabilisée à partir du moment où le NBAS répond à l'appel dirigé vers lui par le commutateur de l'appelant. La durée de la période d'écoute est déterminée par le type d'utilisateur.
- à ce moment là, l'abonné entend son appelant, sans que ce dernier ne le

sache.

- jusqu'à ce que l'appelant raccroche, l'abonné peut intercepter l'appel en faisant un "flash-hook". Le "flash" est considéré comme étant une dépression du crochet d'une durée de 0.3sec. à 1.5sec. Sur certains appareils il existe aussi une touche qui produit un "flash" de 0.75sec.

Le fonctionnement du service ne doit pas dépendre du type de NBAS qui renvoie l'appel, mais doit plutôt offrir un mécanisme général d'écoute du dernier appel reçu.
L'application de ce service requiert des circuits de conférence.

3.2 Modélisation du service CSMI

L'approche proposée dans ce mémoire en vue de la modélisation de ce service téléphonique est l'approche modulaire de la théorie de la commande des systèmes à événements discrets dans ses aspects modulaires.
On se situe toujours dans un mini-réseau à trois commutateurs qui a comme modèle local l'automate représenté aux figures 2.2 et 2.3.

Pour la modélisation de CSMI nous utilisons la notation suivante:

- le commutateur "i": l'abonné au service CSMI.
- le commutateur "j": l'appelant de "i".
- le commutateur "k": le NBAS.

L'alphabet local utilisé a été présenté au chapitre 2.1.1.1.

La notation:

nombre -> événement -> nombre

dénote une transition dans l'automate en question. Le premier nombre représente l'état de départ, ce qui est suivi par le nom de l'événement qui provoque la transition et le dernier nombre représentant l'état d'arrivée.

3.2.1 Les deux aspects du service CSMI

Afin de formaliser la spécification du service, on suit les étapes d'établissement de la connexion décrites dans le document de spécification du service (NORTEL 1996).
La spécification sera représentée par un automate qui devrait respecter la fonctionnalité et les contraintes imposées par ce document. La modélisation est faite du point de vue du commutateur "i" où on implante le service.

Notons d'abord que tout appel destiné à "i" est automatiquement renvoyé vers "k", le NBAS. Il est donc nécessaire d'éliminer de la structure de la spécification les transitions:

"con_ji", "no_con ji", "fwd_jij"

qui seront toujours interdites par ce service.

3.2.1.1 Mode d'utilisation partielle du service comme messagerie

Considérons pour l'instant uniquement la fonction de messagerie du service. L'état initial de l'automate de la spécification est l'état 0 où on permet les événements de l'alphabet local mentionnés dans la boucle, sauf "off_hi" et "fwd_jik" qui quittent cet état. Cet état est considéré comme étant marqué pour les raisons exposées dans le chapitre précédent. L'état 0 est l'état de repos où le service n'a pas encore été activé.

L'événement "req_ji" se produit au moment où l'appelant "j" compose le numéro de "i". Il correspond à l'étape suivante:

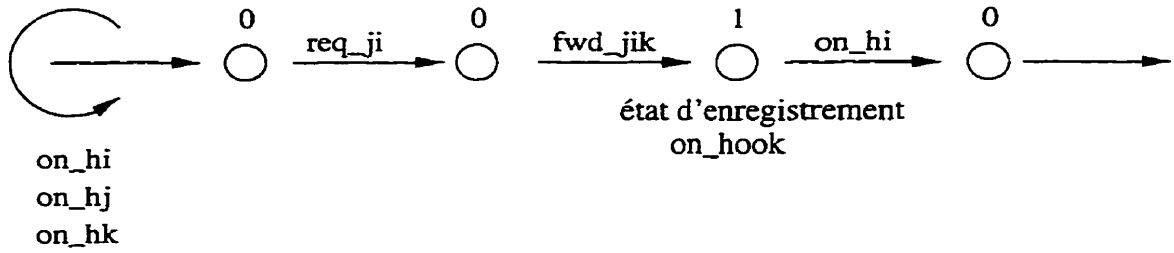
- un appelant essaie de rejoindre l'abonné (NORTEL 1996, étape 1) et peut déclencher la suite de transitions:

  req_ji fwd_jik

L'événement "req_ji" est bouclé à l'état 0. Comme tous les événements qui peuvent suivre le "req_ji" sont interdits par la spécification sauf le "fwd_jik", il est possible de quitter cet état seulement si le commutateur "i" génère le "fwd_jik".

Cette situation correspond à:

- l'appel est renvoyé vers le NBAS (NORTEL 1996, étape 2);



L'événement "fwd_jik" amène le système à l'état 1 qu'on appelle "état d'enregistrement on_hook" parce que, de fait, une fois l'appel renvoyé vers le NBAS il est automatiquement enregistré par ce dernier après quoi il y a retour à l'état initial 0 à condition que l'abonné reste toujours inactif et ne décroche pas:

1 ---> on_hi ---> 0

De l'état 1, les événements "on_hj", "k" mènent aussi le système vers l'état 0.

Au moment où commence l'enregistrement du message, l'abonné en est avisé par la tonalité d'avertissement. Nous supprimons de tels détails de signalisation afin d'alléger notre modèle qui, bien que se situant à un niveau d'abstraction assez élevé, permettra de rendre compte des événements reliés au contrôle du déroulement de l'appel. Ainsi donc l'étape - une fois que le NBAS répond, ce dernier envoie un "ring- splash" vers l'abonné (NORTEL, étape 3) - ne sera pas représentée dans la structure de transitions de l'automate.

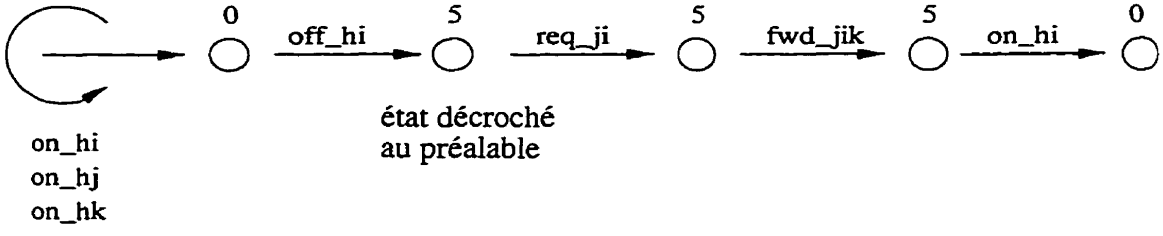
Notons que les boucles "req_ji", "fwd_xyz" présentes à l'état 1 correspondent aussi à des événements qui ne sont pas observables au niveau du commutateur "i". Il en est de même

de l'événement "con_jk" qui représente l'enregistrement.

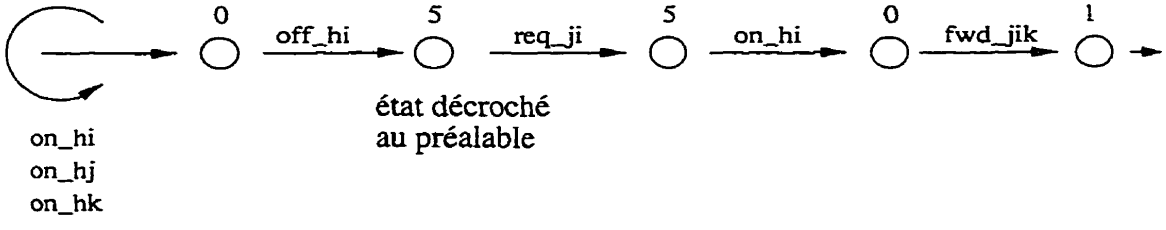
La deuxième transition à partir de l'état initial est provoquée par l'événement "off_hi" qui représente l'intention de la part de "i" d'initier un appel avant que le "req_ji" ne se produise. L'automate avance alors vers l'état 5. On appelle l'état 5 "état décroché au préalable".

Si dans cet état on reçoit un "req_ji", il y a la possibilité d'enregistrer le message, mais l'abonné n'est plus averti par la tonalité spécifique d'avertissement parce que l'abonné est décroché (NORTEL 1996, page 15, 2ème paragraphe).

Dans ce cas, on obtient la suite:

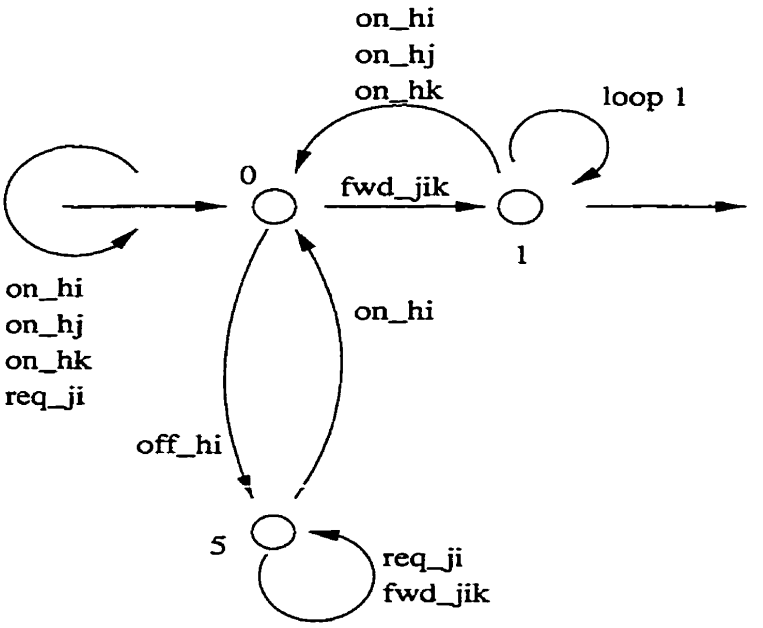


ou bien la suite:



si par exemple "i" termine son appel avant le "fwd_jik".

En combinant les deux modèles partiels de la spécification en un seul automate, on obtient le résultat suivant:



Notons qu'à chaque état de l'automate on ajoute des boucles avec les événements qui font partie de l'alphabet local mais qui ne figurent pas au diagramme. Ainsi, pour ne pas alourdir la représentation, dans tous les états il existe des boucles correspondant aux événements suivants:

{req_ki, con_ki, no_con_ki, fwd_kik, fwd_kij}

Ce modèle contient aussi quelques autres transitions, par exemple:

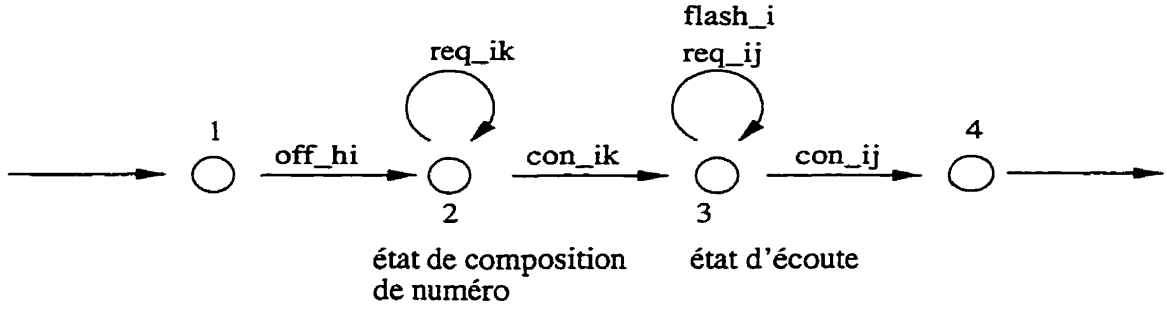
1 ---> off_hi ---> 5 (voir la figure précédente)

qui modélise un éventuel décrochement avant ou après le "req_ji".

3.2.1.2 Utilisation maximale du service CSMI

Dans ce qui suit nous caractérisons un scénario d'utilisation de toutes les possibilités de CSMI, c'est à dire l'écoute et l'interception de l'appel.

Comme le document NORTEL (1996) le spécifie, l'abonné ne peut exploiter toutes les possibilités du service CSMI que si son téléphone est dans l'état raccroché au moment où l'appel est renvoyé vers NBAS. Dans ce cas, la tonalité d'avertissement est envoyée vers l'abonné et la période d'écoute débute. On est alors à l'état 2 de l'automate à la figure 3.2 (où l'abonné est dans l'état accroché). À partir de cet état, il suffit de suivre les étapes 4, 5, 6 de la spécification pour obtenir la structure de transitions suivante:



Cette structure présente quelques particularités, notamment la modélisation du décrochage de "i" qui est représenté dans ce cas, par la séquence:

off_hi req_ik con_ik

même si on spécifie qu'on n'a pas toujours besoin de composer un code d'accès pour rejoindre le NBAS pendant la période d'écoute.

Il a été nécessaire de remplacer le "off_hi" par cette séquence afin de préserver la

compatibilité avec le modèle local du générateur qui prévoit que l'on ne puisse établir la connexion de conversation entre deux partenaires que si on génère la séquence:

req_xy con_xy

L'état 2 est un "état de composition de numéro". Dans chaque état, nous avons prévu les événements "on_hi", "on_hj", "on_hk" qui sont tous non commandables et peuvent modifier l'état du système à chaque moment.

Avec l'hypothèse que "j" reste toujours en ligne, "i" peut toujours l'intercepter s'il fait un "flash_i". L'état 3 est un "état d'écoute" parce que à ce moment débute, éventuellement, l'écoute du message.

L'interception est modélisée de la même façon dont on a déjà modélisé la connexion "i"-"k", en ce que le "flash_i" est remplacé par la séquence:

flash_i req_ij con_ij

Le résultat final est représenté à la figure 3.2. On remarque dans ce modèle la présence de quelques transitions qui relient les états 2 à 4 à l'état 5.

Les transitions:

2 -> on_hk -> 5 et

3 -> on_hk -> 5

décrivent la possibilité pour "i" d'évoluer vers l'état 5 au moment où le NBAS raccroche.

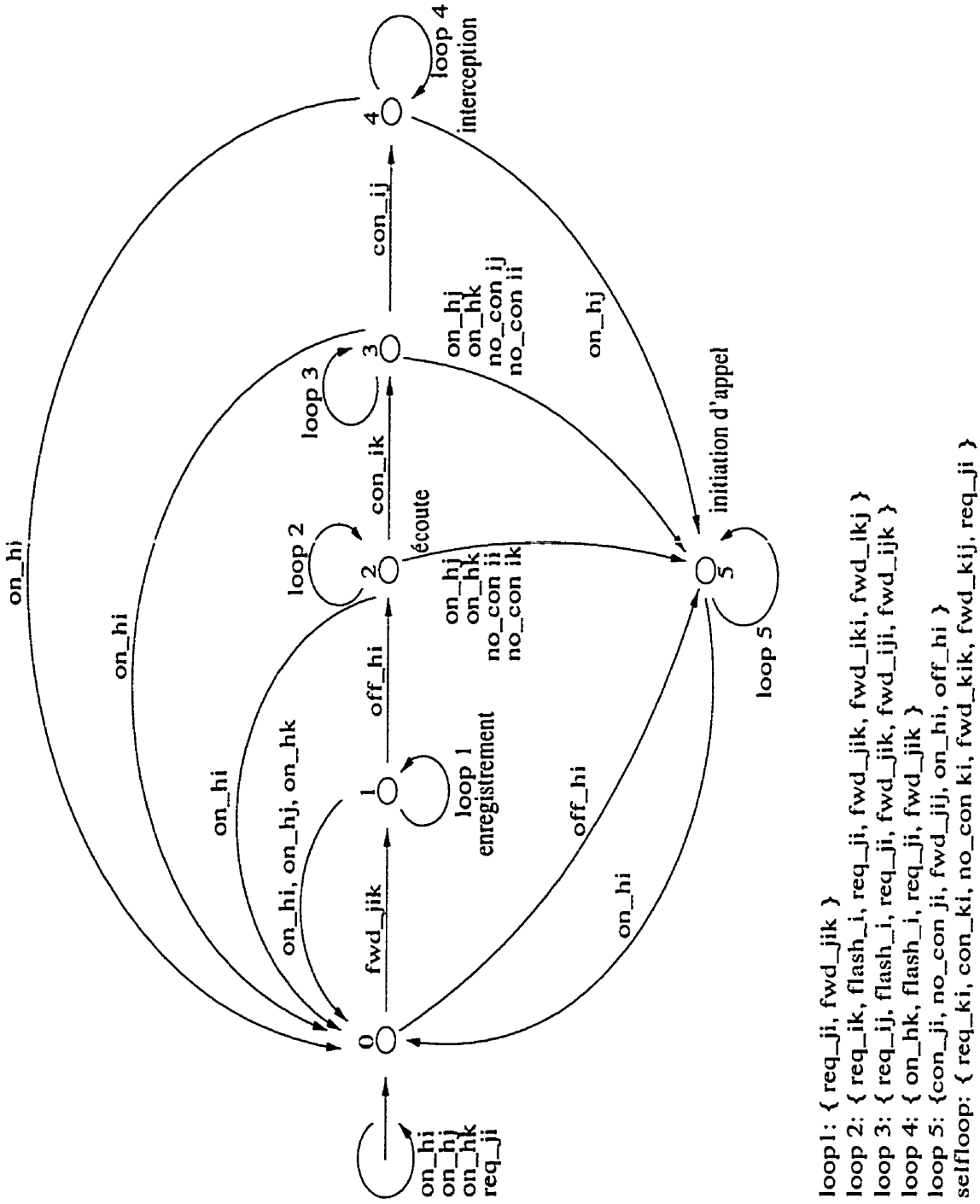


Figure 3.2 La spécification simple du service CSMI: première version

Les transitions:

2 -> no_con ik5 -> et

2 -> no_con ii5 ->

sont la suite normale d'un "req_ik" dans l'éventualité où un imprévu arrive dans le réseau empêchant alors "con_ik" de suivre.

Les transitions:

2 -> on_hj -> 5

3 -> on_hj -> 5

représentent l'évolution du système au cas où l'appelant raccroche, ce qui annule la possibilité d'interception.

Une fois que "i" a fait le "flash_i", l'état du NBAS (raccroché ou non) n'a plus d'importance quant à l'interception de "j". L'effet de "on_hk" est donc représenté par une boucle à l'état 4. Les transitions:

3 -> no_con ij -> 5 et

3 -> no_conii -> 5

sont la suite du "req_ij" et ont le rôle de permettre l'évolution du système vers un autre état au cas où, pour quelque raison, le "con_ij" ne pouvait être généré par le commutateur "i". Évidemment, la transition:

4 -> on_hj -> 5

représente la fin de l'étape d'interception et la transition vers l'état 5 où "i" peut

éventuellement initier un autre appel.

3.2.2. Les particularités du modèle

Le modèle présenté ci-dessus possède quelques particularités importantes. Tout d'abord, rappelons qu'il comprend deux parties:

1 le sous-modèle limité à la fonction de messagerie pour laquelle on a les états 0 et 5 conjointement avec toutes les transitions et les boucles correspondantes;

2 le sous-modèle correspondant au déploiement maximal du service de messagerie, écoute et interception représenté par les états de 0 à 4 et toutes les transitions associées.

En même temps, dans la figure 3.2, l'on retrouve certaines transitions qui lient ces deux parties de la structure de la spécification, c'est-à-dire toutes les transitions des états 2 à 4 vers l'état 5. La structure finale comprend 6 états et 98 transitions.

En deuxième lieu et en vue de modéliser l'écoute, nous avons interdit à partir de l'état 2 les événements "req_ij", "req_ii", qui pouvaient générer des connexions indésirables au moment où l'on désire que l'écoute commence, c'est-à-dire la suite: "req_ik con_ik". Nous avons alors permis à cet état la possibilité de générer toutes les tonalités d'occupation: "no_con ii", "no_con ik" afin d'éviter un blocage dans l'éventualité où la connexion "i"-"k" ne pouvait se réaliser. De plus, afin de modéliser l'interception à partir de l'état 3, nous

avons interdit les "req_xy" autres que le "req_ij" qui représente cette interception et avons permis les "no_con ii" et "no_con ij" pour les raisons citées plus haut. Enfin, tous les autres événements appartenant à l'alphabet local paraissent dans des boucles.

3.2.2.1 Analyse des transitions de retour

On appelle "transitions de retour" toutes les transitions qui amènent l'automate des états 1 à 4 vers l'état marqué, 0. L'événement responsable dans la majorité des cas est soit l'événement non commandable "on_hi" ou des événements "on_hj", "on_hk".

Les transitions:

1 -> on_hi -> 0 et

1 -> on_hj -> 0

représentent le retour à l'état initial après qu'un message ait été déposé alors que le téléphone de l'abonné était raccroché. L'interprétation de ces transitions est la suivante: si "j" raccroche après que le message a fini d'être transmis et "i" ne décroche pas, il y a de facto retour à l'état initial.

La transition "on_hi" est permise même si "i" est raccroché, pour des raisons de cohérence par rapport aux règles établies à la conception du générateur et qui permettent la succession:

fwd_jik on_hi.

En effet, nous avons adopté le principe que la spécification formelle doit être telle qu'elle restreint le moins possible tout comportement dynamique du générateur qui ne soit pas en

contradiction avec la fonction souhaitée. Si l'on passe à l'état 1, on observe le même événement "on_hi" qui peut intervenir après le "off_hi" si l'abonné décide d'abandonner l'écoute.

Notons que la transition est possible vers cet état en tout temps, sauf après l'événement "req_ik" (qui, conformément aux règles établies lors de l'élaboration du mini-réseau téléphonique, doit être suivi par un des événements "con_ik", "no_con ik", "no_con ii", etc.)

Nous retrouvons la même situation à l'état 3 où le "req_ij" pourrait être suivi par un "on_hi". À cet état, les événements "fwd_iji" et "fwd_ijk" sont bouclés, puisque de tels renvois sont "physiquement" possibles selon le modèle du générateur.

3.2.2.2 Analyse des transitions "on_hj"

Les événements "on_hi" et "on_hj" expriment l'intention imprévisible d'un des participants impliqués dans un appel de renoncer à continuer son appel et de raccrocher. C'est pour cette raison que l'on prévoit ces transitions non commandables dans tous les états de l'automate. Évidemment, il y a encore d'autres transitions non commandables, mais contrairement aux événements "on_hx", elles ne peuvent apparaître que suite à un événement commandable qui lui, pourra être empêché si l'on désire interdire les transitions en question. Nous concentrons donc, à présent, notre attention sur les transitions "on_hj" et "on_hk".

Dans la modélisation proposée ici, l'appelant ("j") est considéré à la fois comme étant l'appelant de "i" et comme un quatrième participant éventuel. Cette approche a été choisie afin de conserver la simplicité du modèle de mini-réseau. Lorsqu'on parle de "j", on se doit alors de spécifier s'il agit comme premier appelant ou comme un appelant subséquent.

Les transitions "on_hj" ont un rôle clé aux états suivants:

- à l'état 1: si l'on quitte cet état à cause d'un "on_hj" qui se produit avant que "i" ne décroche, on revient à l'état initial.
- aux états 2 et 3: si "j" raccroche à partir d'un de ces états, "i" ne pourra qu' écouter son message (sans interception) et donc le système évoluera vers l'état 5.
- à l'état 4, "i" a déjà parlé à "j" et si ce dernier raccroche, "i" peut engendrer ses propres appels, ou bien raccrocher lui aussi.

Ainsi donc, et tel que mentionné plus haut, "j" a deux rôles à jouer dans cette spécification: il est le premier appelant de "i" aux états 0 et 5 et il devient un appelant subséquent pour les états 1, 2, 3 et 4 où on retrouve les boucles "req_ji", "fwd_jik". La possibilité pour le service d'enregistrer des messages ultérieurs à l'événement "off_hi" dans l'état 1 est considérée conformément à la spécification NORTEL (1996).

3.2.2.3 Analyse des transitions "on_hk"

Pour les transitions "on_hk" on peut faire les remarques suivantes:

- à l'état 1, cette transition retourne l'automate à l'état 0.
- aux états 2 et 3, le raccrochage de "k" signifie le retour de "i" à l'état 5 où il peut initier de nouveaux appels ou bien raccrocher.
- à l'état 4 cette transition ne change plus l'état du système parce que la présence de "k" dans l'établissement de la connexion ne compte plus.

3.2.2.4 Analyse des transitions "flash_i"

La transition "flash_i" est interprétée par le service seulement à l'état 3. La situation est alors comme suit:

- "i" ayant répondu pendant la période d'écoute, décide d'intercepter son appelant.
- "j" ne raccroche pas et continue de faire son message.
- "k" est branché à "i" et enregistre en même temps le message de "j".

À cet état, nous pouvons constater que l'événement "flash_i" doit être considéré comme une "macro- transition" parce qu'il génère la séquence:

flash_i req_ij con_ij.

Encore une fois il aura fallu maintenir la cohérence avec le modèle du générateur du mini-réseau qui impose que le "con_xy" ne puisse être généré qu'à la suite d'un "req_xy".

3.2.3 Transitions clés dans la spécification CSMI

Désignons par le terme "transitions clés" les transitions qui marquent un progrès clairement identifiable dans l'utilisation du service CSMI. Une autre catégorie d'événements que l'on pourra qualifier de critiques apparaît dans certaines boucles. La détection de tels événements aura été jugée nécessaire en vue de l'implantation du superviseur de service CSMI.

Plus précisément:

- la transition 1 -> off_hi -> 2

  représente le début de l'intervalle durant lequel l'abonné écoute le message.

  À partir de cet instant, tous les trois commutateurs du mini-réseau sont engagés dans l'appel.

- la transition 1 -> on_hj -> 0

  est exécutée si l'événement "on_hj" précède l'événement "off_hi" mentionné ci-dessus; elle amène l'automate dans l'état 0 parce que cet accrochage met fin au traitement par CSMI de l'appel.

- la transition 3 -> flash_i -> 3

  représente le moment où l'interception commence.

Nous interprétons la succession d'événements suivante:

flash_i req_ij con_ij

comme représentant une seule opération de l'usager et considérons que l'événement "flash_i" est seul responsable du changement d'état du système. Nous avons considéré également le cas d'une chute de la couche physique du réseau ou d'une interférence avec un autre service, situation où le "con_ij" ne peut pas être engendré. Dans ce cas, les tonalités d'occupation sont générées dans la séquence:

4 -> no_con ij -> 5 et

4 -> no_con ii -> 5

pour signaler que l'interception n'aura pas lieu.

Au cas où "i" reçoit des appels successifs, il semble être dans l'impossibilité d'engendrer ses propres appels si les périodes d'écoute se "collent" l'une à l'autre. Le document de spécification NORTEL (1996) donne l'explication suivante:

- CSMI restreint le nombre d'appels qui peuvent être écoutés pendant une seule période d'écoute, pour donner à l'abonné la chance d'initier ses propres appels.
- le service empêche tous les autres appels renvoyés par "j" vers "k" d'être interceptés par "i" jusqu'au moment où l'appel en cours est soit terminé soit intercepté.

L'appelant doit alors exécuter les opérations suivantes:

- il entend la tonalité d'avertissement et il décroche:

  1 -> off_hi -> 2

- il raccroche pour un intervalle supérieur à 1.5 secondes, sinon son geste sera interprété comme un "flash_i" et résultera en une interception de l'appel:

  2 -> on_hi -> 0 ou

3 -> on_hi -> 0

- enfin, il décroche et il peut composer le numéro de son choix:

0 -> off_hi -> 5

5 -> req_ix -> 5, etc.

3.2.4 Analyse des transitions non commandables dans les états du système

Rappelons que lors du développement du générateur, les événements respectifs de l'alphabet ont été classifiés commandables ou non commandables. Les événements non commandables jouent un rôle important dans la synthèse du superviseur susceptible de mettre en oeuvre la spécification du service.

L'analyse des événements non commandables "on_hi", "on_hj", "on_hk" a déjà été faite à la section 3.1.1. Analysons à présent les autres événements non commandables de l'alphabet Σ_i que l'on retrouve dans la spécification CSMI.

À partir de l'état 5 nous avons permis tous les événements non commandables de l'alphabet. Cependant, ces événements sont associées à une transition bouclée vers l'état 5. C'est à dire qu'elles laissent l'état invariant. À cet état, "i" peut composer les numéros de son choix et donc générer des transitions telles que: "con_ix", "no_con ix" (x=j,k), "fwd_ikx", "fwd_ijx" (x = i,j,k), "flash_i". Il peut également recevoir des "req_xi" où x=j,k. Notons que "k" est traité comme un appelant quelconque.

Pour tous les états de 0 à 5 nous avons considéré comme légales les boucles {req_ki} et pour les états de 1 à 4, les boucles {req_ji}. L'interprétation de cette dernière boucle a été donnée à la section 3.2.2.3. Les boucles {req_ki} sont permises parce que nous attribuons à nouveau un rôle double au commutateur "k" qui peut à la fois recevoir des appels renvoyés de "i" en tant que NBAS, ou bien initier ses propres appels en tant qu'appelant ordinaire.

Les transitions {flash_i} sont présentes comme des boucles aux états 2, 3, 4 et 5.

3.3 Le superviseur CSMI

La théorie des systèmes à événements discrets a pour but la résolution des problèmes de commande logique pour des processus complexes. La problématique des services téléphoniques offre un territoire fécond pour l'application, la validation et l'extension de cette théorie. Après avoir convenablement conceptualisé la spécification d'un problème, il faut être en mesure de synthétiser, à l'aide de la théorie, le contrôleur du système sous considération, ainsi qu'expliqué au chapitre 2.3.1.5.

Le logiciel utilisé à des fins de synthèse est le TCT du "Systems Control Group" du Département de génie électrique et informatique de l'Université de Toronto (Wonham, 1989).

Une description de ses opérations est fournie à l'annexe I. De fait, TCT permet de calculer le langage: $\sup C(Lm(G) \cap E)$

à partir de la sous-routine "supcon":

SUP_CSMI = supcon (GIM, E(CSMI))

où

Lm(G) - le langage achevé(marqué) du générateur.

E - le langage de la spécification du service téléphonique.

C(Lm(G) ∩ E) - est la classe des sous-langages commandables du langage E par rapport à G.

sup C(Lm(G) ∩ E) - est la borne supérieure de C(Lm(G) ∩ E)(dans le sens des treillis ou des semi_treillis comme par exemple (C(E), ⊆)).

GIM - la projection du générateur G pour le commutateur "i"enrichi avec la fonction "flash".

E(CSMI) ou CSMI - la spécification du service CSMI.

SUP_CSMI - un automate représentant le plus grand sous-langage commandable du langage de spécification CSMI.

SUP_CSMI - est un automate à 132 états et 916 transitions. Le logiciel TCT nous offre la possibilité de minimiser les SEDs pour obtenir des automates équivalents aux automates initiaux mais avec moins d'états et de transitions. Il suffit d'appliquer la procédure "minstate" au superviseur CSMI:

minstate (SUP_CSMI) = SUP_CSMIM

afin d'obtenir l'automate SUP_CSMIM à 126 états et 885 transitions.

3.3.1 Concordance du superviseur CSMI avec la spécification CSMI

Le superviseur CSMI est la solution au problème de contrôle au commutateur "i" au moment où l'on implante ce service. Bien que nous ayons démarré la synthèse avec des automates de petite taille, le résultat se présente sous une forme complexe et assez difficile à manipuler. À ce stade déjà, il est aisé de reconnaître la nécessité et l'importance d'un outil théorique aidant à la résolution du problème. Il est également possible de justifier a posteriori la nécessité de limiter autant que permis la taille du mini-réseau téléphonique de base afin d'éviter l'explosion subséquente dans le nombre d'états et de transitions lors de l'implantation d'un superviseur.

Au moment où le plus grand sous-langage commandable est calculé, nous nous devons de vérifier si l'on retrouve bien dans l'ensemble des transitions permises par le superviseur, les suites typiques de transitions présentées dans la spécification du service. Nous nous préoccupons tout d'abord du passage de l'état 0 à l'état 4 que l'on identifie facilement dans la suite suivante (voir le fichier SUP_CSMI.DES):

0(0) -----> req_ji -----> 2(0) -----> fwd_jik -----> 10(1) -----> off_hi -----> 26(2)

26(2) ---> req_ik ---> 49(2) ----> con_ik -----> 73(3) -----> flash_i -----> 73(3)

73(3) -----> req_ij -----> 95(3) -----> con_ij -----> 108(4)

Les symboles entre parenthèses indiquent l'état dans l'automate de la spécification du service CSMI en correspondance avec l'état de l'automate SUP_CSMI.

Le passage de l'état 0 à l'état 5 de la spécification est retrouvée dans la suite:

0(0) -----> off_hi -----> 1(5) -----> req_ji -----> 7(5) -----> fwd_jik -----> 23(5)

(voir le fichier SUP_CSMI.DES)

3.3.2. Étude des comportements permis par le superviseur CSMI

Une fois calculé le plus grand sous-langage commandable, il est possible de poser d'autres questions sur la validité de la modélisation. Il serait par exemple intéressant de comparer l'ensemble des comportements permis par le superviseur actuel à celui obtenu en utilisant une approche alternative à la théorie standard, celle des langages achevés multiples (Wong, Thistle, Hoang et Malhamé, 1997). Dans cette approche, plusieurs langages achevés coexistent et chaque langage achevé est interprété comme une nouvelle spécification qui s'ajoute à la spécification initiale. Il faut qu'il existe un superviseur qui est non blocant par rapport à chacun de ces langages achevés. Dans le contexte présent, nous marquerons par exemple l'état 4 de la spécification CSMI (figure 3.2), puis que le mot qui comprend les événements de l'état 0 à l'état 4 représente une tâche achevée du système, c'est-à-dire une écoute suivie par une interception, la version CSMI_4 est ainsi obtenue.

Afin de s'assurer que notre superviseur est non blocant, il faudra vérifier par la suite si cette version de la spécification est non blocante par rapport au générateur GIM_M.

En termes de la théorie SED, cette vérification est réalisée par la sous-routine "nonconflict" de TCT: nonconflict (GIM_M, CSMI_4) = TRUE

Ce résultat confirme que l'état 4 de la spécification est toujours atteignable et qu'il est donc impossible que le système soit bloqué par rapport au langage achevé additionnel. Ainsi la diversité des fonctions associée au service CSMI est préservée.

3.4. Commandabilité de la spécification CSMI

La spécification CSMI synthétisée est commandable, c'est-à-dire:

condat (GIM, CSMI) = C_CSMI

contient seulement des événements commandables. Ce résultat veut dire que tous les événements qu'on devrait empêcher dans différents états de la spécification CSMI sont commandables. De plus:

nonconflict (GIM, CSMI) = true

ce qui veut dire que cette spécification est non blocante pour le générateur local GIM.

Nous devrons analyser la possibilité de simplifier cette spécification.

Supposons que l'on élimine les transitions "fwd_ixy" des états 2 et 3. Il est possible de constater que la spécification appelée C_TST n'est pas commandable:

condat(GIM, C_TST) = C_TST2C

parce qu'on a dans la liste générée les événements "fwd_iyz" dans les états 2 et 3, respectivement. Le calcul:

supcon(GIM,C_TST) = C_TSTS

donne un superviseur qui contient seulement une partie de la séquence 0-4 de la

spécification (à savoir les états 0-2):

0(0) ------> req_ji ------> 2(0) ------> fwd_jik ------> 10(1) ------> off_hi ------> 26(2)

et non pas la séquence complète comme celle de la section 3.3.1.

3.5 Commentaires sur la synthèse du superviseur CSMI

Jusqu'ici notre approche est restée fidèle à la théorie de la commande des SED: nous avons modélisé le "procédé" à contrôler, puis formulé une spécification des "suites légales" d'événements; enfin, nous avons procédé à la synthèse du superviseur le plus permissif qui répond à la spécification (en l'occurrence cette dernière étape est triviale).

Afin d'étudier la validité de cette approche, procédons à une comparaison de notre superviseur avec l'implantation préconisée dans la spécification industrielle NORTEL (1996).

Si l'on suit les étapes 1 à 6 décrites dans la spécification NORTEL, il est aisé d'obtenir l'automate de la figure 3.3 nommé CSMI2, qui englobe toutes ces étapes.

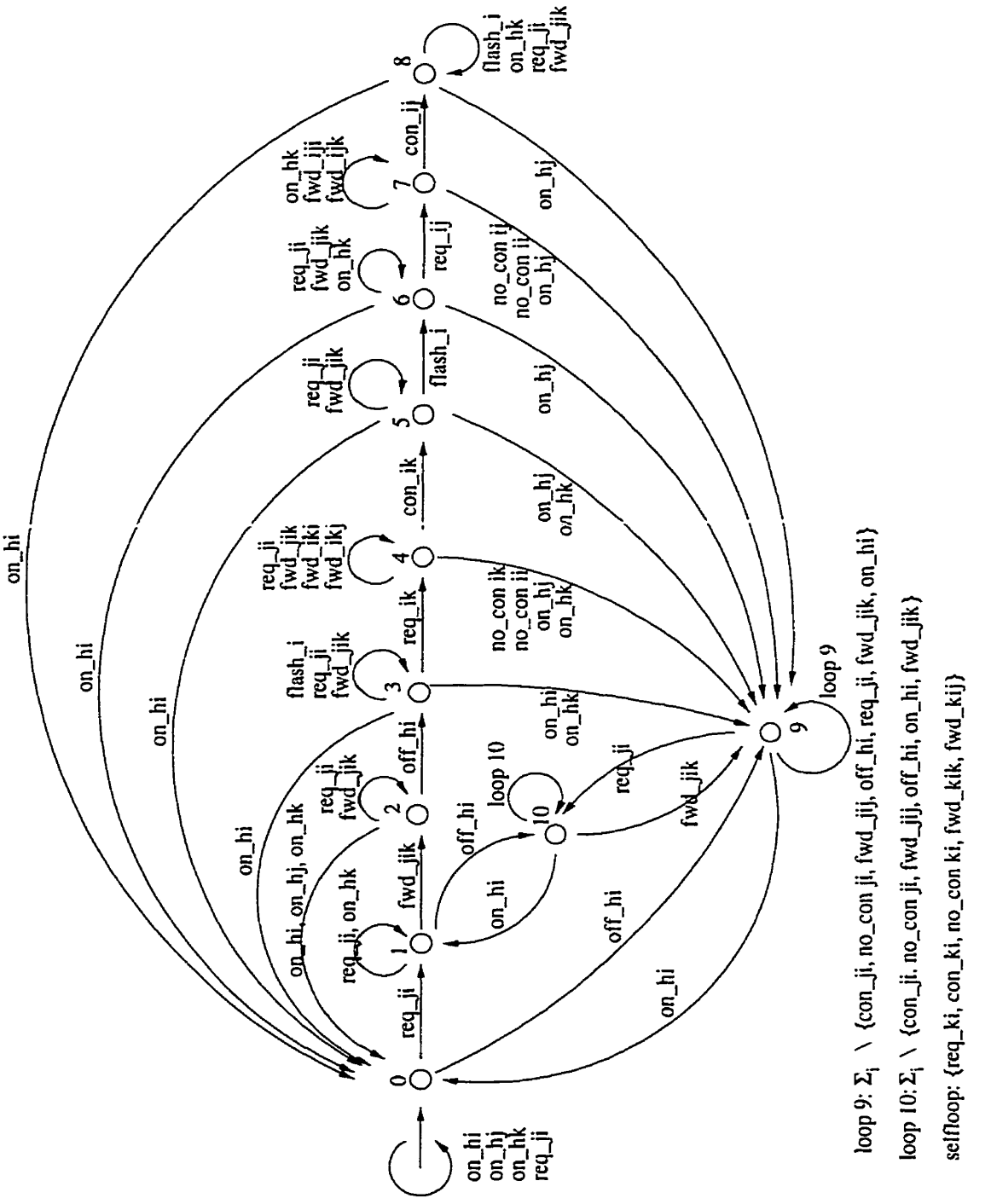


Figure 3.3 La spécification complexe du service CSMI- première version

3.5.1 Les caractéristiques de la version CSMI2

Dans cette modélisation, l'on retrouve encore une fois les deux aspects suivants du service:

- le premier aspect, celui de messagerie, est représenté par les états 0, 9 et 10 ou par les états 0 et 1.
- le deuxième aspect, celui d'écoute et d'interception, est représenté par les états 3 à 8. Les deux premiers événements représentent l'initiation de l'appel par "j" et son renvoi vers "k". L'état 2 est équivalent à l'état 1 dans l'automate de la figure 3.2. Dans l'hypothèse où "i" ne décroche pas, il y a retour à l'état initial lorsque "j" raccroche. Si "i" décroche, l'écoute commence et il y a transition vers l'état 5 qui est équivalent à l'état 3 de la figure 3.2. Si "i" fait alors le "flash_i", l'interception est déclenchée et il y a transition vers l'état 8 qui est équivalent à l'état 4 dans la version précédente.

Dans cette version, nous avons également explicité la possibilité qu'un événement "off_hi" interrompe la séquence:

req_ji fwd_jik

et l'état 10 a été ainsi ajouté.

Tous les autres événements qui paraissent comme des boucles sont indiqués à la figure 3.3.

Mentionnons en particulier l'événement "fwd_jik" qui boucle à l'état 0, pour rendre possible l'enregistrement du message (au cas où on était dans un des états 3 à 8, le "req_ji"

était produit et on revenait à l'état 0 à cause d'un "on_hi" avant que le "fwd_jik" était engendré). La boucle "selfloop" est la même que celle de la version précédente.

3.5.2. Analyse des deux spécifications

Si l'on vérifie la commandabilité du langage engendré par l'automate CSMI2, le résultat suivant est obtenu:

condat(GIM, CSMI2) = CCSMI2 commandable

Nous constatons également que:

nonconflict (GIM, CSMI2) = TRUE

Cet automate représente donc un superviseur non blocant pour le générateur local G_I. Puisque le langage reconnu par l'automate contient aussi les deux mots correspondant au déroulement typique des deux aspects du service (tel qu'expliqué à la section 3.2.1), nous pouvons le considérer comme une solution du problème de supervision analysé.

Comparons maintenant les deux solutions obtenues, c'est-à-dire le SUP_CSMI (132 états et 916 transitions) et:

SUPCSMI2 = meet (GIM, CSMI2) (162 états et 1077 transitions)

Comme les nombres d'états et de transitions sont différents, vérifions les inclusions suivantes:

(1) $M1 \subseteq M2$ et (2) $M2 \subseteq M1$

avec les sous-routines:

(2) trim (meet (A1, complement (A2,_)))

(3) trim (meet (A2, complement (A1, _)))

(où A1, A2 sont les automates qui génèrent les langages M1, M2)

Dans notre cas, on obtient les résultats:

(4) trim (meet (SUPCSMI2, complement (SUP_CSMI,_))) ≠ empty

(5) trim (meet (SUP_CSMI, complement (SUPCSMI2, _))) ≠ empty

ou de façon équivalente

(4') Lm (SUP_CSMI2) ⊄ Lm (SUP_CSMI)

(5') Lm (SUP_CSMI) ⊄ Lm (SUP_CSMI2)

Nous avons constaté qu'il y a quelques raisons pour ces différences, même si au départ les critères de conception étaient les mêmes pour les deux versions. L'une des raisons identifiées est la transition:

2 -> no_con ii -> 5

de l'automate CSMI qui se retrouve quelque peu déguisée dans l'automate CSMI2. Cette dernière transition a été introduite dans l'automate CSMI afin d'offrir la possibilité d'interrompre un appel suite à l'un des événements: "req_ik", "fwd_iki", "fwd_ikj". En même temps, dans l'automate CSMI, il est permis que le système suive cette transition sans avoir généré aucun des événements déjà mentionnés, et ce à cause du modèle local du générateur qui permet au "no_con ii" de se produire après un "off_hi" (figure 2.2). Il est

donc possible de générer avec cet automate un mot:

req_ji fwd_jik off_hi no_con ii

Dans l'automate CSMI2 par contre, les événements "req_ik", "fwd_iki", "fwd_ikj" ne font plus partie de la boucle de l'état 3 (figure 3.3) atteint suite à la transition "off_hi" et l'événement "no_con ii" n'est plus permis après le "off_hi".

Une autre différence entre les deux superviseurs est due au même événement "no_con ii" qui se produit à l'état 3 dans CSMI, mais qui ne se produit qu'à l'état 7 dans CSMI2. Dans le premier cas, cet événement est prévu comme un moyen d'avorter un appel suite à l'un des événements "req_ij", "fwd_ijk", "fwd_iji" qui sont bouclés à l'état 3. Néanmoins, et conformément au modèle du générateur, cet événement peut aussi se produire après un "con_ik". Par contre, le "no_con ii" n'a pas été prévu comme une suite possible du "con_ik" équivalent de l'automate CSMI2.

Ces deux différences peuvent être facilement éliminées si l'on ajoute à l'automate CSMI2 les transitions suivantes:

3 -> no_con ii -> 9

5 -> no_con ii -> 9

Nous trouvons toutefois que même après ces modifications une différence persiste et une analyse plus détaillée nous conduit aux résultats suivants:

- le modèle CSMI2 empêche l'interception de certains appels de la façon suivante:

si dans un des états 2 à 8 on génère une suite d'événements telle que:

off_hi req_ji on_hi ou

off_hi req_ik req_ji con_ik on_hi, etc

le système reviendra à l'état 0, où un événement "fwd_jik" peut se produire. Conformément à la spécification du service, dans cette situation, le prochain événement "off_hi" donne lieu à l'interception de l'appel en question. Comme la figure 3.3 le montre, ce traitement de l'appel est impossible dans CSMI2, parce que le "off_hi" mènera le système à l'état 9 et non à l'état 3 qui représente le premier pas vers l'état 5 (état d'interception). Dans l'automate CSMI (figure 3.2) la modélisation est plus exacte car n'importe quelle séquence: req_ji on_hi

mène le système à l'état 0 où il est alors possible de générer la séquence correcte

fwd_jik off_hi req_ik con_ik

ce qui représente l'écoute de l'appel traité par le service.

Ces remarques conduisent à la version CSMI3 (figure 3.4) qui constitue une synthèse de CSMI et CSMI2.

De son côté, le modèle simple CSMI mène lui aussi à un comportement en contradiction avec le document de spécification NORTEL (1996). En effet, conformément au modèle du générateur, une fois qu'un appel a été traité, il est possible d'engendrer un autre appel sans raccrocher. Puisque tous les événements "req_ij", "con_ij" et "flash_i" sont bouclés à l'état 3 de la figure 3.2, l'automate CSMI peut générer la séquence suivante:

fwd_jik off_hi req_ik con_ik no_con ii ...

Ceci affecte le fonctionnement du service de façon indésirable. Afin d'être sûr d'avoir une

interception après une écoute, la transition "flash_i" est introduite à l'automate CSMI (figure 3.2) menant ainsi à l'automate CSMI4 (figure 3.5).

À présent, si les calculs des plus grands sous-langages commandables pour ces deux automates sont refaits, les résultats obtenus sont comme suit:

SUP_CSMI3 = supcon (GIM, CSMI3)

SUP_CSMI4 = supcon (GIM, CSMI4)

où les deux automates ont 138 états et 953 transitions. Puisqu'ils ont la même taille, il est possible de vérifier les inclusions (1) de leurs langages par la procédure "isomorph" de TCT. Le résultat obtenu est:

isomorph (SUP_CSMI3, SUP_CSMI4) = VRAI

et les deux modèles génèrent donc des langages marqués "identiques".

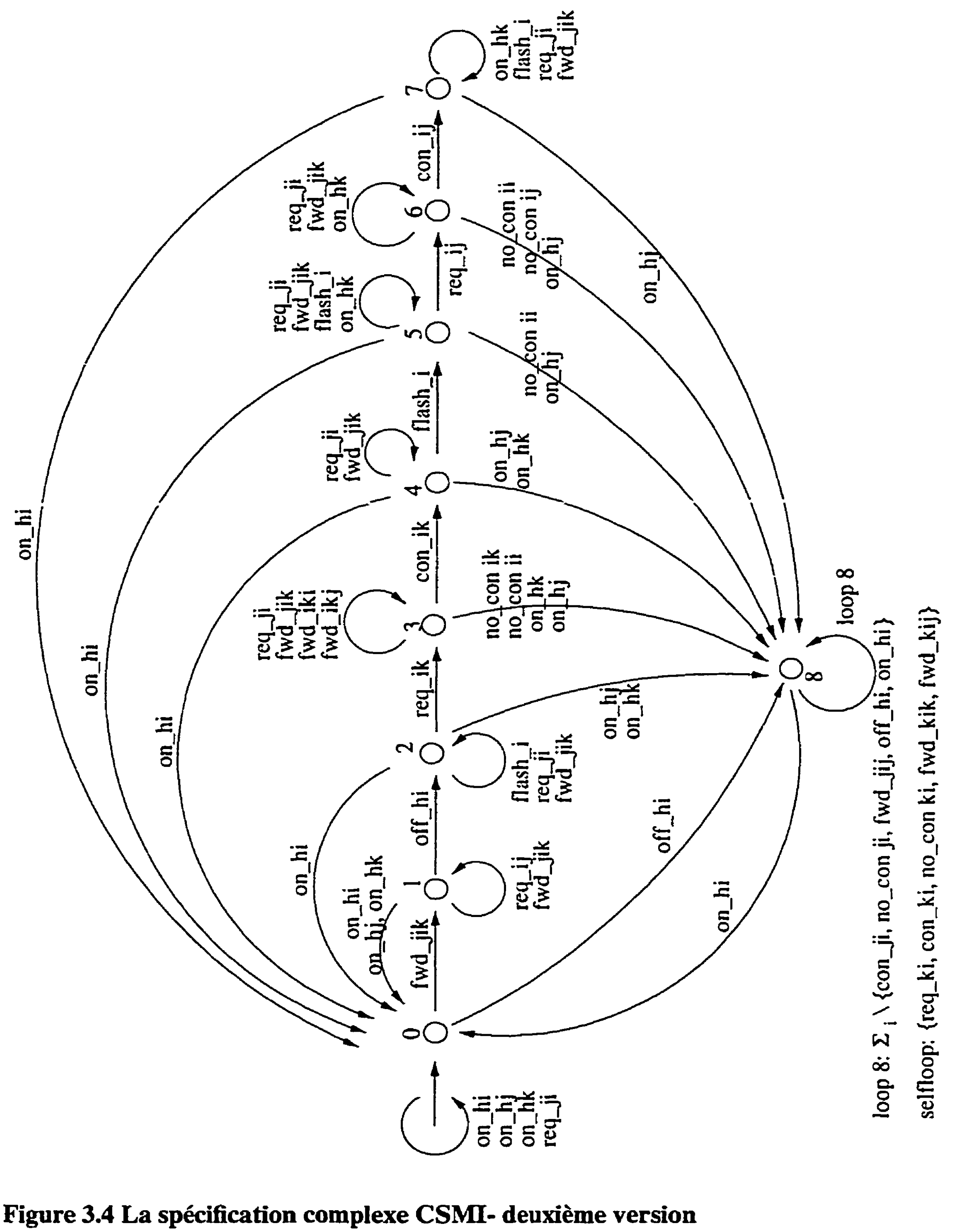


Figure 3.4 La spécification complexe CSMI- deuxième version

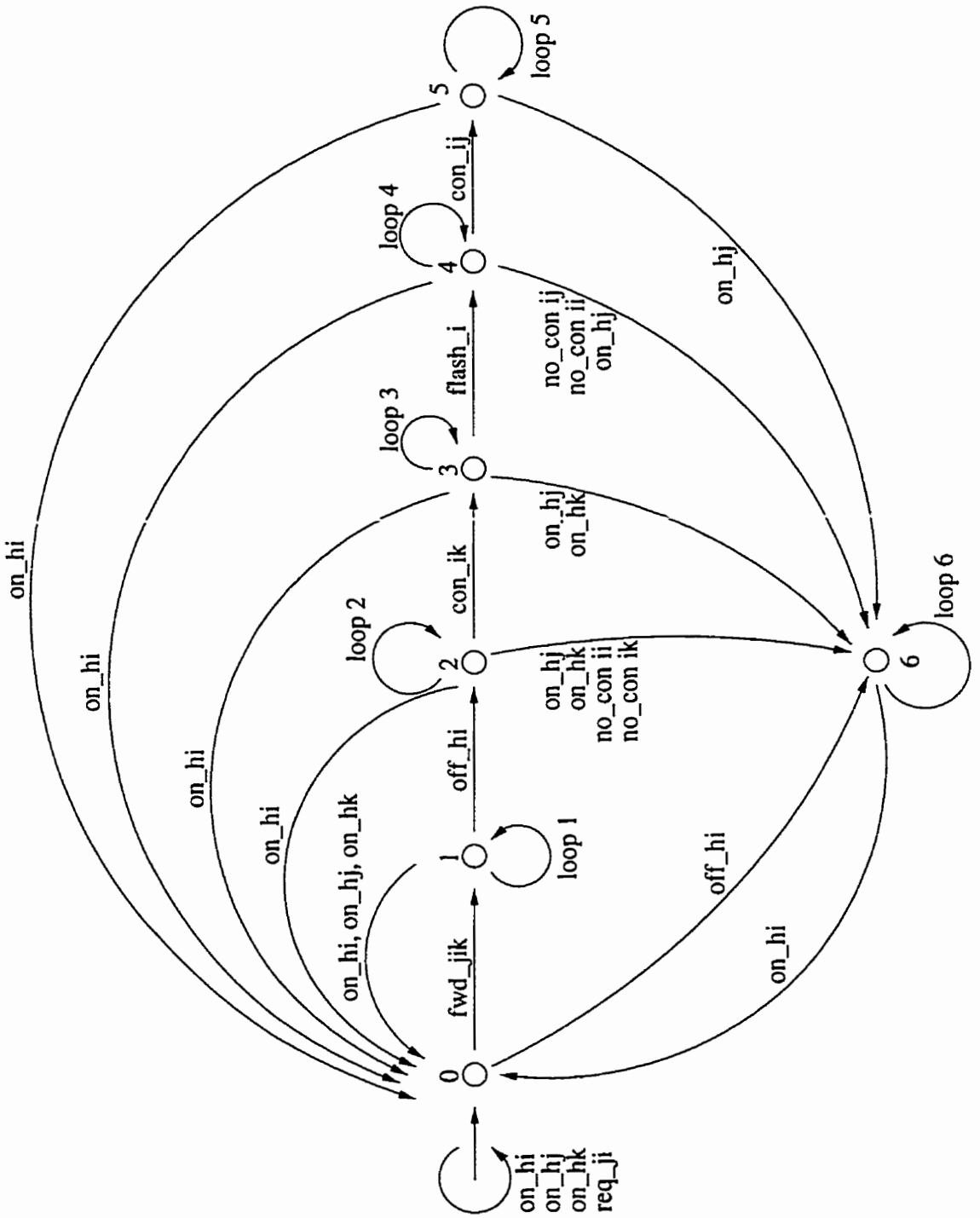


Figure 3.5 La spécification simple CSMI- deuxième version

3.6 Conclusions

Nous venons de comparer deux modèles de haut niveau du service CSMI. L'un, dénoté CSMI était obtenu à partir d'une spécification formelle du service et l'autre, dénoté CSMI2 à partir du document NORTEL. Le premier avait le mérite d'être un peu plus simple (en termes des nombres d'états et de transitions) et il permettait des suites d'événements admissibles qui étaient empêchées par le deuxième.

La modélisation de CSMI, nous a fourni des informations intéressantes pour une méthodologie à suivre au cours de modélisations futures. L'idée de départ était celle d'une synthèse de spécification très abstraite et formelle, incluant un nombre minimal de transitions, mais comprenant tous les autres événements le l'alphabet local comme des boucles pour donner un maximum de liberté au système sous contrôle. Cette approche a été confrontée avec celle d'une modélisation conforme au document de spécification NORTEL et utilisant de ce fait un modèle beaucoup plus détaillé du service analysé.

Les résultats obtenus montrent que le chemin à suivre passe par une modélisation qui tient compte de tous ces facteurs afin que l'on puisse obtenir une solution réelle. Un modèle avec un nombre restreint d'états peut être trop général pour le problème spécifique sous analyse; en même temps, un modèle trop détaillé pourrait être trop rigide pour inclure toutes les situations réelles.

La version finale obtenue CSMI4, répond à toutes les contraintes imposée par la spécification de service NORTEL, en conservant en même temps la représentation la plus simple possible.

CHAPITRE IV

LES INTERFÉRENCES DU SERVICE CSMI AVEC LES AUTRES SERVICES TÉLÉPHONIQUES

Comme nous l'avons mentionné au début de ce mémoire, chaque nouvelle implantation de services téléphoniques dans le commutateur pose éventuellement des problèmes de compatibilité avec les services existant au préalable. Le nouveau service CSMI crée des interférences avec certaines services antérieurs. Le document de spécification NORTEL (1996), décrit d'une façon assez détaillée les interférences provoquées par ce service. La théorie des systèmes à événements discrets nous donne un outil de détection de ces interférences basé sur la notion des langages conflictuels. Ainsi on peut vérifier le comportement de ce service en présence de tous les services mentionnés dans le document cité. Des recherches actuelles (WONG, 1997) essaient de donner une solution théorique à ce problème dans le cadre de la théorie SED. Ce chapitre traite d'un exemple d'interférence entre CSMI et CW résolu dans l'esprit de cette nouvelle approche.

4.1 L'interférence CSMI - CW

Comme mentionné à la section 2.3.2, on observe l'interférence de deux services au moment où on compare les plus grands sous-langages commandables générés par ceux-ci, par la procédure "nonconflict" de TCT. Le document NORTEL donne une solution d'implantation simultanée de ces deux services en les attribuant des priorités (NORTEL,

1996, page 23). Normalement les deux services sont conflictuels à cause de l'interprétation différente de l'événement "flash_i" qui peut survenir au moment où deux abonnés sont engagés dans une connexion de conversation. Prenons comme exemple le cas suivant. Il y a quatre abonnés "i", "j_1","j_2" et "k". L'abonné "i" est en conversation avec "k" (considéré d'être le NBAS) dû à l'appel de "j_1" transféré par le CSMI. À ce moment là, un nouvel appel arrive de la part de "j_2". Si "i" possède aussi le service CW et s'il fait le "flash_i", la façon dont sera interprété cet événement par le commutateur téléphonique est ambiguë: comme une demande d'interception de l'appel engendré par "j_1" ou bien un transfert de la connexion vers "j_2", conformément à CW. La spécification NORTEL donne priorité dans cette circonstance au service CW.

4.2 Survol théorique de l'approche des "filtres d'interface" (reporter maps)

L'une des solutions théoriques présentée dans la littérature (Wong, K.C., 1995) consiste dans l'assignation de priorités aux services en question, en favorisant toujours l'un des deux services. On voit ce schéma à la figure 4.1, où S_1 serait le superviseur prioritaire.

Dans ce schéma d'arbitrage, on a introduit une interface entre le superviseur de basse priorité S2 et le système à contrôler, G. Conformément à la théorie des systèmes hiérarchiques, G représente le système de bas niveau et le superviseur S2 représente le système de haut niveau. Le filtre θ fait une "synthèse" de chaque suite d'événements de

bas niveau en forme des suites d'événements de haut niveau. À haut niveau:

$$G_f = \theta\,(G)$$

est le système à contrôler pour S2 et S2' est l'agent qui interprète et implante les actions de S2. De cette façon, (θ, θ^{-1}) représente une interface entre S2 et G qui contrôle le flux d'information de G à S2 et les actions de contrôle de S2 à G.

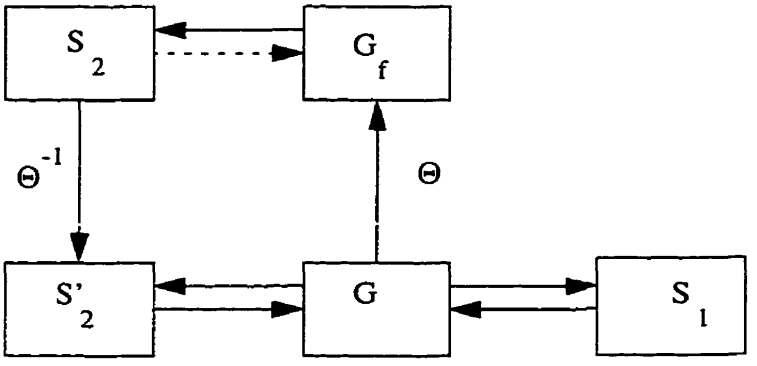


Figure 4.1 Un schéma résolution d'interférences avec des priorités fixes

L'approche proposée (WONG, 1995) consiste dans une allocation flexible de priorités telle que montré dans la figure 4.2. On définie la notion "d'interférence" au lieu de la notion de "non conflictualité" (présentée à la section 2.3.2), comme suit: on dit que deux langages K_1, $K_2 \subseteq \Sigma^*$ sont non interférents si $\overline{K}_i$ et K_j sont non conflictuels pour $i,j \in \{1,2\}$, $i \neq j$. Cette condition est plus faible que la condition de non conflictualité imposée pour les deux langages. Le problème était de trouver une expression formelle pour ces deux fonctions représentées à la figure 4.2. L'article mentionné ci-dessus donne les conditions sous quelles on peut synthétiser les deux interfaces θ_1, θ_2.

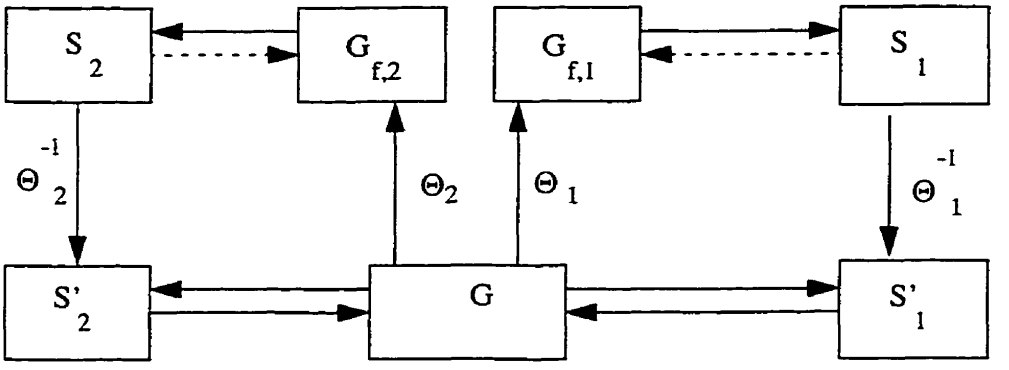


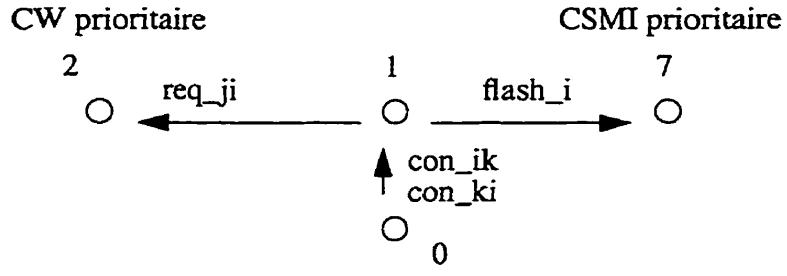
Figure 4.2 Un schéma résolution d'interférences avec priorités flexibles

4.3 Un exemple de résolution d'interférence: CSMI - CW

Le rôle des fonctions filtres mentionnées à la section précédente est de cacher certains événements de l'observation du superviseur considéré secondaire. On note en même temps qu'il ne s'agit pas d'un blocage d'événements par le superviseur.

Étant donné que la spécification CSMI est commandable, nous la prenons comme superviseur pour faciliter les calculs et réduire la taille des automates. Quant au service CW, nous utiliserons le superviseur simplifié fourni par l'article de WONG. (1997), nommé CWS (Call Waiting Supervisor). La première étape dans la synthèse est d'établir le moment où l'interférence se produit. Comme expliqué à la section 4.1, nous nous préoccupons de l'évolution du système suite à l'établissement de la connexion entre les abonnés "i" et "k". Nous accordons la priorité à CW si c'est l'événement "req_ji" qui fait

évoluer le système de l'état 1 et nous considérons CSMI prioritaire s'il s'agit d'un "flash_i" qui se produit à cet état:



La structure de transitions obtenue est celle montrée à la figure 4.3. Appelons la RP. Tous les états de cet automate sont marqués.

Dans cet automate, nous avons permis tous les événements de l'alphabet local Σ_i de se produire pour éviter d'introduire des restrictions de fonctionnement causée par la synthèse de ces filtres. En effet, nous cherchons à obtenir un résultat:

$$\text{condat (GIM, RP)} = \text{empty}$$

ce qui est vrai pour cet automate.

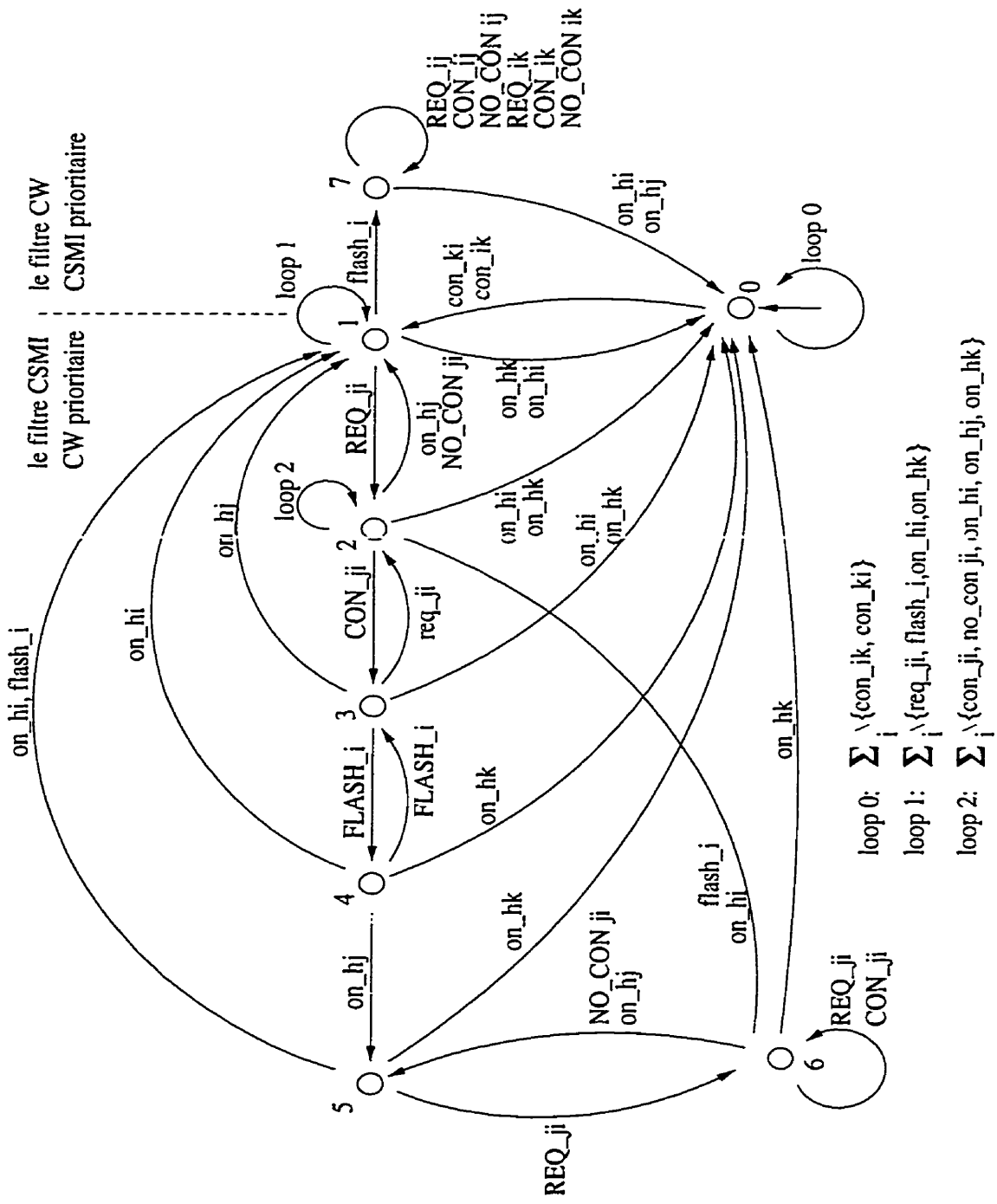


Figure 4.3 Les filtres CSMI et CW

4.3.1. La synthèse du filtre CSMI

Le filtre CSMI représente la partie gauche de la figure 4.3. où le service CW est prioritaire. Le mot:

req_ji con_ji flash_i

représente la structure de transitions caractéristique de CW pour l'établissement de la connexion avec le deuxième appelant, "j". Le "flash_i" met "k" en attente et fait la connexion de conversation avec "j". C'est utile de noter l'état des deux appelants de "i" dans les différents états de cet automate:

- état 1:	"k" parle	"j" inactif
- état 2:	"k" parle	"j" a composé le numéro de "i"
- état 3:	"k" parle	"j" est en attente
- état 4:	"k" est en attente	"j" parle
- état 5:	"k" est en attente	"j" a raccroché
- état 6:	"k"est en attente	"j" a composé le numéro de "i"

Nous remarquons les deux transitions "flash_i" entre les états 3 et 4 qui font basculer la connexion de conversation de "i" - "k" à "i" - "j". Si à l'état 4 "j" raccroche, "k" maintient son état, c'est à dire qu'il reste en attente jusqu'au moment où "i" raccroche (état 4 et 5) ou il fait un "flash_i" (selon la spécification CW).

L'état 6 a été ajouté pour la raison suivante: si à l'état 5, un "req_ji" se produit, on ne peut pas considérer que cet événement amène le système vers l'état 2 parce que la situation de

"k" est différente pour les états 2 et 5, comme expliqué ci-dessus.

La deuxième étape (THISTLE, 1996) est d'établir quelles transitions seront cachées pour le service CSMI lorsque CW est prioritaire. Ces événements apparaissent en majuscules à la figure 4.3. C'est évident qu'il faut cacher au début l'événement "req_ji" qui déclenche CW. Sinon, CSMI renverrait cet appel vers NBAS qui l'enregistrerait. En effet, cet événement sera toujours caché pour CSMI. Il semble naturel de cacher aussi tout autre événement qui suit "req_ji" pour que l'appel puisse être achevé, alors nous cachons également les événements "con_ji", "no_con ji" et "fwd_jik". Nous cachons en plus le "flash_i" aux états 3 et 4 pour qu'il ne soit pas interprété par CSMI qui, en ce cas, déclencherait l'interception. Nous considérons qu'il faut cacher aussi les "on_hi" qui mènent des états 4 et 5 vers l'état 1 parce que cet état est atteint après "con_ki" et alors l'accrochage de "i" ne représente pas la fin de la connexion. S'il était observé par CSMI, il pourrait changer l'état du système. Les deux transitions "on_hj" sont cachées pour qu'elles ne soient pas interprétées par CSMI et qu'elles créent de cette façon une interférence.

Notons aussi que nous cachons partout les événements "con_ji", "no_con ji", "fwd_jij" qui ne sont pas permis par le CSMI, justement pour qu'ils ne soient pas empêchés.

Désignons l'automate avec ces transitions cachées, RCSMI. Nous nous intéressons à calculer le produit synchrone de cet automate avec la spécification CSMI4 modifiée à ce propos en CSMI41R, de façon qu'on renomme dans celle-ci les événements qui

apparaissent en majuscule dans l'automate RCSMI.

Nous obtenons ainsi:

SYNCSMI = sync (RCSMI, CSMI41R)

En revenant à la notation normale des événements (sans éléments cachés) et SYNCSMI devient CSMI_R et on vérifie que cet automate est commandable par rapport au générateur local, GIM.

Le calcul:

condat (GIM, CSMI_R) = CCSMI_R

donne une liste d'événements commandables et donc l'automate CSMI_R remplit les conditions de commandabilité. La dernière vérification est la condition de non blocage avec le générateur local marqué, GIM_M:

nonconflict (GIM_M, CSMI_R) = true

4.3.2. La synthèse du filtre CW

Le filtre CW représente la partie droite de la figure 4.3. où le service CSMI est prioritaire. Nous remarquons la simplicité de cette partie, où nous avons considéré seulement l'événement "flash_i". Tous les autres éléments de l'alphabet Σ_i sont bouclés à l'état 7.

Nous répétons les étapes de synthèse précédentes en utilisant toujours l'automate RP. Cette fois nous cherchons à identifier les événements à cacher pour le service CW. Nous

considérons que n'importe quel appel futur doit être caché pour CW et ensuite nous cachons: "req_ij", "con_ij", "no_con ij", "fwd_iij" (qui peut être suivi par "req_ij"), "req_ik", "con_ik", "no_con ik", "fwd_iik". Nous obtenons l'automate RCW.

Le produit synchrone de cet automate avec le superviseur simple CWS modifié en CWSR pour cacher les événements ci-dessus mentionnés est l'automate:

$$SYNCCW = sync\ (RCW, CWSR)$$

En revenant à la notation normale des événements (sans éléments cachés) SYNCCW devient CW_R et on vérifie que cet automate est commandable par rapport au générateur local marqué, GIM_M. Le calcul:

$$condat\ (GIM\_M, CW\_R) = CCW\_R$$

donne une liste d'événements commandables et donc l'automate CW_R remplit les conditions de commandabilité. La dernière vérification est la condition de non blocage avec le générateur local marqué, GIM_M:

$$nonconflict\ (GIM\_M, CW\_R) = true$$

4.3.3. L'analyse des résultats

Après avoir obtenu ces deux résultats importants, nous sommes intéressé à étudier s'il y a des circonstances sous lesquelles les filtres pourraient bloquer le réseau téléphonique. Pour ce faire nous étudions ces filtres sous tous leurs aspects fonctionnels en jouant avec

les états marqués des superviseurs et du générateur local, GIM. Les cas présentés dans les sections 4.3.1. et 4.3.2 représentent la première étape où les superviseurs utilisés n'avaient que l'état initial marqué et le générateur local avait tous les états marqués. À l'étape II, nous utilisons les superviseurs marqués et le générateur local simple GIM et, suivant les mêmes étapes, nous obtenons les résultats suivants:

- pour le cas du filtre CSMI, nous utilisons le même filtre RCSMI et comme superviseur nous utilisons l'automate CSMI41R précédent sauf que nous marquons tous les états. Soit CSMI41RM l'automate résultant. Le produit synchrone de ces deux automates est:

SYCSMI2 = sync (RCSMI, CSMI41RM)

qui, après le retour aux étiquettes initiales devient CSMI_R2. La procédure "condat" de TCT donne le résultat:

condat (GIM, CSMI_R2) = CCSMI_R2

qui contient seulement des événements commandables, comme attendu. La procédure "nonconflict" donne:

nonconflict (GIM, CSMI_R2) = true

ce qui confirme la synthèse de ce filtre.

Quant au filtre CW, nous utilisons comme filtre l'automate RCW et comme superviseur l'automate CWSR qui, après qu'on lui marque les états devient CWSRM. Nous obtenons le produit synchrone:

SYNCCW2 = sync (RCW, CWSRM)

qui sera re-étiqueté en CW_R2. Par l'application de la procédure "condat" on obtient:

condat (GIM, CW_R2) = CCW_R2

qui est commandable; la procédure "nonconflict" donne:

nonconflict (GIM, CW_R2) = true

ce qui confirme cette partie de la synthèse.

À la troisième étape nous marquons tous les états des superviseurs et du générateur locale. Comme nous avons déjà calculé à l'étape précédente les produits synchrones des filtres avec les superviseurs qui ont tous les états marqués, nous répétons seulement les derniers pas, c'est à dire que nous vérifions la commandabilité de ces automates par rapport au générateur local marqué, comme suit:

- pour le filtre CSMI: condat (GIM_M, CSMI_R2) = CCSMI_R3

ce qui indique que L(CSMI_R2) est commandable.

- pour le filtre CW: condat (GIM_M, CW_R2) = CCW_R3

ce qui montre que L(CW_R2) est aussi commandable. Nous vérifions ensuite si les deux automates sont blocants pour le générateur marqué:

- pour CSMI: nonconflict (GIM_M, CSMI_R2) = true

- pour CW: nonconflict (GIM_M, CW_R2) = true

ce qui confirme aussi la synthèse pour cette version.

La liste complète des automates utilisés se trouve dans la liste des fichiers.

4.3.4. Le calcul des superviseurs non conflictuels CSMI_N, CW_N

Une dernière étape dans cette synthèse est le calcul des nouveaux superviseurs pour les deux services qui peuvent fonctionner en même temps dans le commutateur téléphonique. En ce faisant, nous obtenons:

- pour l'étape I:

S_CSMI1 = meet (GIM_M, CSMI_R) (1)

S_CW1 = meet (GIM_M, CW_R) (2)

-pour l'étape II:

S_CSMI2 = meet (GIM, CSMI_R2) (3)

S_CW2 = meet (GIM, CW_R2) (4)

- pour l'étape III:

S_CSMI3 = meet (GIM_M, CSMI_R2) (5)

S_CW3 = meet (GIM_M, CW_R2) (6)

Nous cherchons à vérifier si les différentes combinaisons de ces versions sont non blocantes, comme suit:

- les superviseurs (1) et (4):

nonconflict (S_CSMI1, S_CW2) = true

- les superviseurs (2) et (3):

nonconflict (S_CW1, S_CSMI2) = true

- les superviseurs (1) et (6):

nonconflict (S_CSMI1, S_CW3) = true

- les superviseurs (2) et (5):

nonconflict (S_CW1, S_CSMI3) = true

Comme toutes ces relations sont vraies, nous considérons qu'il n'y a pas d'interférences indésirables entre ces deux services.

CHAPITRE V

CONCLUSIONS

La modélisation du service CSMI nous a donné quelques résultats intéressants du point de vue des étapes à suivre dans ce processus dans le cadre de la théorie SED.

Premièrement, nous pouvons dire que, dans la synthèse des automates des spécifications de services, il faut tenir compte principalement des événements dits "clés" et pas nécessairement de tous les événements générés dans un certain appel. Par exemple, même s'il semble naturel de considérer comme première transition importante l'événement "req_ji", qui représente le début du processus CSMI, nous avons constaté que c'est l'événement "fwd_jik" qui doit changer l'état du système. L'analyse a été poursuivie pour déterminer par étapes successives les autre transitions significatives de l'automate de spécification. Cette analyse nous a permis de déterminer les suites "légales" qui devraient être générées par l'automate de spécification. Les deux versions présentées au chapitre III nous montrent qu'on peut trouver une version plus simple d'un automate de spécification si on se situe dans le cadre de la théorie SED, version qui est identique, du point de vue de la supervision, à la solution complexe obtenue en suivant les étapes décrites dans le document de spécification de ce service fournit par Nortel.

Pendant la modélisation, nous avons souligné l'importance de chaque événement de l'alphabet local du commutateur téléphonique dans les différentes boucles de l'automate.

Une synthèse correcte de la spécification nous permet ensuite de calculer le plus grand sous-langage commandable de la spécification, qui représente une solution au problème du contrôle du commutateur téléphonique où on implante le service.

Un autre aspect que nous avons abordé est l'interférence de ce service avec les services existants, tels que CW. Nous avons pu détecter cette interférence avec le logiciel TCT sans pouvoir indiquer l'étape où elle se produit. Dans ce cas, une analyse sommaire des deux spécifications CSMI et CW nous a permis d'identifier l'événement "flash_i" comme étant responsable de cette interférence, laquelle est due à l'interprétation différente que cet événement peut avoir dans les deux services. Nous avons proposé ensuite un schéma basé sur l'approche des filtres (reporter maps) pour pouvoir synthétiser une nouvelle version de ces deux superviseurs qui peuvent fonctionner ensemble, dans le même commutateur téléphonique. Cette méthode est basée sur un arbitrage du conflit entre services avec des priorités flexibles. Elle consiste à cacher certaines transitions à l'un des superviseurs au moment où le conflit arrive. De cette façon, nous synthétisons un "filtre" pour le service considéré en première étape de basse priorité. Après avoir calculé le produit synchrone de ce filtre avec le superviseur prioritaire, qui a été enrichi d'un état supplémentaire où bouclent les événements cachés, on revient à la notation initiale des événements qui nous permet de vérifier si l'automate ainsi obtenu est commandable et non conflictuel avec le générateur G_I (qui représente la fonction de base du commutateur téléphonique "i"). Lorsque ces résultats sont vrais, nous considérons la synthèse du filtre accomplie et nous répétons les mêmes étapes pour le cas où l'autre service est considéré prioritaire.

Les résultats obtenus dans l'analyse du service CSMI nous montrent qu'on peut arriver à une solution assez simple pour le problème de supervision dans le commutateur téléphonique. Cette première étape de recherche ouvre la voie à la synthèse de systèmes plus élaborés, qui tiendraient compte de plusieurs facteurs rencontrés dans les réseaux réels: temporisateurs, nombre de répétitions d'un événement afin de confirmer une opération, accumulation de services, etc. En même temps, cette modélisation est un outil d'analyse des principes qui ont conduit à l'élaboration du modèle de base du réseau, tels que l'alphabet local du commutateur téléphonique, etc.

RÉFÉRENCES

BOSTROM, M et ENGSTEDT, M., (1995). Feature Interaction Detection and Resolution in the Delphi framework. Feature Interaction in Telecommunications III. IOS Press, pages 157- 172.

BOUMA, L.G. et VELTHUIJSEN, H., Rédacteurs, (1994). Feature Interaction in Telecommunications Systems II. IOS Press.

Call Waiting FSD 01- 02- 1201., (1989). Bellcore Technical Reference TR-TSY-000571, Issue 1, octobre.

CAMERON, E. J. et al., (1994). A Feature Interaction Benchmark. Feature Interaction in Telecommunications Systems II. IOS Press, pages 1- 23.

CAMERON, E.J., GRIFFETH, N., LIN, Y., NILSON, M.E. et SCHNURE, W.K., (1993). A feature interaction benchmark in IN and beyond. IEEE Communications Magazine, 31(3).

CHENG, K.E. et OTHA, T., Rédacteurs, (1995). Feature Interaction in Telecommunications III. IOS Press.

CHENG, K. E., (1995). Towards a Formal Model for Incremental Service Specification and Interaction Management. Feature Interaction in Telecommunications III. IOS Press, pages 152- 164.

FACI, M. et LOGRIPPO, L., (1994). Specifying Features and Analysing Their Interactions in a LOTOS Environment. Feature Interaction in Telecommunications Systems II. IOS Press, pages 136-150.

GRIFETH, N. D., (1994). The Negotiating Agents Approach to Runtime Feature Interaction Resolution. Feature Interaction in Telecommunications Systems II. IOS Press, pages 217- 235.

KAWARASAKI, Y. et OTHA, T., (1995). A New Proposal for Feature Interaction Detection and Elimination. Feature Interaction in Telecommunications III. IOS Press, pages 127-139.

NORTEL, (1996). Feature Specification Document. Call Screening, Monitor and Intercept (CSMI), 10 janvier.

RAMADGE, P.J., WONHAM, W.M., (1989). The Control of Discrete Event Systems Procedings of the IEEE, 77(1):81- 98, janvier.

STEPIEN, B. et LOGRIPPO, L., (1995). Representing and Verifying Intentions in Telephony Features Using Abstract Data Types. Feature Interaction in Telecommunications III. IOS Press, pages 141-155.

THISTLE, J.G., MALHAMÉ, R.P. et HOANG, H.H. (1993). Modélisation et analyse de traitement des appels par l'approche Ramadge-Wonham. Rapport technique EPM/RT-93/ 15, École Polytechnique de Montréal, août.

THISTLE, J.G., MALHAMÉ, R.P., HOANG, H.H. et LAFORTUNE, S. (1996). Supervisory Control of Distributed Systems Part I: Modelling, Specification and Synthesis, soumis à une revue technique.

WONHAM, W.M et RAMADGE, P.J., (1988). Modular supervisory control of discrete-event systems. Mathematics of Control, Signals and Systems, 1:13-30.

WONHAM, W.M., (1989). On the Control of Discrete - Event Systems. Three Decades of Mathematical System Theory. A Collection of Surveys at the Occasion of the Fiftieth Birthday of Jan C.\ Willems, pages 542-562.

WONG, K.C., THISTLE, J.G., HOANG, H-H. et MALHAMÉ, R.P., (1995). Conflict resolution with flexible priority in modular control with application to feature interaction. Dans Proc.34 IEEE Conference on Decision and Control, pages 416- 421, décembre.

WONG, K.C., THISTLE, J.C., HOANG, H-H et MALHAMÉ, R.P., (1998). Supervisory control of distributed systems, Part II: Conflict resolution, (en préparation).

WONG, K.C., THISTLE, J.C. et MALHAMÉ, R.P., (1995). Conflict resolution with flexible priority in modular control. Congrès canadien de génie électrique et informatique, pages 797-800, septembre.

WONG, K.C. et WONHAM, W.M., (1992). Hierarchical and modular control of discrete-event systems. Dans Proceedings of 30th Ann. Allerton Conference on Communication, Control and Computing, pages 614- 623, septembre.

ZIBMAN, I., WOOLF, C. et O'REILLY, P., (1995). Minimizing Feature Intercations: An Architecture and Processing Model Approach. Feature Interaction in Telecommunications III. IOS Press, pages 65- 73.