MASTER EN SCIENCES INFORMATIQUES

Les contraintes logiques suspensibles dans les bases de données

Sade Elhadj, Kassoum

Award date:
1995

Awarding institution:
Universite de Namur

Facultés Universitaires Notre Dame De La Paix
Institut d'informatique
Rue Grandgagnage, 21, B-5000 Namur

LES CONTRAINTES LOGIQUES SUSPENSIBLES DANS LES BASES DE DONNEES

Kassoum Sade Elhadj

Promoteur : Pierre-Yves Schobbens

Mémoire présenté en vue de l'obtention du
diplôme de Licencié et Maître en informatique

LBS 685-295

3 of 368

Résumé

D'une manière générale, une base de données est un ensemble de faits (formules atomiques) et de contraintes d'intégrité. Une contrainte d'intégrité est alors une formule qui doit être au moins consistante avec l'ensemble des autres formules de la base de données. Cette vision masque quelque peu la distinction entre, par exemple, la contrainte selon laquelle l'*âge* d'une personne est un entier naturel (qui est une vérité dans l'univers du discours, mais peut être faux dans une implémentation particulière de la base de données), et la contrainte selon laquelle dans une classe il doit y avoir un seul professeur qui donne un cours (qui est faux dans un univers de discours, si on constate qu'il y a deux professeurs dans une même classe). La seconde est appelée *déontique* et contraint l'univers de discours; la première est une *vérité forte* de l'univers de discours et ne contraint pas cet univers; c'est plutôt une contrainte sur l'implémentation. Nous pensons que cette distinction est pertinente pour la conception de base de données, car elle impose une certaine discipline sur le concepteur de base de données. Nous montrons que ces deux types de contraintes peuvent être spécifiées dans un cadre unique, au moyen d'une variante déontique de la logique dynamique.

Abstract

Generally a knowledge base is a set of facts (atomics sentences) and integrity constraints. An integrity constraint is then a sentence which must be consistent with the other sentences in the knowledge base. This view obliterates the distinction between, for exemple, the constraint that *age* is a naturel number (which is true of the universe of discourse but may be false in a particular implementation of a knowledge base), and the constraint that a class must have precisely one teacher (which is false of the univers of discourse if actually a class has two teachers). The second constraint is called *deontic* and constrains the univers of discourse; the first constraint is a *necessary truth* of the universe of discourse and does not constrain this universe. Instead, it constrains the implementation of the knowledge base. We argue that this distinction between necessary and deontic integrity constraints is relevant and it imposes a more complicated discipline on the knowledge base designer. We show that both types of constraints can be specified in a single framework provided by a deontic variant of dynamic logic.

TABLE DES MATIERES

AVANT-PROPOS

Remerciements à
Mr. Pierre-Yves Schobbens, qui a encadré et guidé ce travail,
à son assistant J.F. Raskin qui a bien voulu consacrer un peu de son temps à ce mémoire,
ainsi qu'à l'ensemble du personnel enseignant de l'institut d'informatique;
je n'oublie pas, pour finir, de remercier toute ma famille, pour son soutien de tout instant.

INTRODUCTION

Les contraintes d'intégrité sont des propriétés formelles que les données d'une base de données doivent vérifier à tout instant. Ces propriétés font partie intégrante du modèle des données, qu'on appelle habituellement schéma de la base de données, et sont choisies de manière à décrire le plus précisément possible le réel perçu.

Si on peut dire qu'une base de données qui viole les contraintes d'intégrité de son schéma est une mauvaise représentation du réel perçu, par contre rien n'assure qu'une base de données qui respecte ces contraintes en soit une bonne, surtout lorsque la base de données est utilisée pour représenter des règles auxquelles le réel perçu doit se soumettre.

En effet cette vision en termes de schéma de données ne permet pas de faire la distinction entre les contraintes dites fortes et les contraintes dites déontiques.

Pour mieux marquer cette distinction, nous adopterons une vision différente de l'usage habituel, dans les bases de données, de la notion de modèle de base de données.

Ainsi nous adopterons la définition de M. Sergot [JoSe94] qui conçoit une base de données comme un système normatif, c'est à dire un ensemble d'acteurs interagissant (humains ou systèmes logiciels ou une combinaison des deux) dont le comportement est régi par des normes; les normes définissent comment les acteurs doivent se comporter, mais - et c'est très important - elles n'excluent pas la possibilité de mauvais comportement; alors ces normes définissent aussi ce qu'il y a lieu de faire en cas de violation de normes.

D'où le rôle des contraintes déontiques; ces contraintes ne sont pas des descriptions mais plutôt des prescriptions pour un certain univers de discours. Cette perspective de système normatif suggère l'utilisation de la logique déontique, qui est une logique modale, pour l'expression des contraintes d'intégrité.

Première partie

Chapitre 1. Les contraintes d'intégrité

1.1. Les contraintes fortes et les contraintes déontiques

Les *contraintes fortes* sont des formules qui sont vraies dans le monde réel parce que ne pouvant pas y être violées; c'est le cas des vérités analytiques et des généralités empiriques. De l'autre coté les contraintes déontiques sont des règles normatives qui s'appliquent au monde réel, et qui peuvent par conséquent être violées. De ce fait une violation de ce genre ne doit pas être enregistrée dans la base de données, seulement comme un fait parmi tant d'autres, mais bien comme la violation d'une norme, et ceci doit être réparé.

Nous allons illustré notre propos au sujet de ces différentes contraintes à l'aide de quelques exemples .
Ainsi considérons une base de données soumis à la contrainte sur l'attribut âge : "l'âge est un entier naturel " ; ceci est une vérité analytique résultant du sens des symboles qui apparaissent dans la formule, c'est une contrainte forte.
Les contraintes d'intégrité sont destinées à prendre en compte les propriétés de données dans les différents modèles de structuration de données; ce qui amène les remarques suivantes :

- Un changement d'une contrainte analytique est un changement de sens, de certains symboles y apparaissant; Par exemple, on peut décider de mesurer l'attribut "température" en dégrés Kelvin plutôt qu'en degrés centigrades; la contrainte " température$\in$R " devient température$\in$R $\wedge$ température$\geq$ 0 (puisque par définition les degrés Kelvin commencent à 0^{o}).
- Une contrainte analytique peut être utilisée pour vérifier si l'état courant d'une implémentation de la base de données représente un monde possible; si l'implémentation possède un enregistrement qui a une valeur négative pour le champ "âge", alors ce n'est pas la représentation d'un monde possible (si une implémentation représente un monde possible, il faut que ce soit une représentation correcte); ainsi une contrainte analytique contraint l'implémentation et non la base de données, car c'est une vérité forte d'un univers de discours.

Considérons l'attribut " âge" d'une personne (en années) soumis à la contrainte " âge$\leq$ 150 "; ceci n'est pas une vérité forte, mais dépend de facteurs contingents tel que, entre autres, l'état général de notre santé; c'est une *généralité empirique* à propos d'un univers de discours.

- Un changement de contrainte traduit un changement dans la façon dont se comporte un certain univers de discours.

- D'un point de vue implémentation, les vérités analytiques sont assimilées à des vérités fortes.

Le troisième type de contrainte que l'on trouve souvent dans les bases de données, concerne la formalisation de certaines règles; ainsi considérons la règle concernant une gestion de livres dans une bibliothèque : "tout emprunteur doit rendre un ouvrage au bout de trois semaines ";
Ceci n'est pas une vérité analytique qui explique le sens de la relation entre "usager", "ouvrage" et "emprunt", ce n'est pas non plus une généralité empirique à propos des usagers de la bibliothèque, mais c'est bien une règle instituée dans un univers de discours pour contraindre certains acteurs. Nous appellerons ces contraintes, *déontiques.*

- Un changement d'une contrainte déontique est un changement de norme qui concerne un certain univers du discours; on peut décider par exemple d'étendre la période de prêt d'un ouvrage à un mois.

- Une contrainte déontique est essentiellement utilisée pour vérifier si des acteurs, dans un certain univers du discours, se comportent de manière conforme aux règles établies; par rapport à la règle, une action appropriée peut-être exécutée si, par exemple, on constate que dans une classe il y a deux professeurs (en violation de la règle " dans une classe il y a précisément un seul professeur qui donne un cours").

En résumé une contrainte d'intégrité est analytique, si c'est une vérité exprimée par le sens intuitif des symboles utilisés dans sa formulation; elle est empirique si elle établit une généralité à propos d'un univers du discours, elle est déontique si elle exprime une norme, à laquelle cet univers du discours doit obéir.

En considérant la base de données comme un ensemble de mondes (états) possibles, on dira qu'une contrainte d'intégrité est statique si elle est vérifiée dans chaque état possible, et dynamique s'il faut au moins deux états pour vérifier sa validité.

1.2. **Exemple: contraintes d'intégrité.**

	STATIQUE	DYNAMIQUE
analytique	âge$\in$ N	un employé doit être embauché avant de pouvoir être licencié
empirique	âge$\leq$ 150	tous les étudiants suivent C11 avant de suivre A105
déontique	la balance d'un compte en banque doit être positive	tout emprunteur doit rendre un livre au bout de 3 semaines

Remarque

Les contraintes déontiques ont principalement un aspect dynamique, par exemple si on dit qu'un état w est interdit, alors toute transition d'un état permis vers w est aussi interdite.
De plus les obligations sont de nature dynamique, dans le sens où elles concernent les actions et non pas les états.

Chapitre 2. Logique modale

Dans le langage courant on parle souvent d'événements hypothétiques, de but à atteindre, de prévisions pour le futur; les phrases du langage peuvent alors être tantôt vraies tantôt fausses, selon les circonstances, le moment, le point de vue de chacun.
Pour prendre en compte cette notion de vérité contextuelle, on regroupe sous le nom de logique modale, différentes logiques qui sont des généralisations de la logique classique. Ces logiques font intervenir des opérateurs modaux portant sur des formules pour en modifier le sens. Ainsi la valeur de vérité des formules modales ne dépend pas uniquement de celles des formules F qu'elles contiennent, mais en outre de l'instant où F est énoncée (logique temporelle), de la personne qui pense que F ou qui croit que F (logique des croyances), ou du caractère nécessaire, possible ou aléatoire de F (logique aléthique).

Un autre champ d'application de la logique modale est la modélisation et l'analyse du comportement de systèmes considérés comme normatifs; on définit alors les modalités d'obligation de permission et d'interdiction: cette logique est la logique déontique ou logique des normes.

2.1 Logique déontique

2.1.1. Origines

La logique déontique a commencé avec les travaux de von Wright qui a construit une logique des normes, qu'on a appelé logique déontique; Le terme déontique est la traduction d'un terme grec qui signifie "obligation".
Deux idées principales se trouvent à l'origine du système proposé par von Wright. La première : l'idée de l'analogie entre l'obligation, la prohibition et la permission d'un coté et respectivement la nécessité, l'impossibilité et la possibilité. La seconde : l'idée de l'analogie entre la théorie des prédicats et la théorie des actions; Ce qui amène G. von Wright à adopter pour ses prédicats pratiques (nous appelons ainsi les noms généraux d'actions, que l'auteur représente par les variables A, B, ..) des règles syntaxiques valables pour les prédicats en général; Il en résulte que des formules comme $\neg$ A, A$\Rightarrow$ B, A$\vee$ B, A$\wedge$B, etc.. sont des expressions bien formées de OS. Les actions qu'elles soient simples ou composées, c'est-à-dire niées, alternatives, implicatives, etc... sont soit permises, défendues ou obligatoires, au moyen des expressions suivantes :
" **P**" pour est permis
" **F** " pour est défendu
" **O** " pour est obligatoire

Ces expressions sont des foncteurs (opérateurs, connectifs) créateurs de normes, à un argument prédicatif symbolisé par les variables nominales générales A, B, C, etc..; Si au départ ces variables représentaient des noms d'actions, plus tard elles représentèrent aussi des propositions p, q, r décrivant un certain état du monde.
" **P** " est l'unique terme primitif du système de la logique déontique de von Wright, appelé système ancien " OS", comme nous l'avons déjà dit précédemment.
Quant à l'idée initiale de von Wright, elle est basée sur l'observation suivante: obligation et permission sont reliées l'une à l'autre de la même manière que nécessité et possibilité : ainsi une proposition est nécessaire si et seulement si sa négation n'est pas possible, de même un état du monde (ou un acte) p est obligatoire si et seulement si non- p n'est pas permis ; on appelle obligation et permission les modalités déontiques (ou modes);
Cette idée est empruntée à Aristote.

Les notions d'obligation et d'interdiction sont définies au moyen de la permission comme suit :

$\mathbf{O}p \equiv \neg \mathbf{P} \neg p$

$\mathbf{F}p \equiv \neg \mathbf{P}p$

2.1.2. Le système OS

Il existe trois méthodes de constructions des systèmes logiques : la méthode des matrices, la méthode de déduction naturelle et la méthode axiomatique. Selon la première on compose des tables dites " matrices " posant les conditions auxquelles satisfont toutes les thèses du système et elles seules. En d'autres termes, est thèse du système donné chaque formule remplissant les conditions déterminées par les matrices construites pour le système en question. Lorsque l'on énonce les axiomes et les règles d'inférence permettant de démontrer les théorèmes, on utilise la méthode axiomatique. La méthode de déduction naturelle consiste à formuler uniquement les règles d'admission des thèses du système. G. H. Von Wright a construit son OS, d'abord selon la méthode des matrices; mais plus tard, lorsqu'il l'a remplacé par son nouveau système NS, il a reconstruit OS selon la méthode axiomatique.

Le système comprend les axiomes et règles suivantes :

OS_0 Toutes les tautologies du calcul des propositions

OS_1 $\mathbf{O}p \equiv \neg \mathbf{P} \neg p$

OS_2 $\mathbf{P}p \vee \mathbf{P} \neg p$

OS_3 $\mathbf{P}(p \vee q) \equiv \mathbf{P}p \vee \mathbf{P}q$

OS_4 $p \equiv q \ / \ \mathbf{P}p \equiv \mathbf{P}q$

Remarque

OS_2 est appelé le principe de permission selon lequel:
pour tout acte p, p ou non-p est permis, ce qu'on exprime aussi par " il n'est pas vrai que p et non-p sont obligatoires ", autrement dit: à l'impossible nul n'est tenu.

OS_3 est le principe de distribution déontique

Comme indiqué plus haut il existe quelques analogies avec la logique modale aléthique, ainsi les deux axiomes ci-dessus ont leurs correspondants:
" p ou non- p est possible", ou encore " p et non- p ne peuvent à la fois être impossibles "
Il existe aussi un principe de distribution.
Cette analogie est cependant limitée; en effet la logique déontique n'accepte pas le principe de necessitation, selon lequel " si p est nécessaire alors p est vrai "; à la place correspond une notion plus faible de ce principe : "tout ce qui est obligatoire est permis" ce qui s'écrit aussi

$\mathbf{O}p \Rightarrow \mathbf{P}p$

et qui se prouve comme suit :

$\mathbf{P}\neg p \vee \mathbf{P}p$ d'après (OS_2)

$\neg(\neg\mathbf{P}\neg p) \vee \mathbf{P}p$

or $\mathbf{O}p \equiv \neg\mathbf{P}\neg p$

$\neg(\neg\mathbf{P}\neg p) \vee \mathbf{P}p \equiv \neg(\mathbf{O}p) \vee \mathbf{P}p \equiv \mathbf{O}p \Rightarrow \mathbf{P}p$

2.1.3. Quelques réserves au sujet de la logique déontique OS de Wright

L'analogie entre les propositions normatives et les propositions modales, s'ajoutant à celle entre la théorie des prédicats et celle des actions, aboutit à une transposition mécanique des notions d'une théorie dans le domaine de l'autre, et en fin de compte à l'invention d'un système formalisé, qui théoriquement parlant, pourrait avoir une application importante pour la morale et le droit, mais en réalité ce système ne capte pas toujours nos intuitions dans ce domaine.
Ainsi ce qu'on appelle le paradoxe du libre choix:

OS_3 $\mathbf{P}(p \vee q) \equiv \mathbf{P}p \vee \mathbf{P}q$

Si on nous dit que nous pouvons faire ceci ou cela, nous comprenons normalement que nous pouvons faire ceci, mais aussi cela; en d'autres termes le principe de distribution semble être :

P(p$\vee$ q) $\equiv$ **P**p $\wedge$ **P**q ; mais ce principe va de pair avec l'idée du " permis" autre que l'idée obéissant au schéma " **P** = $\neg$ **O**$\neg$ " . C'est ce qui est appelé la permission forte.

Certains considèrent que l'on outrepasse l'analogie heureuse entre l'aléthique et le déontique, lorsque l'on adopte des expressions comme **O**(P$\rightarrow$Q) ou **O**(P $\vee$ Q), car ce qu'elles signifient n'est que construction intellectuelle et non un produit d'abstraction induit du réel, de la pensée normative, surtout morale et juridique. Il est en effet difficile de trouver des normes ayant la structure de **O**(P$\rightarrow$Q) ou **O**(P $\vee$ Q); par contre on peut rencontrer des normes du genre:

O(p)$\rightarrow$ **O**(q) " si x doit payer l'impôt sur le revenu, alors il doit choisir l'un des modes de paiement ".

Un problème important est celui de la formalisation de l'obligation dérivée (**O**(P$\rightarrow$Q))
Le système de Wright admet les thèses suivantes:

FA$\rightarrow$ **O**(A$\rightarrow$B)

OB $\rightarrow$ **O**(A$\rightarrow$B)

lues respectivement " si A est non permis (autrement dit défendu) alors (si A, alors B) est obligatoire ", et " si B est obligatoire, alors (si A, alors B) est obligatoire.
Ce qui veut dire d'une part, qu'un acte prohibé nous oblige à accomplir n'importe quel autre acte, fut-il également prohibé, et de l'autre que tout nous oblige à accomplir l'acte obligatoire.
Ces thèses constituent les paradoxes de l'obligation dérivée. Mais finalement Wright avoue en 1956 que **O**(A$\rightarrow$B) n'est pas une expression adéquate en langage symbolique de la notion d'obligation dérivée (commitment).

2.2. Le système standard de logique déontique SDL

Très largement inspiré du système de Wright, c'est un système modal normal KD, selon la classification de CHELLAS[Che80]; il comprend les axiomes et règles suivant :

KD_0 Toutes les tautologies du calcul des propositions.

KD_1 **O**(p$\supset$q) $\supset$(**O**p$\supset$**O**q) axiome K

KD_2 **O**p$\supset$ **P**p axiome D

KD_3 **P**p $\equiv \neg$ **O**$\neg$ p

KD_4 $\mathbf{F}p \equiv \neg \mathbf{P}p$

KD_5 Modus ponens: $p, p \supset q / q$

KD_6 **O**-necessitation : $p / \mathbf{O}p$

Nous donnons quelques théorèmes de KD

1. $\mathbf{O}p \equiv \neg\mathbf{P}\neg p$
2. $\mathbf{O}(p \wedge q) \equiv \mathbf{O}p \wedge \mathbf{O}q$
3. $\neg(\mathbf{O}p \wedge \mathbf{O}\neg p)$ " il ne peut avoir de conflit entre obligations "
4. $\mathbf{P}(p \vee q) \equiv \mathbf{P}p \vee \mathbf{P}q$ " permission unilatérale "
5. $\mathbf{O}p \vee \mathbf{O}q \supset \mathbf{O}(p \vee q)$
6. $\mathbf{O}p \supset \mathbf{O}(p \vee q)$ " Paradoxe de Ross "
7. $\mathbf{P}(p \wedge q) \supset \mathbf{P}p \wedge \mathbf{P}q$
8. $\mathbf{F}p \supset \mathbf{F}(p \wedge q)$ " Paradoxe du pénitent "
9. $\mathbf{O}q \supset \mathbf{O}(p \supset q)$ " obligation dérivée "
10. $\mathbf{O}\neg p \supset \mathbf{O}(p \supset q)$ " obligation dérivée "
11. $\neg p \supset (p \supset \mathbf{O}q)$ " paradoxe de l'obligation conditionnelle "
12. $\neg\mathbf{O}(p \wedge \neg p)$ " pas d'obligations contradictoires "
13. $(\mathbf{O}p \wedge \mathbf{O}(p \supset q) \wedge (\neg p \supset \mathbf{O}\neg q) \wedge \neg p) \equiv$ false " le paradoxe de Chislom "
14. $p \supset q / \mathbf{O}p \supset \mathbf{O}q$ " le paradoxe du bon samaritain"

Remarque

Ce que l'on considère comme paradoxe c'est le fait que les déductions que l'on peut faire dans SDL ne correspondent pas souvent à notre intuition première.
Les théorèmes 3. et 12. ne sont pas réellement des paradoxes, mais ils expriment le fait, quelque peu irréaliste, que l'on ne peut être confronté à des obligations conflictuelles.
On retrouve aussi le paradoxe du libre choix, comme dans OS, avec le théorème 4.
Le théorème 6. concerne le paradoxe de Ross [Ross41], illustré par l'assertion suivante " si l'on est obligé de poster une lettre, alors on est obligé de poster la lettre ou de la brûler" ; ceci n'est pas en accord notre entendement en langage naturel, concernant l'obligation du choix entre des actions.
Le théorème 8. induit des conséquences absurdes, ainsi un exemple de lecture libre de
$\mathbf{F}p \supset \mathbf{F}(p \wedge q)$ peut-être celui-ci: " s'il est interdit de commettre un crime, alors il est aussi interdit de commettre un crime et de purger une peine lorsque l'on a commis un crime ".
Le théorème 9. a des conséquences peu heureuses, si on considère $\mathbf{O}(p \supset q)$ comme la formalisation de l'obligation conditionnelle; on déduit ainsi que si q est obligatoire alors toute autre action arbitraire nous oblige à faire q.
La même lecture erronée de $\mathbf{O}(p \supset q)$ dans 10., implique que "s'il est interdit de commettre un meurtre, alors en commettant un meurtre on est obligé de commettre n'importe quel autre acte (exemple le suicide), ce qui est absurde.
De même, dans 11. si on considère $p \supset \mathbf{O}q$ comme une obligation conditionnelle, on obtient:
" s'il n'est pas vrai que l'on a sonné à la porte, alors si on sonne à la porte on est obligé de tuer quelqu'un.
Néanmoins certains de ces paradoxes n'apparaissent plus lorsque l'on prête suffisamment attention à la signification véritable des formules, par exemple en considérant leur interprétation dans la sémantique des mondes possibles du système standard.
Mais parmi les plus sérieux, il y a 13. et 14. (cf. [Chis63, Aqv67]).

Selon 14. on peut dire que :
" si X aide Y, qui a eu un accident, alors Y a eu un accident; on dérive de ceci que :
" si X est obligé d'aider Y, alors Y est obligé d'avoir eu un accident ";
ce qui n'est certainement pas vrai.

2.2.1. Sémantique du système standard SDL

Comme les autres logiques modales la sémantique du système standard est basée sur la notion de mondes possibles [HuCr68, Che80]. Ce que l'on appelle un modèle de Kripke, consiste en un ensemble S de mondes possibles, d'une fonction π qui attribue des valeurs de vérité à des propositions primitives dans chaque monde, et d'une relation $R \subseteq S \times S$ qui relie chaque monde à un ensemble de mondes possibles, qui sont des mondes alternatifs parfaits.
L'idée derrière cette formulation est, qu'étant dans un monde possible (le monde actuel) l'on peut lui associer un ensemble de mondes alternatifs parfaits, dans lequel toutes les normes seraient satisfaites. Les opérateurs **O**, **P** et **F** sont traités comme des opérateurs modaux associés à la relation d'accessibilité R, comme en logique modale.

Un modèle de Kripke M = (S, π, R) s$\in$S , s est un monde possible
Op est vrai dans le monde s si et seulement si p est vrai dans tous les mondes alternatifs de s donnés par la relation d'accessibilité R.
Pp est vrai si et seulement si il existe un monde alternatif où p est vrai
La validité est définie comme d'habitude en logique modale:
p est valide par rapport à une classe de modèles C si p est vrai dans tout monde de tout modèle de Kripke de C.

2.2.2 Critiques du système standard

- Si A est un théorème alors **O**A est aussi un théorème; de ce fait cette logique considère que les obligations existent toujours, quand bien même seraient-elles triviales. Mais il semble raisonnable de considérer qu'il existe des mondes possibles, où il n'existe pas d'obligation.

Op $\supset$ **P**p peut s'écrire aussi $\neg$ (**O**p $\wedge$ **O** $\neg$p)

ceci est considéré comme inconsistant dans le système; par conséquent d'un point de vue sémantique il est impossible de distinguer le fait qu'une obligation entraîne une permission .
D'autre part la sémantique des mondes possibles ne permet pas de représenter de façon satisfaisante les obligations secondaires (contrary-to-duty-obligation) [Chis63]; d'où le recours à un raisonnement en logique non monotone, qu'exige la modélisation d'un tel type de contrainte; nous y reviendrons plus tard.

2.3. Approche dyadique : le système NS [VW65]

Le système NS a été proposé par von Wright pour mieux traiter l'obligation conditionnelle, qui est la source de nombreux paradoxes dans l'approche OS.
La logique déontique primitive de von Wright était une logique monadique : les foncteurs déontiques, créateurs de normes, n'y avaient qu'un argument nominal, représenté par les variables A, B, etc, (remplacé plus tard par un argument propositionnel p représentant un certain état des choses). La nouvelle logique de von Wright sera dyadique : les foncteurs créateurs de normes y auront pour argument (tantôt nominal, tantôt propositionnel) une relation, un couple, une dyade représentée par des fonctions telles que A/B ou p/c, etc... , selon le cas ces fonctions sont lues respectivement "A sous condition de B " et "p sous condition de c " .

Le système comprend les axiomes et règles suivantes:

NS_0 toutes les tautologies du calcul des propositions

NS_1 **O**(p$\wedge$q/r) $\equiv$ **O**(p/r) $\wedge$ **O**(q/r)

NS_2 $\mathbf{O}(p/q \vee r) \equiv \mathbf{O}(p/q) \wedge \mathbf{O}(q/r)$

NS_3 $\neg(\mathbf{O}(p/q) \wedge \mathbf{O}(\neg p/q))$

NS_4 $\mathbf{P}(p/q) \equiv \neg\mathbf{O}(\neg p/q)$

NS_5 $\mathbf{O}p \equiv \mathbf{O}(p / T)$

NS_6 Modus Ponens

NS_7 Les règles de substitution

Dans ce système on reconnaît un certain nombre de théorèmes et axiomes de OS mais sous une forme conditionnelle (NS_1, NS_2, NS_4);
Cependant il a été démontré que NS permet de dériver les mêmes paradoxes que OS .

2.4. Réduction de la logique déontique à la logique modale aléthique

Anderson[Ande58]

Alan Ross suggère de réduire la logique déontique à la logique modale aléthique, au moyen du schéma suivant:

G_1. $\mathbf{O}p \equiv \mathbf{N}(\neg p \Rightarrow S)$

Où "S" est une constante propositionnelle, **N** est l'opérateur modal de nécessité.
"S" est interprété par Anderson comme une pénalité ou une sanction, qui résulte d'un manquement à une obligation; ainsi d'après G_1, p est obligatoire si et seulement si non-p entraîne (nécessairement) une sanction; dit autrement, p est interdit si et seulement s'il entraîne une sanction. Notons aussi les schémas suivants:

G_2. $\neg$ **N**S toute sanction est évitable

G_3. $\mathbf{P}p \equiv \mathbf{M}\ (p \wedge \neg S)$

M est l'opérateur modal (aléthique) de possibilité ($\mathbf{M}p \equiv \neg\mathbf{N}\neg p$)
D'après G3. Un état p est permis si et seulement si il est compatible avec l'absence de pénalité "S".

2.5. Raisonnement en logique non monotone

Le problème fondamental avec les paradoxes, qui semblent persister (le paradoxe du bon Samaritain, le paradoxe de Chisholm [Chis63]) en SDL, est que la sémantique des mondes possibles n'est pas assez flexible, car seulement deux types de mondes sont considérés: le monde actuel et le monde idéal.

Les mondes idéaux doivent satisfaire toutes les obligations, et lorsque ces obligations sont en conflit, cela constitue un problème. Par exemple dans le cas du paradoxe de Chisholm, on déduit **O**(t) et **O**($\neg$t), or aucun monde idéal ne peut satisfaire aussi bien t et $\neg$t; d'où le paradoxe. Il y a donc nécessité d'adopter une notion de mondes sous idéaux, dans lesquels certaines normes mais pas toutes, seraient satisfaites; au moyen de cette distinction il est possible de définir un ordre de préférence entre ces états sous-idéaux, et d'appliquer un raisonnement non monotone pour trancher en cas de conflit entre obligations (cf. [Rei87], [AM81, AB81]).

2.6. Réduction de la logique déontique à la logique dynamique (Meyer1988)

Inspirée par la méthode d'Anderson de réduction à la logique modale aléthique, cette méthode utilise aussi un atome spécial V pour indiquer qu'une violation d'une contrainte déontique a été commise.

La logique dynamique [Harel, 84] est une extension de la logique propositionnelle avec un opérateur modale [α] pour chaque action α. Ces actions sont soit atomiques ou composées au moyen d'opérateurs comme la composition séquentielle(;), le choix non déterministe ($\cup$), et la composition parallèle (&).

Une expression [α] φ est lue comme ceci " après α, φ ". La sémantique formelle est donnée au moyen de la structure de Kripke, où une relation d'accessibilité R_α est associée à chaque action α.

Dans cette approche, α est permis (**P**), interdit (**F**), obligatoire (**O**) sont exprimés en logique dynamique comme suit:

F(α) $\equiv$ [α] V
P(α) $\equiv \neg$ **F**(α) ($\equiv <\alpha> \neg$ V)
O(α)$\equiv$ **F**(-α) ($\equiv$ [-α] V) (-α est la non-exécution de α)

Intuitivement, une action α est interdite si et seulement si l'exécution de α produit un état dans lequel V est vrai; α est permis si et seulement si α n'est pas interdit (si et seulement si il existe une façon d'exécuter α qui ne mène pas dans un état où V est vrai); α est obligatoire si et seulement si ne pas l'exécuter est interdit.

L'approche de réduction de la logique déontique à la logique dynamique offre l'avantage d'éliminer certains paradoxes qui existent dans le système traditionnel de déontique logique(cf [Meyer87, Meyer88, Meyer89, And67, Cas81]).

2.7. Réduction de la logique déontique à la logique linéaire temporelle

Si les formules déontiques sont utilisées pour contrôler et guider le comportement des acteurs d'un certain univers du discours, il a été proposé une interprétation des concepts déontiques à l'aide des modalités temporelles. Dans [FM92, GAB76] Fiadeiro et Maibaum proposent une réduction (sémantique) des spécifications déontiques aux spécifications temporelles. Essentiellement, ils interprètent l'obligation de faire une action p, comme la propriété temporelle selon laquelle p doit survenir au moins une fois dans un instant futur. Dans cette approche, il n'y a aucune relation directe entre permission et obligation : l'obligation de faire une action ne signifie pas qu'il n'est pas permis de ne pas faire l'action. Les auteurs utilisent un fragment de la logique linéaire temporelle, avec les opérateurs suivants :

X : dans l'état suivant l'état actuel
F : un jour dans le futur, fatalement
G : toujours dans le futur, désormais.

Chapitre 3. Applications de la logique déontique en informatique

La logique déontique trouve d'abord ses premières applications dans le domaine juridique, celui de la morale philosophie, pour l'usage normatif du langage; elle est utilisée non seulement dans la modélisation des règles de loi et des institutions sociales, mais aussi pour la spécification des systèmes d'information, les besoins de sécurité dans les réseaux, les contraintes d'intégrité dans les bases de données etc. ...; dans des études récentes, plusieurs applications de la logique déontique ont été proposées. Nous passerons en revue un certain nombre d'entre elles.

Nous définirons tout d'abord les éléments constitutifs d'une structure de toute application informatique. (cf. MW 93) .
Un système informatique est un système qui stocke et manipule un certain nombre de données; ce système peut -être une base de données, un système expert, un système de base de connaissance, un système d'aide à la décision, un système d'exploitation, etc...
Chaque type d'application informatique contient des données qui représentent une partie de la réalité, que nous appellerons le système-objet ou encore l'univers du discours; les utilisateurs du système informatique fournissent des données, émettent des requêtes et reçoivent en retour des réponses du système informatique. Les utilisateurs et le système font partie d'une organisation; l'univers du discours pouvant alors partiellement être à l'intérieur et partiellement à l'extérieur de l'organisation (fig.a.).

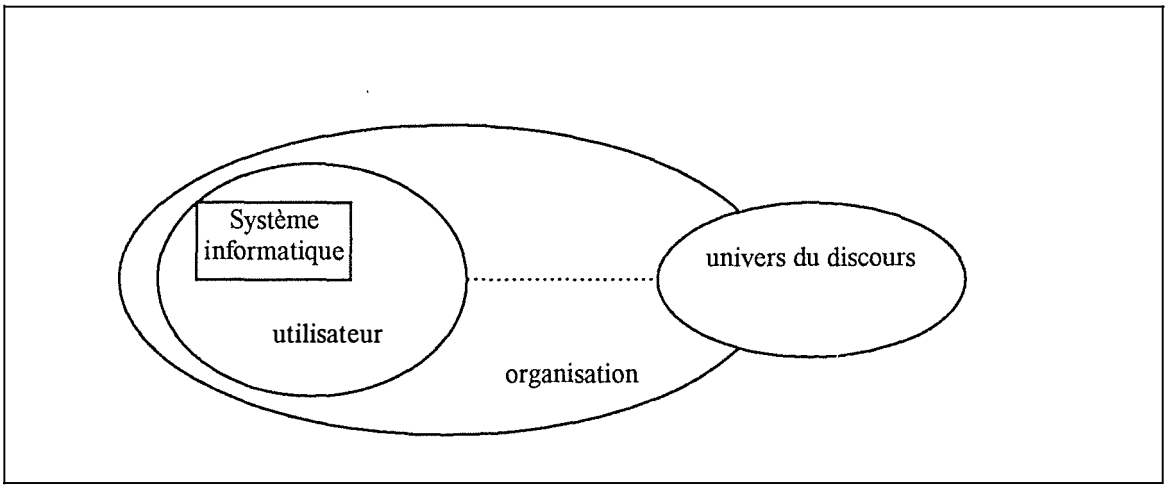

Fig.a Structure d'une application informatique

Sur la base de cette structure (fig.a.) on peut classer les différentes applications de la logique déontique à l'informatique en considérant différents domaines, dont le comportement est spécifié en logique: le système informatique, ses utilisateurs, l'organisation ou l'univers du discours. Dans chaque cas on spécifiera aussi bien le comportement réel que le comportement idéal; nous considéreront successivement :

- les systèmes informatiques tolérant aux pannes

- le comportement normatif d'un utilisateur

- le comportement normatif dans ou de l'organisation
    - spécification d'une stratégie
    - comportement normatif de l'organisation

- le comportement normatif de l'univers du discours

3.1 Les systèmes informatiques tolérants aux pannes

Tout système informatique peut être sujet à des défaillances, et dans de nombreux cas il est important de pouvoir spécifier les actions à entreprendre en cas de comportement anormal du système, comme par exemple une panne d'un composant matériel.
Les systèmes tolérants aux pannes autorisent un fonctionnement qui, sans être idéal, est acceptable compte tenu de certaines circonstances; la logique déontique peut être utilisée dans la spécification de la gestion des cas d'exception dans le comportement du système informatique.

3.2. Spécification d'un comportement normatif pour un utilisateur

Les utilisateurs d'un système informatique ont toutes sortes de comportements, parmi lesquels certains ne sont pas souhaitables: ils peuvent enfoncer les mauvaises touches, fournir des données inconsistantes avec les données déjà stockées, refuser de fournir les données que le système demande, ou poser des questions qu'ils ne sont pas autorisés à formuler, etc....Indépendamment du fait de savoir si un tel comportement est détectable par le système informatique (parfois il l'est, d'autres fois il ne l'est pas) on doit pouvoir faire une distinction claire entre le comportement souhaité de la part d'un utilisateur et son comportement réel; de cette manière on simplifie la spécification du comportement attendu, car on intègre aussi la spécification de mesures à prendre lorsque l'utilisation du système n'est pas idéale.
Cette application de la logique déontique peut être considéré comme la spécification de la gestion des erreurs, qui sont générées par les utilisateurs.

3.3 Spécification du comportement de et/ou dans l'organisation

L'application de la logique déontique dans la spécification du comportement de l'organisation concerne d'une part la définition de règles internes de gestion et d'autre part son attitude externe.

3.3.1 Politique interne

En considérant l'organisation dans son ensemble, la logique déontique est utilisée pour décrire le comportement des différents composants, tels que les employés ou les départements; ceci est fait dans la politique interne de l'organisation qui définit les lignes de conduite pour les uns et les autres. Dans ces cas on s'attellera à définir la démarche à suivre en cas de manquement aux règles établies; la logique déontique permet de définir sans ambiguïté ces règles et d'explorer leurs conséquences. Ce genre d'application est identique à ce qui se passe dans le domaine juridique, à ceci près que les règles internes ne sont pas des lois et ne relèvent que du domaine privé d'une organisation.
Nous donnons l'exemple de [MW 93], qui concerne les mécanismes d'autorisation.
Il s'agit dans ce cas d'octroyer à certains agents la permission d'exécuter certaines actions; ainsi en informatique, ces mécanismes sont utilisés pour protéger l'intégrité de certaines ressources dans les systèmes d'exploitation et les bases de données, pour la sécurité de certaines données stratégiques.
N. Minsky et A. Lockman pensent que les mécanismes d'autorisation sont déficients parce qu'ils n'intègrent pas la notion d'obligation.
Mais d'une part le concept d'obligation d'exécuter une action est pertinent, indépendamment même de celui de permission; par exemple le début d'une transaction dans une base de données nous oblige à protéger les données pour empêcher aux autres utilisateurs d'y accéder pendant la transaction. De même l'exécution du *Begin* de transaction contient une obligation implicite de terminer la transaction dans un délai raisonnable par un *Commit* ou un *Rollback* de cette transaction. Ainsi en exécutant une action un acteur est, de fait, soumis à une certaine obligation.
D'autre part, accorder des permissions, sans stipuler en même temps des obligations, est inadéquat comme mécanisme d'autorisation. Ainsi considérons les exemples suivants.

- Traditionnellement la permission d'exécuter une action, par exemple lire un fichier ou mettre à jour un champ dans une base de données, est accordée à un acteur sans soumettre ce dernier à aucune obligation, lorsqu'il exécute cette action. Cependant dans de nombreux cas cette obligation existe de façon implicite; par exemple, si une bibliothèque accorde à un lecteur la permission d'emprunter un ouvrage, il y a, associée à cette permission, l'obligation implicite de rendre l'ouvrage endéans un certain délai. Par conséquent ceci doit être spécifié conjointement avec la spécification de la permission d'emprunter.

- Certaines contraintes dans les systèmes informatiques peuvent être temporairement violées, mais ces violations créent de nouvelles obligations, celles de restaurer l'état normal.

Par exemple, supposons que l'on accorde à un manager la permission de répartir les tâches entre le personnel, avec la contrainte qu'un poste de travail stratégique ne doit jamais rester vacant plus de cinq jours. Nous accordons ainsi une permission, mais nous stipulons dans le même temps une obligation.

3.3.2. Comportement normatif de l'organisation

On peut aussi spécifier le comportement d'une organisation dans son environnement en utilisant la logique déontique. Le système de gestion de commande proposée par Lee [Lee88] en est un exemple; et parce que le comportement d'une organisation doit être conforme à certaines règles légales, ceci peut-être considéré comme un cas spécial d'application de loi au moyen de la logique déontique.

Bon de commande électronique [Lee88]

Lee et Nees [MW 93] ont souligné le fait que certains documents commerciaux avaient non seulement un contenu informatif, mais aussi un aspect contractuel; par exemple un bon de commande client contient des informations, comme le nom, l'adresse d'un clients, l'identification des marchandises commandées, etc; dans son aspect contractuel, il constitue un engagement pour le fournisseur, qui doit livrer les marchandises, et pour le client, qui doit payer la facture une fois les marchandises livrées; cet aspect est souvent matérialisé par une signature ou tout autre moyen. Depuis l'avènement des systèmes informatiques, les informations sont stockées et manipulées automatiquement; par exemple le fournisseur peut être relié à ses clients par un système d'EDI (système d'échange de données électroniques). Cependant les méthodes traditionnelles de développement de systèmes d'information n'intègrent pas les aspects contractuels. Les modèles de données utilisés, sont des sous ensembles de la logique du premier ordre, pour représenter la structure des données et leur manipulation. Il y a donc une nécessité d'étendre ces méthodes, pour pouvoir traiter des aspects contractuels. Kimbrough et *al.*[MW 93] proposent l'utilisation de la logique déontique pour la représentation des mécanismes impliqués dans des contrats; quelques exemples d'actions, jugées pertinentes, qui doivent être représentées par la logique :

Oblige :une action qui n'était pas obligatoire, le devient ;
Waive : une action obligatoire ne l'est plus;
Permit : une action interdite devient permise;
Forbid : une action permise devient interdite;

En 1988, Lee traduit cette spécification en un système servant de moniteur à toute une séquence d'activités incluses dans un contrat [Lee88]. Un point important dans les contrats concerne les délais, qui à leur tour impliquent la représentation des processus en temps réel. Lee utilise les réseaux de Petri pour donner une sémantique de ces processus et la logique du Changement, développée par Von Wright [vW65], pour leur représentation en logique. La logique du temps absolu présenté par Rescher et Urquhart [MW 93], est utilisée pour représenter les dates limites. Les opérateurs déontiques sont ajoutés en utilisant la réduction de Anderson [And67] de la logique déontique à la logique aléthique. Cette spécification est traduite dans un langage Prolog-exécutable, auquel une interface en langage naturel est ajoutée. Il permet la spécification formelle de contrat comme suit :

Mr X accepte de payer 50fb à Mr Y le 3 Mai 1996,
Suivant cela,
Mr Y accepte de livrer une machine à laver à X, dans une semaine.
Le système peut être interrogé comme suit:

?- que se passe-t-il le 5 Mai si de rien n'était.
Ce à quoi le système répond:
La partie X a failli au 4 Mai 1996, car X n'a pas payé les 50fb à Y le 3 Mai 1996.

Notons qu'aucune sémantique formelle n'est donnée pour le langage de spécification.

3.4. Spécification du comportement de l'univers du discours

L'univers du discours est une partie de la réalité, qui sera représentée dans le système informatique. Cette réalité peut tout aussi bien être une bibliothèque qu'un système d'ascenseurs ou tout autre application.

3.4.1. Spécification de règles juridiques au moyen de la logique déontique

Le domaine juridique constitue une vaste aire d'application de la logique déontique, avec ou sans l'informatique; l'univers du discours dans ce cas est constitué d'un ensemble de personnes dont le comportement est gouverné par un certain nombre de lois. Les faits et les lois sont formulées et manipulées selon les règles d'une certaine logique déontique, pour inférer des conclusions.
Un tel système peut-être considéré comme un système-expert servant de tuteur pour des étudiants en sciences juridiques ou comme un système d'aide-conseil dans le processus d'application de lois; Sergot [MW 93] fournit quelques exemples.

3.4.2. Simulation du raisonnement juridique

L'objectif ici est de simuler le processus de raisonnement par lequel juges et avocats accomplissent leurs tâches; l'univers du discours dans ce cas n'est pas un système de lois et de personnes, mais le processus psychologique qui se déroule dans le cerveau d'éminents spécialistes (juges et avocats), car ce processus implique lui aussi des normes et des faits, d'où l'utilité de la logique déontique; cependant la représentation de l'univers du discours dans le système informatique ne peut en aucun cas être considéré comme une représentation des lois; c'est plutôt la représentation de la manière dont juges et avocats font usage des lois. Dans cette application, la logique déontique n'est pas utilisée pour prescrire le comportement de l'univers du discours, mais comme un médium dans lequel s'expriment des hypothèses empiriques à propos de la manière dont un tel comportement (raisonner à propos de systèmes normatifs) se déroule chez les hommes de loi.

3.4.3. Spécification de contraintes d'intégrité déontiques

La spécification de contraintes déontiques est un exemple de formalisation des règles de l'univers du discours ; c'est le sujet principal de ce mémoire.
Dans le même ordre d'idées, Khosla et Maibaum propose un langage de spécification de système d'information. Ces auteurs font remarquer à propos de certains langages de spécification formelle, (style VDM) qu'il y a deux usages différents de la notion de précondition d'une action, qu'il convient de dissocier.

- En effet, d'une part la précondition d'une action est utilisée pour identifier le contexte dans lequel l'action est exécutée, de sorte que la postcondition puisse définir de manière déclarative le résultat de cette exécution; dans cette optique, si une action n'a pas de précondition cela voudrait dire que le résultat de son exécution ne dépend pas du contexte.
- D'autre part, les préconditions sont utilisées pour signifier l'autorisation d'exécuter une action; dans ce cas une absence de précondition signifie qu'il est toujours permis d'exécuter une action.

Khosla [KM87, Kho88] propose d'utiliser les préconditions, seulement pour définir le contexte d'exécution d'une action, de façon à aboutir au résultat défini dans la postcondition;
Et l'on utiliserait la logique déontique pour exprimer les conditions dans lesquelles, il est permis d'exécuter une action.
Ainsi en séparant la spécification de la permission d'exécuter une action, de la spécification des pré et postconditions, la spécification des effets d'une action est simplifiée.
D'autre part ceci permet la spécification des systèmes tolérants aux pannes, dans lesquels une panne ne serait-ce que celle d'un composant matériel, est toujours possible; la logique déontique permet de spécifier quelle action corrective il faut entreprendre pour restaurer ou tout au moins permettre un fonctionnement à peu près acceptable.
Khosla et Maibaum définissent une extension de la logique modale des actions, appelée logique déontique des actions. Dans cette logique tout état du système est soit permis, soit interdit.
Tout d'abord, dans un état permis du système, les actions sont autorisées si et seulement si elles mènent dans des états permis, et elles sont interdites si et seulement si elles mènent dans des états interdits. L'on ne dit pas dans quel cas une action, qui débute dans un état interdit, est permise ou interdite. Si le système est dans un état interdit, on peut spécifier toute action comme étant interdite, ou permise seulement pour une action qui mène dans un état permis du système, ou de façon sélective, autoriser certaines actions, même si elles ne mènent pas le système immédiatement dans un état permis.
Utilisant cette logique, Khosla et Maibaum spécifie un système téléphonique. Quelques exemples d'axiomes de cette spécification sont :

1. [Ex, BusyInd(t)] BusyTone(t)

2. [Ex, RingInd(t)] RingTone(t)

3. [Ex, RingBell(t)] BellRinging(t)

4. Busy(t') $\rightarrow$ [Ex, Connect(t,t')] **O**(Ex, BusyInd(t))

5. $\neg$ Busy(t') $\rightarrow$ [Ex, Connect(t,t')] **O** (Ex, RingInd(t) & RingBell(t'))

Nous donnons une lecture informelle de la modalité " [ ] "; ainsi la syntaxe de l'exécution d'une action est: [Acteur, Action]Etat du monde, l'exécution de "Action" par "Acteur" produit "Etat du monde"
Les trois premiers axiomes spécifient le résultat de trois actions de façon indépendante du contexte, parce que il n'y a pas de précondition.
Axiome 1. le système téléphonique (Ex) exécute une action pour informer l'abonné (t) que son correspondant est occupé.
Axiome2. : Ex informe l'abonné t que le téléphone de son correspondant est entrain de sonner
Axiome 3. : Ex exécute une action, qui indique à un abonné que son téléphone sonne.
Les deux derniers axiomes spécifient ce que le système doit faire si l'appelé est occupé, et lorsque celui-ci est disponible; dans le premier cas, il informe l'appelant que le correspondant est occupé; et dans le second cas il doit informer en même temps les deux correspondants de l'établissement de la connexion.

La logique déontique des actions est basée sur la logique modale des actions, et contient des opérateurs comme la composition séquentielle, le choix, pour combiner des actions en des processus plus complexes.

3.4. 4. Autres applications : problème d'ordonnancement

Dans ce type de problèmes plusieurs tâches requièrent les mêmes ressources pour s'exécuter, mais ces ressources sont limitées; c'est par exemple le cas de plusieurs bases de données qui doivent exécuter périodiquement des processus, toutes les semaines par exemple, avec des contraintes qui imposent un ordre séquentiel et un temps limite pour l'exécution de ces processus; les ressources matérielles étant limitées, ces contraintes ne sont pas toujours respectées et on doit pouvoir spécifier les actions à entreprendre lorsqu'une contrainte n'est pas respectée.

2éme PARTIE

Cette deuxième partie est inspirée de l'article de MEYER et Wieringa [WIER 89] , dont elle contient de larges extraits; ainsi nous reprenons pour notre compte la notion de modèle présenté dans cet article, et pour finir l'étude de quelques exemples se fera au moyen de la logique definie par Meyer.

Chapitre 1. Notion de modèle

Pour comprendre la classification des contraintes d'intégrité en contraintes statiques, dynamiques et déontiques, nous devons avoir une vision claire du rôle des modèles dans la conception et l'utilisation des bases de données. La vision que nous adoptons ici est principalement celle utilisée en sciences naturelles. C'est pourquoi nous l'appelons modèle descriptif.

1.1. Modèle descriptif

Un *modèle descriptif* d'un univers du discours est un système abstrait qui représente les entités de cet univers ainsi que leur comportement à un certain niveau d'abstraction; le système possède un espace état et une fonction de transition entre états, qui décrit comment il évolue dans cet espace.

Remarque

Nous allons préciser certains concepts contenus dans cette définition.

- L'univers du discours d'une base de données est habituellement un système social (ou partie de) comme une entreprise, mais peut tout aussi bien être une réalité technique comme un système d'ascenseurs .

- Le modèle descriptif représente l'univers de discours : on voudrait avoir, autant que faire se peut, une représentation fidèle, avec une certaine approximation; le modèle est abstrait en ce sens que tout dans l'univers du discours n'est pas modélisé; d'où certaines dissemblances qui font qu'on peut dire que le modèle soit une représentation incorrecte ou incomplète de l'univers du discours. Mais même en sciences naturelles, on étudie les masses et les gaz idéaux, qui représentent des masses et des gaz du réel, avec un certain dégrée d'approximation.
Une idéalisation similaire a lieu dans les bases de données; par exemple si on doit représenter la couleur des yeux de la personne X, comme étant soit bleu ou vert, bien que ce soit plutôt bleu-

vert, aucune des représentations n'est correcte, si on prend en compte la différence entre le bleu, le bleu-vert et le vert. Et si nous ne représentons pas toutes les propriétés de la personne X (la base de données ne représente pas, en général, toutes les propriétés des objets du réel) alors dans ce sens la base de données est un modèle incomplet de l'univers de discours.

- Le modèle descriptif est un modèle abstrait, qui est considéré comme un système dans le sens de la théorie des systèmes, qui dans le contexte d'une application de base de données est équivalent au concept de machine dans la théorie des automates finis; en cela, un modèle est un ensemble de variables d'état dont les valeurs définissent un point dans l'espace état, et une fonction de transition entre états, qui associe à chaque paire (état, entrée) une paire (prochain-état, sortie) (dans les systèmes non déterministes, on associe un ensemble de paires (prochain état, sortie) à chaque paire(état, entrée)); nous considérons seulement des variables d'état discrètes, de sorte qu'on peut parler de fonction de transition (par opposition à fonction d'évolution d'état).
Dans la communauté des bases de données ce qu'on entend par modèle, c'est habituellement un schéma de données, c'est-à-dire un ensemble définissant une relation avec des contraintes d'intégrité. Un schéma de relation est proche de ce qu'on appelle en logique formelle une signature, c'est-à-dire une liste de symboles, utilisés dans le langage formel, et pour chaque symbole de prédicat on donne son arité et si possible le type de ses arguments.
Par contraste, nous utilisons le concept de modèle d'abord dans un sens physique: ainsi chaque entité e de l'univers de discours est représentée par un objet o, qui est un modèle de e, (s'agissant de l'univers de discours, nous parlerons d'entités et s'agissant de la représentation abstraite de cet univers de discours, nous parlerons d'objets).
Dans la fig.1. Les objets O_1 et O_2 représentent les employés e_1 et e_2. Habituellement les variables d'état d'un objets sont appelés ses attributs. A un niveau élevé d'agrégation, la base de données elle-même est un modèle de l'univers de discours car c'est un ensemble d'objets abstraits, chacun étant un modèle d'une entité de l'univers de discours; ainsi $\{O_1, O_2\}$est une base de données qui représente des informations à propos de e_1 et e_2 . Un état de la base de données est caractérisé en donnant :

1. L'ensemble des objets abstraits qui existe dans cet état de la base
2. L'état de chaque objet existant

On suppose ici que l'univers de discours comprend des états qui peuvent être représentés de façon abstraite par des états du modèle; et de façon similaire les entités dans l'univers de discours ont des états qui peuvent être représentés par les états d'objets abstraits.
Chaque système a une fonction de transition entre états. Pour définir une fonction de transition d'un modèle, nous devons spécifier les événements de mise à jour (transition simple entre états) et les processus (composés de séquences, de choix et d'exécutions parallèles).
Quand un ensemble d'objets forment une base de données, ces objets s'échangent des messages, ce qui complique la définition de la fonction de transition de la base de données. Wierenga et Van Riet [WR 88] utilisent des algèbres de processus pour spécifier les processus exécutés, aussi bien par les objets que par la base de données. Nous utiliserons la logique dynamique pour spécifier les objets et la dynamique de la base de données.

Il est à noter que nous distinguons un modèle d'un univers de discours, qui représente une base de données abstraite , et l'implémentation du modèle, qui est une base de données concrète sur une machine particulière. Notre propos concernera ici uniquement les bases de données en tant que modèles d'un univers de discours.

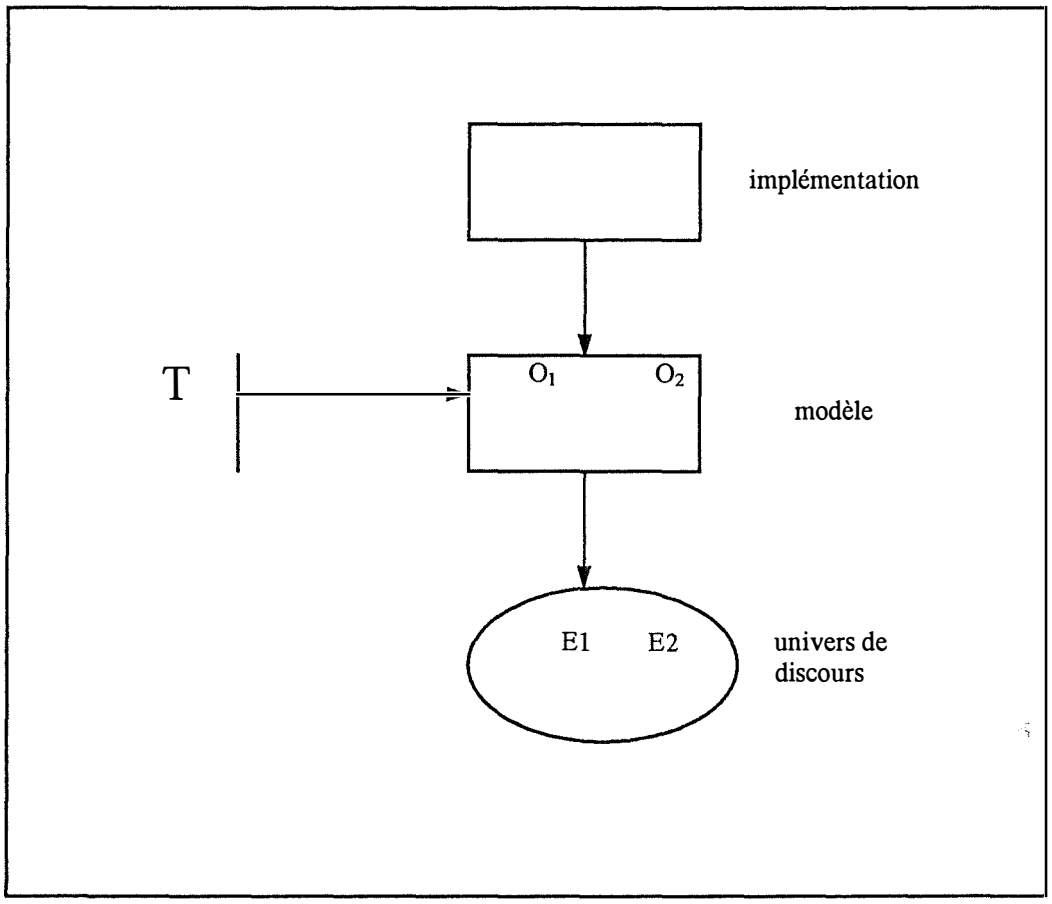

Fig.1. Objets et entités

A cette notion physique de modèle nous rajoutons une notion logique, selon l'usage courant dans la théorie des bases de données (Nicolas et Gallaire [NiGa 78]); le modèle de la fig.1 est une structure abstraite dans laquelle un langage formel L est interprété. Il y a un ensemble T de formules de L, qui sont vraies dans le modèle, appelé *théorie* du modèle.

Nous avons ainsi une vision théorique de la base des données (par opposition à une base de données comme un ensemble de formules qui sont vraies dans un modèle), attribuant au mot modèle deux sens : c'est d'abord une abstraction d'un certain univers de discours et ensuite c'est un modèle logique d'un ensemble de formules.

Notre modèle sera ensuite une prescription pour un univers du discours.

Notre langage sera une extension déontique de la logique dynamique (Meyer[Meye88]), et nos modèles seront tous des structures de Kripke. Utilisant le jargon de la logique modale, l'espace état de la base de données sera appelée l'ensemble des mondes possibles; un état particulier de la

base de données est un monde possible[Hughes et Creswell [HuCr68]); on utilise un prédicat d'existence E pour exprimer l'existence d'un objet dans un monde.

1.2. Modèle prescriptif

Pour comprendre les contraintes déontiques, il est essentiel de comprendre le relation entre un modèle contenant des contraintes déontiques et l'univers du discours. La figure simple d'une base de données comme un modèle descriptif qui a été dessinée plus haut doit donc être étendue pour couvrir l'aspect prescription de la base de données. On établira la distinction entre un modèle descriptif et un modèle prescriptif en utilisant le concept de "direction de convenance", notion que nous empruntons à Seale [Se 85].

Définition

La *direction de convenance* entre deux entités est une flèche, qui indique l'entité à laquelle l'autre entité doit s'ajuster, se conformer, dans le cas où il y a un manque de correspondance entre les deux. Si la direction de convenance est de A vers B, alors B est dit *normatif* pour A.

Remarques

Cette définition suppose qu'il existe un concept significatif entre les deux entités. Dans notre cas nous pouvons considérer les cas suivants, lorsqu'il y a :
- manque de correspondance entre une description et ce qui est décrit ;
- manque de correspondance entre la prescription et l'entité dont le comportement est prescrit.
Dans chaque cas, la direction de convenance est dans le sens où on peut trouver le "maître", qui est l'entité à laquelle l'autre doit s'ajuster.

Nous voyons ainsi qu'un modèle descriptif d'un univers de discours a une direction de convenance, du modèle vers l'univers du discours (fig 1).
La direction de convenance de T vers le modèle (fig.1), reflète le fait que d'après notre conception de modèle, T en tant que théorie du modèle doit être adapté aux changements dans le modèle; nous devons définir d'abord l'univers du discours avant d'utiliser une théorie formelle pour le décrire.

Dans la conception de base de données, nous chercherons les contraintes d'intégrité qui décrivent un modèle de l'univers du discours, plutôt que de chercher un modèle dans lequel nous pouvons interpréter ces contraintes d'intégrité.
La direction de convenance entre une implémentation et un modèle est de l'implémentation vers le modèle, de sorte que chaque modèle joue deux rôles : c'est une description d'un univers de discours, mais c'est une prescription pour une implémentation.

Définition

Un *modèle prescriptif* pour un univers de discours est un système dont les états et le comportement sont des normatifs pour cet univers.

Remarque

On parle de modèle prescriptif pour un univers de discours, et au contraire, de modèle descriptif de l'univers du discours; alors que les modèles descriptifs sont utilisés comme en sciences naturelles, qui étudient les systèmes comme on les trouve dans la nature, les sciences de l'ingénierie mécanique, qui étudient la construction des systèmes qui satisfont les exigences d'un modèle prescriptif, utilisent aussi bien les modèles prescriptifs que les modèles descriptifs.
La relation entre les deux types de modèles, utilisés en ingénierie est illustrée par la figure 2.

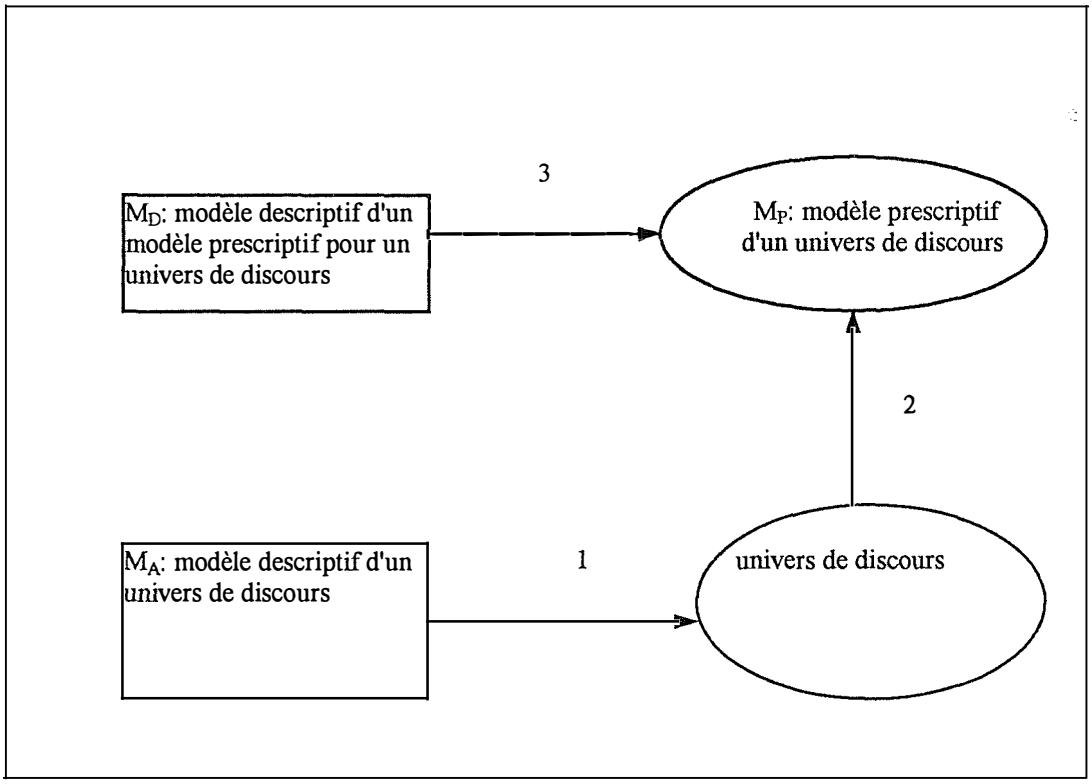

Fig.2 Modèle descriptif et modèle presciptif

La distinction entre M_D (modèle descriptif de M_P) et M_P nous permet de faire remarquer un fait important à propos des contraintes déontiques, M_D est une abstraction de M_P; donc toutes les normes de l'univers de discours ne sont pas représentées dans M_D.

La distinction entre M_D et M_P correspond à deux façons d'interpréter " il est interdit de stationner ici " : c'est la promulgation d'une règle ou l'observation qu'une règle existe (logique des énoncés sur les normes ; Von Wright [vW63], p.53).

- D'un côté une contrainte déontique interprétée comme une formule de M_P est la promulgation d'une norme pour le comportement de l'univers de discours et comme telle, elle peut être violée dans cet univers.
- De l'autre côté, si on l'interprète comme une formule de M_D, c'est une vérité forte à propos d'un ensemble possible de mondes, et ne peut être violée par le comportement des objets dans ces mondes; (ainsi il est vrai, en dehors de tout changement de norme, " qu'il est interdit de stationner ici" dans les états possibles du monde; ce qui n'empêche pas certains automobilistes d'enfreindre la règle dans l'univers de discours.

Pour illustrer la différence entre les trois modèles de la fig.2., considérons ce qui se passe dans le cas où il y a un manque de correspondance le long de chacune des trois flèches.

- Un manque de correspondance sur 1 signifie que M_A (modèle descriptif de l'univers de discours) est faux et doit être ajusté.

- S'il existe un manque de correspondance le long de 2. on essaie d'adapter l'univers de discours, pour être conforme au modèle prescrit; mais si ceci s'avère impossible, ou coûteux financièrement, ou exige un temps trop long, on révisera les normes, c'est-à-dire qu'on affaiblira le modèle prescriptif, de telles manière que les exigences des normes ne soient pas trop pesantes pour l'univers de discours. Par exemple si nous découvrons que, dans la phase de conception d'un modèle, un coût élevé pour remplir toutes les exigences en matière de qualité d'un système d'ascenseurs, on peut réduire ces exigences.
Ce fait rend la situation en ingénierie mécanique plus flexible, et donc plus complexe : en cas de manque de correspondance entre modèle prescrit et l'univers de discours, on peut adapter aussi bien l'univers de discours que le modèle; ceci n'affectera en rien la direction de convenance entre le modèle prescrit et l'univers de discours prescrit (cf. fig.3). Quand nous adaptons nos normes à un niveau abordable, nous ne faisons que remplacer le modèle prescriptif par un autre modèle prescriptif, auquel l'univers de discours doit se conformer.

- Considérons 3, la situation est encore plus compliquée; en cas de manque de correspondance entre le M_A et l'univers de discours actuel, on doit se demander si nous modélisons de manière incorrecte le modèle prescriptif (flèche 3.) ou si l'univers de discours n'arrive pas à remplir ses obligations (flèche 2); dans le premier cas nous devrions améliorer M_D, dans le second, nous devrions améliorer l'univers de discours ou rabaisser M_P.

Pour illustrer ces idées, considérons la simulation des files d'attente à un comptoir d'un supermarché, comme cela se passe dans le domaine réel; cette simulation est une implémentation de M_A; nous évaluons cette simulation par rapport à un certain comportement souhaité M_P, que

nous formalisons au moyen de paramètres adéquats (pour obtenir M_D); enfin nous comparons M_A et M_P; ce que nous devons préciser et qui n'apparaît pas sur la fig.3, c'est qu'il y a plusieurs M_D (modèle descriptif de normes réduites) avec différents paramètres, et nous choisissons, le mieux indiqué.

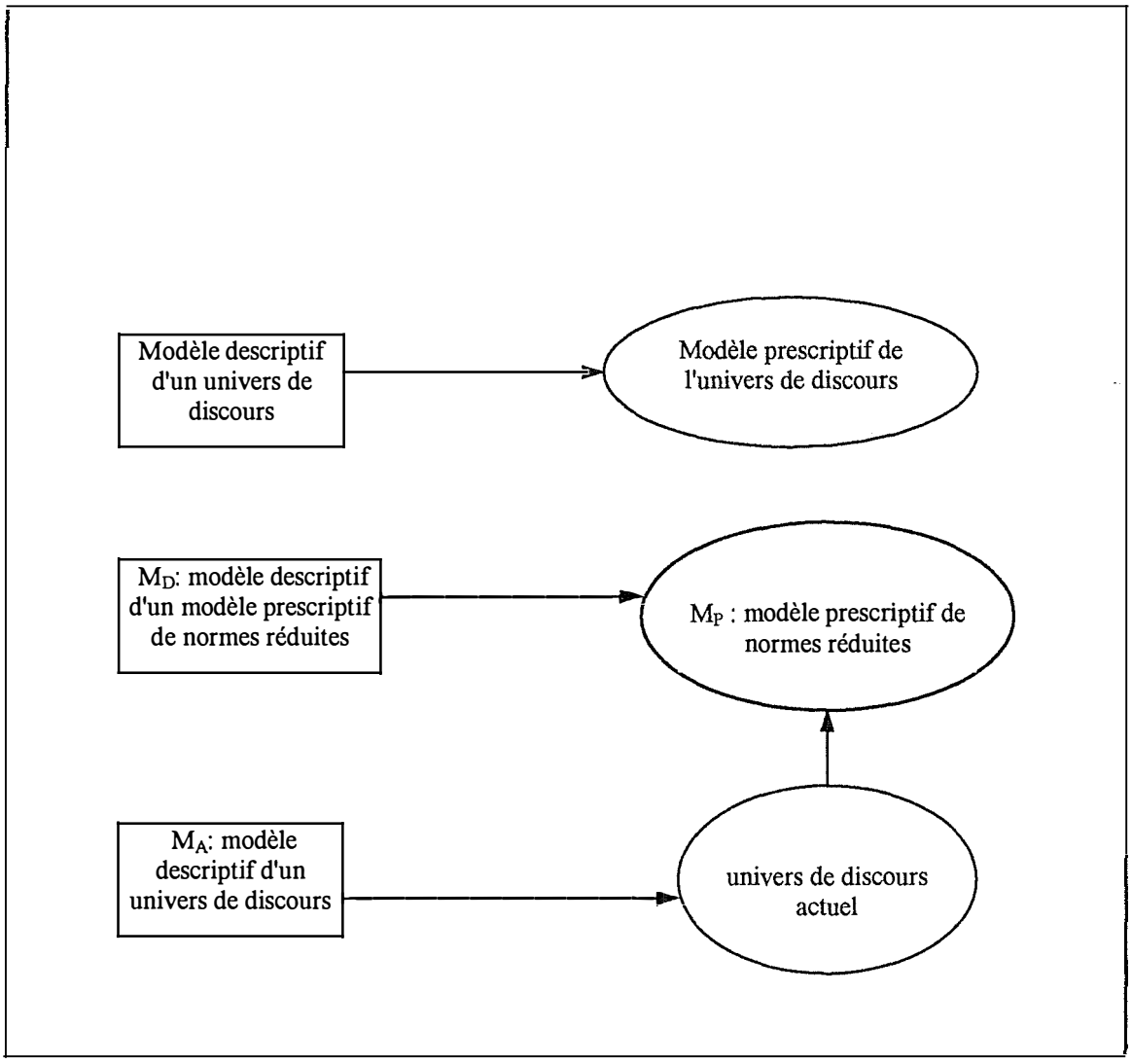

Fig.3. Adaptation du modèle prescriptif à une dimension abordable

Exemple de l'administration d'une bibliothèque qui représente un état courant de la base de données dans son aspect descriptif (M_A)

:" John a emprunté un livre" ;
et comme prescription pour l'univers de discours M_D :

" John doit retourner le livre au bout de trois semaines ";

Si nous voulons représenter les contraintes déontiques dans une base de données avec les autres contraintes, nous obtenons la situation de la fig.4. ;

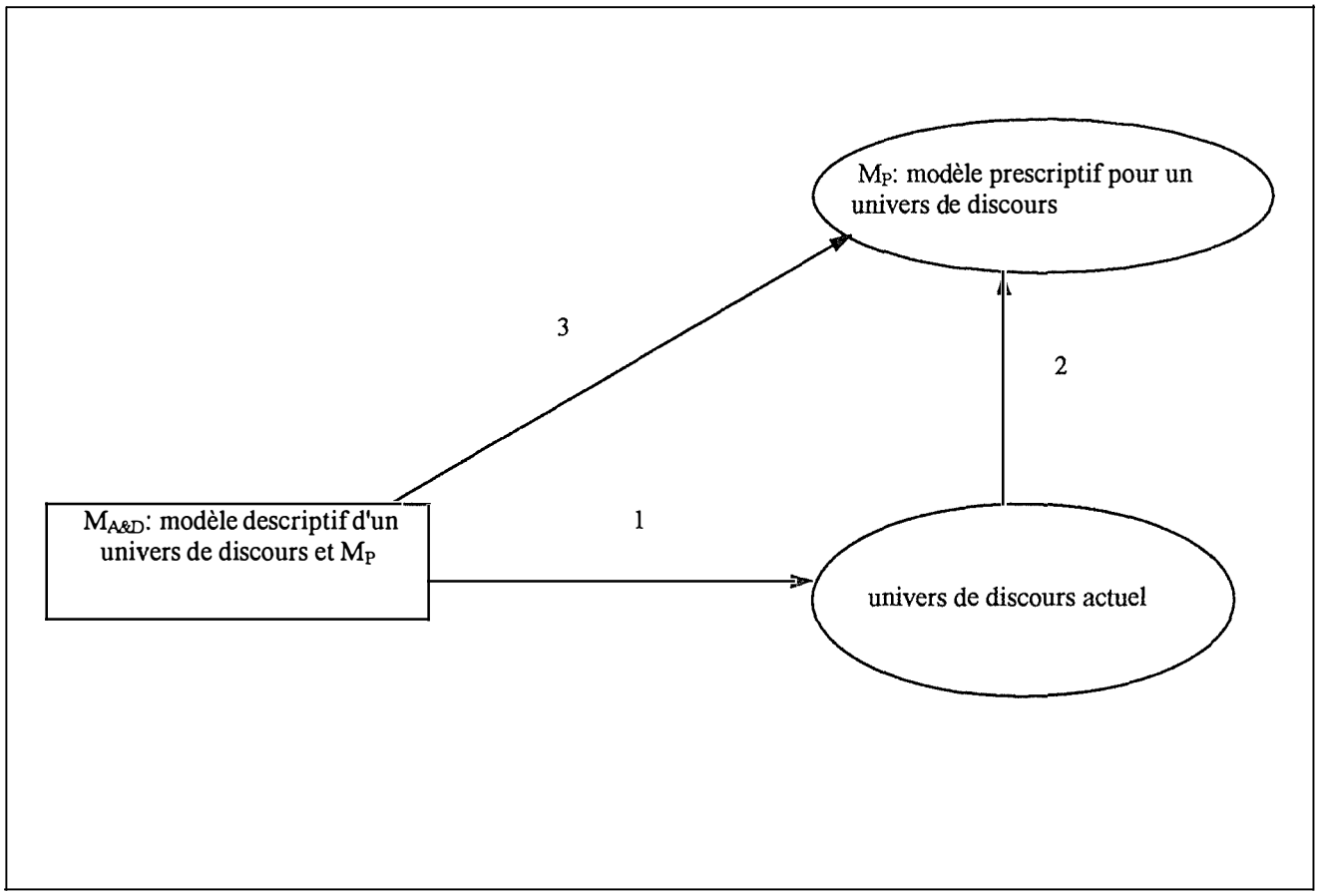

Fig.4. Intégration de modèles descriptif et prescriptif

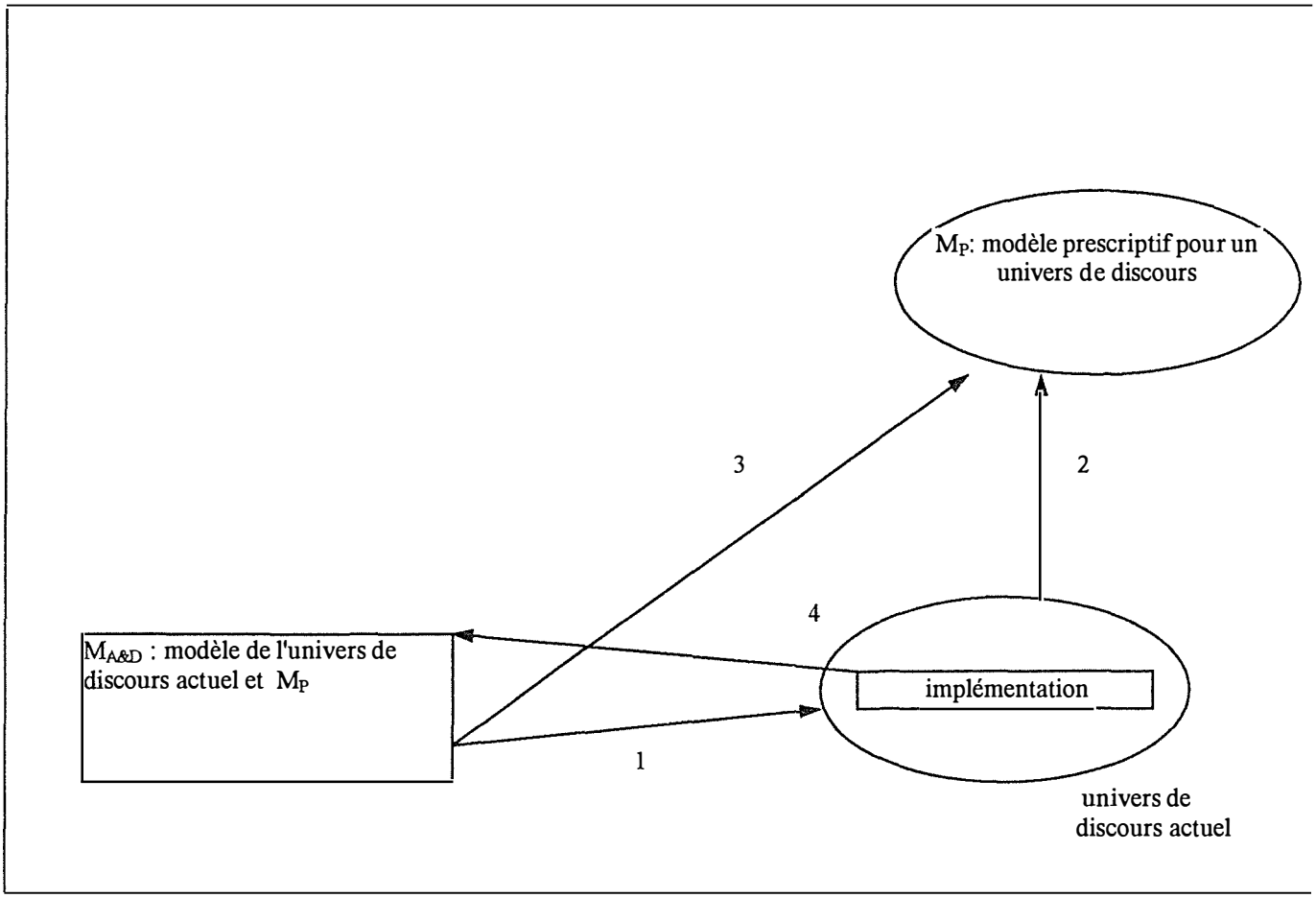

Fig.5 Implémentation et utilisation d'un modèle dans l'univers de discours

Considerons ce qui se passe lorsqu'on insère une implémentation dans l'univers de discours, comme le montre la fig.5.

Nous avons trois cas de figures: implémenter M_A, implémenter M_D, ou implémenter $M_{A\&D}$.

L'implémentation de M_A seule exige que M_A contienne un modèle de l'implémentation car c'est une partie de l'univers de discours. Cette partie du modèle descriptif est appelé habituellement le dictionnaire des données; implémentant M_A nous pouvons relier cette implémentation au reste de l'univers de discours et obtenir un système automatisé; pour ajouter du contrôle, nous devons aussi implémenter une partie de M_D, de façon à avoir un standard de référence pour juger du comportement réel.

Implémenter M_D, seule, est une situation qui arrive souvent en ingénierie; par exemple dans la construction d'avions : le concepteur commence par dresser une liste de desiderata M_P, les modèles de ces desiderata définissent les actions comme des prescriptions pour ces avions.

Considérons maintenant ce qui se passe si les avions, construits suivant la spécification, n'ont pas le comportement attendu. C'est peut être qu'ils n'implémentent pas la spécification correctement (erreur le long de 4.) ou que la spécification n'est pas un bon modèle de la liste des desiderata (erreur le long de 3.), ou enfin que cette dernière liste est irréaliste (M_P est vide).

Chacun de ces cas exige une action corrective différente. (en réalité on opère de façon itérative le long d'une échelle de modèles ou de modèles d'implémentation, jusqu'à obtenir un M_P possible et un M_D souhaitable.

Finalement, si nous implémentons $M_{A\&D}$, entièrement, nous avons l'avantage qu'en plus d'un système d'enregistrement, nous pouvons ajouter une boucle de contrôle (qui sert de feed-back), mais au prix d'une certaine complexité, car il y a plusieurs M_D possibles.

Un exemple illustrant ce propos est le cas d'un programme informatique qui contrôle des missiles; en utilisant un modèle M_A du missile et des capteurs sensoriels dans l'univers de discours, on peut comparer le comportement actuel avec une implémentation d'un modèle abstrait M_D du comportement souhaité; grâce à des moyens de commande on peut ajuster le comportement réel dans l'univers de discours (i.e. du missile).

On fait l'hypothèse qu'un manque de correspondance entre le comportement actuel mesuré par l'implémentation et le comportement désiré, connu du système, est le long de 2.; ce qui veut dire que M_A et M_D à leur tour représentent correctement l'univers de discours et M_P, respectivement, et que M_P est un modèle réaliste pour un comportement du missile.

Remarques

Les exemples données plus haut proviennent aussi bien d'univers de discours concernant un domaine social que technique.

- Une différence importante entre ces deux domaines, est que nous ne pouvons pas, de façon systématique, contrôler le comportement des personnes d'un univers de discours. Les personnes sont sujets à la morale, et les normes qui régissent cet univers ont aussi une dimension morale; cependant le formalisme de la logique des normes n'intègre pas cette dimension; de ce fait des déviations par rapport au comportement normal apparaissent plus souvent dans un univers de discours social que dans un domaine technique, contrôlé par un système de "feed-back"; c'est pourquoi notre modèle de base de données doit être capable de représenter toutes les déviations et permettre des mesures correctives, tendant à réduire ces déviations.

- Une seconde caractéristique d'un univers de discours social est que son état peut être changé, simplement en le représentant comme tel; par exemple de l'argent peut être ajouté à un compte en banque en modifiant l'enregistrement du champ "ba" (k) dans une implémentation. Il y a deux manières de considérer cette mise à jour; dans un cas cette mise à jour reflète un changement

intervenu dans le monde réel; dans le second cas il s'agit "d'assertion performative" à propos de l'univers de discours (cf. Lee[Le88]).
Ce phénomène a lieu fréquemment dans des contextes sociaux; par exemple un maître de séance ouvre une cérémonie en la déclarant ouverte, de même les autorités municipales interdisent l'accès à une place publique en déclarant cette place fermée; dans chaque cas l'agent auteur d'une action établit un état des choses en le déclarant de façon conventionnelle; et seulement une certaine catégorie de personnes est autorisée à le faire.
Ainsi émettre une assertion performative est considéré comme une action (cf Searle and Vanderveken[Se85]).

Remarque à propos des contraintes d'intégrité dans le cadre d'une implémentation

Bien que notre sujet concerne essentiellement la notion de modèle, qui est une base de données abstraite, et non la notion d'implémentation qui représente une base de données concrète sur une machine particulière, nous remarquons les faits suivants:

- Si les contraintes fortes peuvent être utilisées pour contraindre l'implémentation, dans le sens où elle ne peut être dans un état (ou ne peut exécuter une transition) qui viole ces contraintes, les contraintes déontiques, par contre, sont relatives à l'univers de discours et non à l'implémentation;

- Par ailleurs de part notre notion de modèle (base de données abstraite) la base de données ne peut être dans un état erroné, par opposition à son implémentation; néanmoins la base de données doit être capable de représenter un état qui viole une contrainte déontique; C'est pourquoi l'implémentation de la base de données peut être dans plus d'états que le modèle qu'elle implémente, et peut exécuter plus de transitions entre états, qu'il n'est possible dans le modèle; en effet certains de ces états sont considérés comme des états de correction d'erreurs (intuitivement ceci concerne les procédures qui sont déclenchées en réaction à une violation de norme).
De façon similaire certains sous processus sont nécessaires pour simuler les événements dans le modèle; un des ces sous processus doit donc être un processus de correction d'erreur, qui fait correspondre à un état sous idéal (car lieu d'une violation de norme, ce n'est donc pas un monde possible) un état représentant un monde possible; par exemple nous avons besoin d'un événement de réparation, qui restaure l'état erroné avant qu'un événement incorrect ne survienne. Par contraste la base de données doit pouvoir représenter un état qui viole une contrainte d'intégrité, et pour un tel état on doit spécifier une action de correction de violation.

- Au niveau de l'implémentation il y a lieu distinguer entre les contraintes déontiques, celles qui sont imposables et celles qui ne le sont pas; par exemple un système peut imposer la contrainte selon laquelle un client ne peut plus emprunter de l'argent, lorsque son compte en banque est négatif; mais on ne peut pas imposer l'obligation selon laquelle le client doit verser suffisamment d'argent pour que ce compte soit de nouveau positif.

Chapitre 2. Spécification des contraintes d'intégrité

Nous construirons notre langage formel par extension successive, ce sera surtout une extension de la logique propositionnelle dynamique avec les modalités déontiques de permission **P**, d'obligation **O** et d'interdiction **F**.

Satisfaction d'une contrainte

Définition

Pour un langage L et une classe de modèles Λ, une contrainte d'intégrité d'une base de données $M \in \Lambda$, est une formule Φ dans L telle que M satisfait Φ .

2.1. Expression des contraintes statiques

Les contraintes statiques analytiques et empiriques peuvent être exprimées de façon habituelle, au moyen de n'importe quel langage de la logique du premier ordre; nous fixons un langage Lstat, en donnant sa syntaxe et sa sémantique, ses règles de preuves et les axiomes. Plus tard nous étendrons ce langage de façon à pouvoir exprimer les contraintes dynamiques et déontiques.

$\mathbf{L}_{stat}$ = langage des expressions statiques

Lexique

x,y, z variables
a, b, c constantes individuelles
f, g, h constantes fonctionnelles
p, q, r formules
P, Q, R constantes prédicatives
Les formules sont construites comme d'habitude en utilisant :
$\wedge$,$\vee$, $\Rightarrow$, $\forall$, $\exists$ et les symboles de ponctuation (,) , [,]
Deux prédicats spéciaux
- l'un unaire : E prédicat d'existence

- l'autre binaire : = égalité

Une classe distincte de prédicats, appelés prédicats de type sont utilisés pour identifier les différentes entités de l'univers du discours.

SYNTAXE

- Un terme est une variable ou une forme fonctionnelle

- une forme fonctionnelle est la juxtaposition d'une constante fonctionnelle et d'un nombre adéquat de termes. Si f est une constante fonctionnelle dont n est le nombre de places et si $t_1,...,t_n$, sont des termes alors la forme correspondante est habituellement notée $f(t_1,...,t_n)$. si n vaut 0 on écrira f au lieu de f( ).

-Une forme prédicative est la juxtaposition d'une constante prédicative et d'un nombre adéquat de termes. Si P est une forme prédicative dont m est le nombre de places et si $t_1,...,t_m$ sont des termes alors la forme correspondante est habituellement notée $P(t_1,...,t_m)$. Si m vaut 0 on écrira P au lieu de P( )

- Un atome est une forme prédicative ou une égalité, c'est à dire une expression du type (t = s) où t et s sont des termes.

- Le concept de formule est défini de façon récursive au moyen des règles suivantes :
  - un atome est une formule
  - si A est une formule $\neg A$ est une formule
  - si A et B sont des formules ($A \wedge B$) , ($A \vee B$), $A \Rightarrow B$, $A \Leftrightarrow B$ sont des formules.
  - si A est une formule et x une variable $\forall x A$, $\exists x A$ sont des formules.

SEMANTIQUE

Bien que ce soit la syntaxe d'un langage de la logique du premier ordre, sans opérateur modal, nous allons donner une sémantique en termes de structure de Kripke.
On traitera d'univers dans lesquels toutes les entités sont identifiées. De tels univers sont construits à partir de constantes du langage par une construction de Herbrand .

Définition

un symbole de fonction f d'arité $n \geq 1$ est dit *transparent* par rapport à un modèle M de L , si pour toutes constantes $c_1,...,c_n$, il existe une constante c_o tel que M satisfait $f(c_1,...,c_n) = c_0$.

Définition

Soit L un langage

1. *L'univers de Herbrand* U_L de L est l'ensemble de toutes les constantes de L .

2. La *base de Herbrand* B_L de L est l'ensemble de tous les atomes clos de L.

3. Un *modèle de Herbrand* M_L de L est un sous ensemble $M_L \subset B_L$.

Les valeurs de vérités dans M_L sont définies pour les atomes clos par :
M_L satisfait $P(c_1,...,c_n)$ ssi ($P(c_1,...,c_n) \in M_L$ et $E(c_i) \in M_L$ $i=1,...,n$)

4. Un modèle de Herbrand doit également vérifier les conditions de cohérence suivantes :

$\forall x \in U_L \quad (x = x) \in M_L$
$\forall x, y \quad (x = y) \in M_L \Rightarrow (y = x) \in M_L$
$\forall x, y, z \quad (x = y \wedge y = z \Rightarrow x = z)$

5. Pour chaque symbole de prédicat P on a la formule de substitution
$\forall x, y \quad P(x) \in M_L \wedge x_1 = y_1 \wedge \ldots x_n = y_n \Rightarrow P(y) \in M_L$

6. Pour une formule fermée F quelconque, un modèle de Herbrand M_L, la valeur de vérité est définie comme suit:

- Si F est une formule de la forme $\neg G$, $(G \wedge H)$, $(G \vee H)$, $G \Rightarrow H$, $G \Leftrightarrow H$, alors la valeur de vérité de F est donnée par le tableau ci dessous (tab. valeur de vérité).

- Si F est de la forme $(\forall x)$ G alors F est vraie dans M_L ssi pour tout terme clos t G (x/t) est vrai dans I_L.

- Si F est de la forme $(\exists x)$ g alors F est vrai dans M_L si il existe un terme clos t tel que G(x/t) est vrai dans M_L.

7. M_L est un *modèle de Herbrand transparent* de L, si tout symbole de fonction de L est transparent par rapport à M_L.

Tableau : valeur de vérité d'une formule fermée.

G	H	$\neg$ G	G$\wedge$H	G $\vee$ H	G $\Rightarrow$H	G $\Leftrightarrow$H
V	V	F	V	V	V	V
V	F	F	F	V	F	F
F	V	V	F	V	V	F
F	F	V	F	F	V	V

Définition

Une *structure de Herbrand-Kripke* K_L d'un langage L est une collection de modèles de Herbrand transparents de L, qui sont appelés mondes ou états de K_L .

Remarque

On choisit une telle structure de Kripke car on s'intéresse seulement à l'ensemble des mondes possibles et non à une quelconque relation d'accessibilité entre ces mondes ; plus tard nous donnerons la sémantique des actions en termes de transition entre mondes possibles .
Une structure de Herbrand-Kripke peut-être vue comme la collection de tous les états possibles .
Cette collection est l'espace d'états à travers lequel évolue la base de données durant son existence .
Les structures de Herbrand-Kripke sont une formalisation de notre conception de modèle, dans la prise en compte aussi bien de l'aspect descriptif que de l'aspect prescriptif pour un certain univers du discours .

Définition MEYER [Meyer88]

Une *théorie* de M structure de Herbrand-Kripke est un ensemble de formules fermées qui sont vraies dans M.

Remarque

Une théorie est élaborée pour rendre compte d'une certaine réalité plus ou moins concrète(ou de ce que d'aucuns pensent être une réalité). Dans ce cas la notion de vérité préexiste à la théorie. Le but de la théorie est alors de déduire, à partir d'un ensemble d'énoncés "vrais", pris comme des axiomes, tous les énoncés vrais concernant la réalité et rien que ceux-là.
Une théorie du premier ordre est une particularisation du calcul des prédicats à un domaine déterminé.

Nous distinguons un certain nombre de formules qui sont vraies dans nos modèles.

Proposition

Etant donné un langage L et un modèle de Herbrand-Kripke qui représente un certain univers de discours, une théorie d'une base de données, comprend:

1. FOL : l'ensemble de tous les axiomes et théorèmes de la logique des prédicats du premier ordre.

2. HB : l'ensemble des formules énoncées ci-dessous (*)

3. D un ensemble de formules, appelé le domaine de la théorie;

Si FOL et HB sont partagées par tous les modèles de tout univers de discours, D énonce des vérités à propos d'un modèle particulier d'un univers de discours bien précis; les formules de D sont appelés des contraintes d'intégrité.

(*) HB comprend les formules suivantes :

1. Dans toute formule existence d'un domaine fermé
$\forall x \quad (x = c_1 \vee x = c_2 \vee \ldots)$ où toutes et seulement les constantes de L apparaissent parmi les c_i

2. Unicité des noms : $\neg\neg(c_1=c_2)$ pour toutes les constantes c_i et c_j $i \neq j$

3. $-\forall x \quad (x = x)$
$-\forall x, y \quad (x = y \Rightarrow y = x)$
$- x, y, z \quad (x = y \wedge y = z \Rightarrow x = z)$

4. -Pour chaque prédicat P et $\forall x, y$

$P(x) \wedge x_1=y_1 \wedge \ldots x_n=y_n \Rightarrow P(y)$
où $x=x_1, \ldots,x_n$
$y=y_1,\ldots,y_n$

5. Pour chaque prédicat autre que E et les prédicats de type
$\forall$ x $P(x_1,\ldots,x_n) \Rightarrow E(x_1) \wedge \ldots E(x_n)$
E prédicat d'existence
E dénote l'ensemble des objets existants dans un monde .

Preuve

FOL est inclus dans toute théorie, et les autres formules sont vérifiées par notre construction de Herbrand et la définition de la valeur de vérité d'une formule.

Commentaires

1. L'existence d'un domaine fermé signifie que nous choisissons nos objets dans l'univers de Herbrand de L, de sorte que dans les t-uples de la base de données nous trouverons toutes les constantes de L.

2. L'unicité de noms implique que différents noms concernent différents objets;
l'égalité est une relation d'équivalence
Par la formule de substitution, l'égalité est une congruence.

Pour montrer qu'une théorie de base de données est satisfaisable, on construit un univers de Herbrand à partir des constantes du langage et on définit les extensions de E et des autres prédicats dans les différents mondes de la structure.
Notons que si une théorie admet un modèle de Herbrand, alors elle possède un modèle (donc satisfaisable).

Définition

De façon similaire que dans le cas statique, nous définirons trois types de théorie.

1. Si le langage L est L_{STAT} et

$T = FOL \cup HB \cup D$ avec

D = Stat

où Stat est un ensemble de formules dans L_{STAT}, alors T est appelé une *théorie statique* de base de données; D est le domaine statique de la théorie; les formules de Stat sont appelées des contraintes d'intégrité.

2. Si Lest L_{DYn} (le langage que nous définirons au paragraphe suivant) et

T = FOL∪HB ∪DL ∪ D avec

DL = ensemble d'axiomes de la logique dynamique (voir paragraphe suivant)

D = Stat ∪ Dyn

où Dyn est un ensemble non vide de formules de L_{Dyn}, alors T est appelé *une théorie dynamique* de base de données et D est le domaine dynamique de la théorie. Les formules de Dyn sont appelés les contraintes dynamiques.

3. Si L est L_{DEON} (définie plus loin pour les contraintes déontiques) et

T = FOL ∪HB∪ DL ∪D avec

D = Stat ∪Dyn ∪Déon,

où Deon est un ensemble non vide de formules dans L_{DEON}, alors T est appelé une *théorie déontique* de base de données, et D est un domaine de théorie. Les formules de DEON sont appelées les contraintes déontiques.

2.2. Expression des contraintes dynamiques

Nous utiliserons une variante de la logique dynamique pour exprimer aussi bien les contraintes dynamiques que déontiques.
Nous définirons d'abord un langage pour les actions.

Les actions

Soit A un ensemble dénombrable d'actions atomiques, le langage L_{ACT} des actions est défini par la syntaxe suivante

α:: = a / $\alpha_1 \cup \alpha_2$ / $\alpha_1 \& \alpha_2$ / – & / any/ fail
a$\in$ A ensemble d'actions atomiques

$\alpha_1 \vee \alpha_2$ représente le choix non déterministe de α_1 et α_2; il est interprété comme l'ensemble union des interprétations de α_1 et α_2

$\alpha_1 \& \alpha_2$: est l'exécution en parallèle (simultanée) de α_1 et α_2 ; il est interprété comme l'intersection des interprétations de α_1 et α_2

-α : est la non exécution de α; il est interprété comme l'ensemble des états qui sont accessibles par la collection d'actions qui ne contiennent pas α ; (plus précisément on considère tous les sous-ensembles de A qui ne contiennent pas α); on formalise ainsi la non réalisation d'une action

any : est une action quelconque non spécifiée; est interprété comme l'ensemble de tous les mondes possibles accessibles en un pas.

fail : est action vide (ne rien faire)

Définition

Les actions atomiques sont interprétées comme des transformations d'états :
étant donné un état w dans une structure de Herbrand Kripke, on associe à chaque action a$\in$ A un ensemble $W_{a,w}$ de mondes accessibles (c'est-à-dire des modèles de Herbrand) dans K_L ,qui est l'ensemble des états auxquels on peut accéder par l'exécution d'actions arbitraires, qui contiennent a comme élément ; plus précisément on considère tous les sous-ensemble de A qui contiennent a, ces sous ensemble sont des paquets d'actions atomiques qui sont exécutes simultanément.

Sémantique formelle des actions

A ensemble d'actions atomiques

On associe à chaque action a une fonction $\rho(a)$ décrivant la comportement de a:

$\rho(a) : K_L \rightarrow K_L$ K_L une structure de Herbrand Kripke
c'est une collection de modèles de Herbrand transparents de L

Un monde ou état de K_L est donc un modèle de Herbrand transparent

On note $\rho(A) = \{ \rho(a) / a \in A \}$ l'ensemble des fonctions associées à A

Soit $P^+ (\rho(A))$ la collection finie d'ensemble non vides de $\rho(A)$

La sémantique formelle de L_{ACT} est donnée par une fonction sémantique :

$[.] \quad L_{ACT} \rightarrow (K_L \rightarrow P (K_L))$

Nous définissons d'abord une fonction auxiliaire

$[.]' : L_{ACT} \rightarrow P (P^+ (\rho(A)))$

Définition

1. $[a]' = \{ S \subset \rho(A) / \rho(a) \in S \}$
2. $[\alpha_1 \vee \alpha_2]' = [\alpha_1]' \cup [\alpha_2]'$
3. $[\alpha_1 \& \alpha_2]' = [\alpha_1]' \cap [\alpha_2]'$
4. $[-\alpha]' = P^+ (\rho(A)) \setminus [\alpha]'$
5. $[Fail]' = \varnothing$

6. $[\text{ any}]' = P^{+}(\rho(A))$

Remarques:

D'après 1. la sémantique de a est définie par tous les tas d'actions atomiques, exécutés simultanément, qui contiennent le comportement de a.

De même 4. Implique que la sémantique de -a est déterminée par tous les tas d'actions atomiques, exécutés simultanément, qui ne contiennent pas le comportement de a .

Définition 2

On considère la relation définie comme suit

$R : \quad P^{+}(\rho(A)) \rightarrow (\ K_L \rightarrow P(K_L) \ )$

Soit $S = \{ \rho(a_1), ..., \rho(a_n) \} \subseteq \rho(A) \quad n \geq 1$

alors $R(S) = \{ \rho(a_1) o ... o \rho(a_n)$ si $\rho(a_1), ..., \rho(a_n)$ sont compatibles pour l'argument ω

$R(S) = \varnothing$ sinon

où $f_1, ..., f_n : K_L \rightarrow K_L$ sont compatibles pour ω si

$(f_1 o ... o f_n) (\omega) = (f_{i1} o ... o f_{in})(\omega)$ pour toute permutation $(i_1,...i_n)$ de $(1...n)$

R est étendue au domaine $P(P^{+}(\rho(A)))$ de manière suivante :

si $T \subseteq P^{+}(\rho(A))$ alors $R(T) (\omega) = \cup \ R(s) (\omega) \quad s \in T$

Finalement on arrive à la définition suivante

$[\alpha] = \lambda\omega. R([\alpha]')(\omega)$

$= \{ R([\alpha]')(\omega) / \omega \in K_L \}$

Transactions

L_{TRANS} = langage des transactions

Les actions sont instantanées, en ce sens qu'elles ne nécessitent pas d'état intermédiaire; par contre si un évènement a besoin de plus d'un état pour se produire, il sera représenté par une transaction. L'exécution d'une action est appelé un *pas* ; de fait les transaction peuvent durer plus d'un pas . Le langage des transactions est :

$\beta :: = \alpha / \beta ; \beta /$ Clock avec $\alpha \in L_{ACT}$

Définition

1. Etant donné un état w dans une structure de Herbrand Kripke K_L et
   $W_{\beta 1;w}$ l'ensemble des états qui peuvent être atteints à partir de w par l'exécution de β_1
   alors $\beta_1;\beta_2$ est l'union des ensembles $W_{\beta 2,w'}$ pour $w' \in W_{\beta 1;w}$

2. La transaction Clock est interprétée comme suit

Clock = $any^{(n-1)}$; inc(t)

où n est le nombre de pas nécessaires pour passer une unité de temps

$any^{(n-1)}$ = any; any; ...; any (n-1)fois

inc(t) est une action atomique qui augmente d'une unité la valeur de la variable spéciale t (t est du type de l'unité de temps choisie).

Remarque

Clock est une transaction de durée l'unité de temps choisie; intuitivement un tick de Clock correspond à l'écoulement d'une unité de temps , ainsi si l'unité de temps est la minute alors un tick de Clock est l'écoulement d'une minute, celui d'une journée si l'unité est le jour et ainsi de suite.

Il est à noter que durant un tick de Clock, any est exécutée une ou plusieurs fois.

On utilise les abréviations suivantes :

$\beta^n = \beta;...;\beta$ (n fois)

$\alpha_{(n)} = Clock^n; \alpha$ (on note $\alpha_{(0)} = \alpha$)

$\alpha_{(\leq d)} = \alpha_{(0)} \cup ... \cup \alpha_{(d)}$

$- \alpha_{(n)} = Clock^n; -\alpha$

$\alpha_{(>d)} = -\alpha_{(0)} \& ... \& -\alpha_{(d)}$

Exemple

Dans le cas de la spécification des règles d'emprunt dans une bibliothèque :
retourne $_{(\leq 3\ s)}$: est l'action de remettre un livre emprunté au plus tard dans trois semaines à la bibliothèque.

Contraintes dynamiques

Le langage $\mathbf{L_{dyn}}$ des expressions dynamiques :

$\Phi :: = \phi / \Phi_1 \vee \Phi_2 / \neg\Phi / \Phi_1 \wedge \Phi_2 / \Phi_1 \Leftrightarrow \Phi_2 / [\beta]\Phi$ / DONE : α
$\alpha \in L_{ACT}$ $\phi \in L_{stat}$

Une contrainte dynamique (formule de L_{dyn}) est ainsi soit une contrainte statique ou une combinaison logique de contrainte dynamique, ou a la forme $[\beta]\Phi$ ou DONE: α .
DONE: α exprime que l'action α a été réalisée; ce prédicat est vrai dans tous les mondes où α vient juste être exécutée.
Nous utiliserons aussi $<\beta>\Phi$ comme abréviation de $\neg[\beta]\neg\Phi$

Définition

Soit $w \in K_L$ une structure de Herbrand Kripke
w satisfait $[\beta]\ \Phi$ si et seulement si pour tout $w' \in K_L$,
tel que l'exécution de β dans w produit w ', alors
w' satisfait Φ.

Par construction de notre classe de modèles L_{DYN} et de la définition de la valeur de vérité, un certain nombre de formules sont vraies dans tous les modèles de L_{DYN} .

Proposition

Etant donné un modèle $M \in L_{DYN}$

Une théorie T de M a la structure suivante $T = FOL \cup HB \cup DL \cup D$

où FOL et HB comme précédemment définis

D est le domaine de la théorie

DL est l'ensemble d'axiomes de la logique dynamique suivant :

DL_1 $[\alpha]\ (\Phi_1 \Rightarrow \Phi_2) \Rightarrow\ (\ [\alpha]\Phi_1 \Rightarrow [\alpha]\Phi_2\ )$

DL_2 $[\alpha_1 \cup \alpha_2\ ]\ \Phi \Leftrightarrow [\alpha_1]\ \Phi \cap [\alpha_2]\ \Phi$

DL_3 $[\alpha_1 \& \alpha_2]\ \Phi \Leftrightarrow [\alpha_1]\ (DONE : \alpha_2 \Rightarrow \Phi)$

DL_4 $[-(\alpha_1 \cup \alpha_2)]\Phi \Leftrightarrow [-\alpha_1 \& -\alpha_2]\Phi$

DL_5 $[-(\alpha_1 \& \alpha_2)]\ \Phi \Leftrightarrow [-\alpha_1 \cup -\alpha_2]\Phi$

DL_6 $[fail]\ \Phi \Leftrightarrow true$

DL_7 $[\beta_1;\beta_2]\ \Phi \Leftrightarrow [\beta_1]\ ([\beta_2]\ \Phi)$

DL_8 $DONE:(\ \alpha_1 \cup \alpha_2) \Leftrightarrow DONE:\ \alpha_1 \vee DONE:\alpha_2$

DL_9 $DONE :(\ \alpha_1 \&\ \alpha_2) \Leftrightarrow DONE:\ \alpha_1 \wedge DONE:\ \alpha_2$

DL_{10} : $DONE: -\alpha \Leftrightarrow \neg\ DONE : \alpha$

DL_{11} : DONE : any $\Leftrightarrow$ true

DL_{12} DONE : fail $\Leftrightarrow$ false

DL_{13} $[\alpha]$ DONE : α

DL_{14} $[\alpha_1]\ \Phi \Rightarrow [\alpha_2]$ (DONE : $\alpha_1 \Rightarrow \Phi$)

DL_{15} $t = n \Rightarrow$ [Clock] $t = n+1$

Preuve

Intuitivement, tout ceci découle de la sémantique définie plus haut
Les preuves se trouvent dans Meyer[Meye88].

Remarques

1. DL_2 un choix non déterministe entre α_1 et α_2 conduit à Φ si et seulement si tous deux α_1 et α_2 mènent dans un monde où Φ est vrai.

2. DL_3 est vrai car les actions sont instantanées; si les actions prenaient plus d'un pas, alors DL3 ne serait vrai que si les deux actions α_1 et α_2 avaient la même durée.

3. DL_4 est vrai car Φ est vrai après la réalisation de fail, car fail n'a pas d'état successeur.

4. DL_{14}. Si α_1 mène nécessairement à Φ alors si α_2 a été réalisé, Φ est vrai si α_1 a été réalisé aussi;

5. DL_{15}. t est la variable spéciale qui est augmentée d'une unité à chaque tick de clock.

6. Pour que DL soit vrai dans notre modèle, on doit exiger les actions atomiques soient uniques, dans le sens où chaque action atomique a une" signature" unique dans le monde résultant immédiatement de son exécution ; ceci est fait au moyen du prédicat DONE :α qui est vrai dans les mondes ,qui résultent de l'exécution de α et est faux dans les autres; ainsi plusieurs occurrences de la "même" action doivent être identifiés de façon à les distinguer.

Les actions primitives seront habituellement paramétrées de sorte que des exécutions avec des paramètres actuels différents produiront des résultats différents sur un état.

Exemple :

$\forall^E$ e,s, n Salaire(e,s) $\Rightarrow$ [change-salaire(e,n)] Salaire(e, s + n)

e est un employé, s son salaire Salaire (e, s) : e a pour salaire s
change-salaire est une action qui ajoute n au salaire de l'employé;
L'effet d'une action n'est définie que pour des objets existants ($\forall^E$)
E désigne l'ensemble des objets existants dans un monde donné.

2.3 Expression des contraintes déontiques

$\mathbf{L}_{DEON}$ = expression des contraintes déontiques

introduction de prédicats de violation $V_i : \alpha \quad i \in N, \ \alpha \in L_{ACT}$

Les concepts déontiques de permission et d'obligation peuvent être réduits au concept d'interdiction, qui à son tour peut-être réduit au concept d'exécution d'une action entraînant la violation d'une règle. Plutôt que d'exprimer explicitement les règles, nous indiquons quand ces règles sont violées en utilisant des prédicats de violation V_i: α ; un pour chaque raison pour laquelle l'exécution de l'action α est interdite.
Il est important de noter qu'il existe cependant une différence pratique entre une interdiction et une obligation; la violation d'un interdit peut-être observée immédiatement: si il est interdit de voler un livre à la bibliothèque, la violation d'une telle règle peut-être constatée aussitôt le vol commis; de l'autre coté, si l'on est obligé de retourner un livre emprunté à la bibliothèque, la violation d'une telle obligation ne peut-être établie si aucun délai n'est fixé pour la réalisation de cet action; c'est pourquoi notre propos concernera des obligations qui doivent être remplies endéans un intervalle spécifique de temps.
Pour chaque prédicat de violation nous définirons une action corrective qui permettra de sortir d'un état où ce prédicat est vrai.

Définition

On utilisera les abréviations suivantes :

$\mathbf{F}(\alpha) \Leftrightarrow [\alpha] V_i: \alpha \quad i \in N$ " il est interdit que α "

$\mathbf{P}(\alpha) \Leftrightarrow \neg \mathbf{F}(\alpha)$ " il est permis que α "

$\mathbf{O}(\alpha) \Leftrightarrow \mathbf{F}(-\alpha)$ " il est obligatoire que α "

On considère qu'un évènement est interdit s'il mène nécessairement dans un état où il y a violation d'une action.
La réduction des concepts déontiques fut d'abord proposée par Anderson [And 58] et formalisée dans le contexte de la logique dynamique par MEYER [Meyer88].
Une action est permise si et seulement si elle n'est pas interdite, ce qui est équivaut à dire
$\mathbf{P}(\alpha) \Leftrightarrow < \alpha > \neg$ Vi: α c'est à dire qu'il existe au moins une façon d'exécuter α qui n'entraîne pas de violation
Enfin une action est obligatoire si et seulement si ne pas l'exécuter est interdit, ce qui d'après la réduction faite à propos de l'interdiction s'exprime par :
$\mathbf{O}(\alpha) \Leftrightarrow [-\alpha]$ Vi:$(-\alpha)$

2.4. Quelques exemples d'application

Exemple 1 : GESTION D'UN CLUB DE TENNIS

On fait l'hypothèse que la base de données concerne un club unique que nous nommerons par c; D'autre part sur un terrain un match est la confrontation entre deux joueurs j_1 et j_2, par conséquent les rencontres en double sont considérées comme des parties entre deux équipes , représentées chacune par un joueur.

Pour la bonne marche du club, l'administration fixe un certain nombre de règles:

- Pour pouvoir jouer sur un terrain, il faut être membre du club ou être invité par un membre du club ; l'administration loue à cet effet des terrains aux joueurs contre une somme de 100 BEF pour un membre du club et 200 BEF pour un invité .

- Cependant les joueurs doivent être en règle vis-à-vis des dispositions du club; ainsi, pour pouvoir louer un terrain, on ne doit pas avoir d'amendes ou de locations antérieures, non encore réglées .

- Les week-end (vendredi, samedi, dimanche) compte tenu de l'affluence , il est obligatoire de réserver à l'avance un terrain pour pouvoir jouer; cette réservation n'est possible que si l'on est membre du club.

- Pour un bon usage des courts de tennis, il y a obligation d'entretenir un terrain en terre battue après son utilisation, faute de quoi on s'expose à une amende; d'autre part il est interdit de jeter sa raquette par terre au cours d'un match, sous peine d'une amende de 500 BEF et d'une sanction de 15 jours. Une personne suspendue pour la troisième fois ne peut plus se représenter au club.

Prédicats de type

Club(c) : c est un club;

Terrain(t) : t est un court de tennis;

Amende(a) : a est une amende;

Joueur(j) : j est un joueur;

Date(d) : d est une date (jour, heure, minute);

Autres prédicats

Terre-battue(t) : t est un terrain en terre battue;

Synthétique(t) : t est un terrain en synthétique;

Membre(j) : j est membre du club c;

Invité(j) : j est invité du club c;

Sur-terrain(j_1, j_2, t): j_1 et j_2 sont sur un même court de tennis;

Location(l, j, t, f): l est la location du terrain t par le joueur j et dont le montant est fixé à f;

Est-suspendu(j) : le joueur est suspendu;

Suspension(s, j) : s est une suspension de 15 jours infligée à j;

Est-loué-par(t, j, d) : le terrain t est loué par le joueur j pour la date d;

Est-réservé-par(t, j, d) : le terrain t est réservé par le joueur j pour la date d;

Week-end(d) : d est un jour de week-end;

Actions

louer(j, t, d) : j loue le terrain t pour la date d;
réserver(j, t, d) : j réserve le terrain t pour la date d;
payer-amende(j, a) : j paie l'amende a;
payer-location(j, l) : j paie la location l;
entretenir(j, t, d) : j assure l'entretien du terrain t utilisé pendant d;
jeter-raquette(j, t, d) : j jette sa raquette par terre sur t pendant d;
suspendre(c, j) : le club c inflige une suspension à j.

CONTRAINTES STATIQUES

(S_0) $\forall^E$ t terrain(t) $\Rightarrow$ Terre-battue(t) $\vee$ Synthétique(t);

% les terrains du club c sont soit en terre battue, soit en synthétique;

(S_1) $\forall^E$ j Joueur(j) $\Leftrightarrow$ (Membre(j) $\vee$ invité(j))
$\wedge \neg$ ($\exists$ s_1, s_2, s_3, s_4 Suspension(s_1, j)$\wedge$ Suspension(s_2, j)
$\wedge$ Suspension(s_3, j)$\wedge$ Suspension(s_4, j))

% Dans la base de données on enregistre comme joueurs, des membres affiliés au club ou des % invités, avec la condition qu'un joueur doit avoir moins de trois suspensions.

(S_2) $\forall^E$ j_1, j_2 Sur-terrain(j_1, j_2, t, d) $\Rightarrow$ Membre(j_1) $\vee$ Membre(j_2)

% Sur un court de tennis, au cours d'une rencontre, il y a toujours au moins un membre du club;

(S_3) Invité(j) $\Rightarrow$ Location(l, j, t, 200 BEF)
Membre(j) $\Rightarrow$ Location(l, j, t, 100 BEF)

% La location d'un court de tennis pour un invité du club coûte 200 BEF, et 100 BEF pour un % membre du club.

(S_4) $\neg$ ($\exists$ j, j', t, d, j $\neq$j' Est-réservé-par(t, j, d)$\wedge$ Est-réservé-par(t, j', d))

% *Deux joueurs ne peuvent pas réserver un même terrain pour une même période.*

(S5) $\forall^E$ j ,s Est-suspendu(j) $\Rightarrow$ Suspension(s, j)
% *Un joueur suspendu, l'est pour 15jours ;*

CONTRAINTES DYNAMIQUES

(D_0) $\forall^E$ j_1, j_2, t, d $j_1 \neq j_2$ Est-réservé-par(t, j_1, d) $\Rightarrow$ [louer(j_2, t, d)] false

% *Il n'est pas possible de louer un terrain déjà réservé par un autre joueur.*

(D_1) $\forall^E$ j, t, d $\neg$ Est-réservé-par(t, j, d) $\Rightarrow$ [(-réserver(j, t, d))] $\neg$ Est-réservé-par(t, j, d)

% Un terrain qui n'a pas été réservé reste dans cet état, tant que l'on n'exécute pas l'action de reservation ;

(D_2) $\forall^E$ j, t, d [réserver(j, t, d)] Est-réservé-par(t, j, d)

% Lorsqu'on exécute l'action de resevation d'un terrain, ce dernier devient réserver;

(D_3) $\forall^E$ j, t, d $\neg$Est-loué-par(t, j, d) $\Rightarrow$ [(-louer(j, t, d))] $\neg$Est-loué-par(t, j, d);

% Un terrain qui n'a pas été loué, reste non loué, tant qu'on n'execute pas l'action de location;

(D_4) $\forall^E$ j, t, d [louer(j, t, d)] Est-loué-par(t, j, d);

% Lorsqu'on loue un terrain , celui-ci devient loué;

(D_5) $\forall^E$ j [suspendre(c, j)] Est-suspendu(j)

% Lorsqu'on exécute l'action suspendre un joueur, ce dernier est suspendu;

(D_6) $\forall^E$ j $\neg$Est-suspendu(j) $\Rightarrow$ [(-suspendre(c,j))] $\neg$ Est-suspendu(j);

% Si on n'est pas suspendu, on reste ainsi tant que l'action suspendre n'a pas été exécutée;

CONTRAINTES DEONTIQUES

(C_0) $\forall^E$ j_1, d, t **P**(jouer(j_1, t, d)) $\Leftrightarrow$ (Membre(j_1) $\wedge$ Est-loué-par(t, j_1, d))
$\vee$ ($\exists$ j_2 Sur-terrain(j_1, j_2, t) $\wedge$ Est-loué-par(t, j_2, d));

% Pour pouvoir jouer sur un terrain il faudrait être membre du club ou être invité, c'est à dire
% être sur un terrain en compagnie d'un membre du club; dans chaque cas il faut louer d'abord
% le terrain.

(C_1) $\forall^E$ j_1, d, t **P** (louer(j_1, t, d)) $\Leftrightarrow$ $\neg$ ($\exists$ j_2 $j_1 \neq j_2$ Est-réservé-par(t, j_2, d))
$\wedge \neg$ ($\exists$ a_1 V: payer-amende(j_1, a_1))
$\wedge \neg$ ($\exists$ l_1 V: payer-location(j_1, l_1)) ;

% On ne peut pas louer de nouveau un terrain si on s'est pas acquitté d'amendes antérieures ou de frais de locations antérieures.

C_2 $\forall^E$j, t, d **P**(réserver(j, t, d)) $\Leftrightarrow$ Membre(j)

% Pour pouvoir réserver un terrain il faut être membre du club.

(C_3) $\neg$Est-réservé-par(t, j, d) $\wedge$ d$\in$ Week-end(d) $\Rightarrow$ **F** (louer(j, t, d)) ;

% Un joueur qui souhaite disposer d'un terrain le week-end est obligé de le réserver à l'avance;

(C_4) Terre-battue(t) $\Rightarrow$ [jouer(j, t, d)] **O** (entretenir(j, t, d));

% Après avoir utilisé un terrain de terre battue il y a obligation de l'entretenir, faute de quoi on % s'expose à une amende.

(C_5) V: entretenir (j, t, d) $\Rightarrow$ **O** (payer-amende(j, a));

% Compte tenu du fait que l'on n'a pas entretenu un terrain de terre battue, après usage, on est
% dans l'obligation de payer une amende. De plus si on ne paie pas cette amende , d'après D1 on
% n'est pas autorisé à louer à l'avenir un terrain.

(C_5) [louer(j, t, d)] **O** (payer-location(j, l));
% On est obligé de payer une location de terrain;

Exemple 2 : GESTION DE CONTRAT DE BAIL

Les règles suivantes concernent des clauses d'un contrat de location entre un propriétaire et son locataire.

. L'appartement est loué à un seul locataire à la fois, cependant un locataire principal a la possibilité de sous-louer son appartement.

. Le locataire est obligé de payer son loyer au début du mois, faute de quoi au bout de deux retards il risque une amende, équivalente à 10 % du loyer mensuel.

. En cas de rupture de contrat de bail de la part du locataire, celui-ci est obligé de payer trois mensualités à titre d'indemnisations au propriétaire. Si cette rupture est le fait du propriétaire, ce dernier doit verser six mensualités au locataire; toutefois si ce dernier n'a plus payé son loyer depuis trois mois, cette rupture ne donne lieu à aucune indemnisation.

Prédicats de type

Personne(p) : p est une personne
Logement(l) : l est un logement
Amende(a) : a est une amende
Mois(m) : m est l'un des douze mois de l'année

Autres prédicats

Locataire-principal(p) : p est locataire principal d'un logement;

Sous-locataire(p) : p est sous-locataire d'un logement

Est-loué-par(l, p) : l est loué par p

Est-sous-loué-par(l, p) : l est sous-loué par p

A-payé-loyer(p, l, m) : p a payé le loyer du mois m pour le logement l

V: prédicat de violation

Actions

louer(p, p', l, m) : p loue le logement l à p' pour le mois m ;

sous-louer(p, p', l, m) : p sous loue son logement l à p' ;

rompre-contrat(p, p', l) : l'une des parties p ou p' rompt le contrat de bail du logement l avec l'autre partie concernée ;on considère que le premier argument de l'action rompre-contrat est celui qui effectue l'action;

payer-loyer(p, p', l, m) : p paie le loyer du mois m pour le logement l;

payer-loyer(p, p',l, m) $_{(\leq 7J)}$: p paie le loyer du mois m pour le logement l au plus tard dans sept jours;

payer-amende(p, a, l) $_{(\leq 5J)}$: p paie une amende concernant le logement l au plus tard dans cinq jours;

payer-amende(p, a , l) : p paie une indemnité a concernant le logement l;

CONTRAINTES STATIQUES

$\forall^E$ l, $\neg$ ($\exists$ p, p', p $\neq$ p' Est-loué-par(p, l) $\wedge$ Est-loué-par(p', l));

% Un logement ne peut pas être loué en même temps à deux personnes;

$\forall^E$ l, p Est-loué-par(l, p) $\Rightarrow$ Locataire-principal(p)

$\forall^E$ l, p Est-sous-loué-par(l, p) $\Rightarrow$ Sous-locataire(p)

% Lorsqu'on loue un logement on en devient le locataire
% Lorsqu'on sous-loue un logement on ne devient le sous-locataire;

CONTRAINTES DYNAMIQUES

$\forall^E$ p, p', l , m [louer(p, p', l, m)] Est-loué-par(l, p)

% *Lorsque p loue un logement l à p', l'appartement devient loué par p;*

$\forall^E$ p, p', l, m [sous-louer(p, p', , m)] Est-sous-loué-par(l, p')

$\forall^E$ p, p', l Est-loué-par(l, p) $\Rightarrow$ [(-rompre-contrat(p, p', l))] Est-loué-par(l, p)

% *Un logement loué reste ainsi tant qu'il n'y a pas de rupture de contrat*;

$\forall^E$ p, p', l $\neg$Est-loué-par(l, p) $\Rightarrow$ [rompre-contrat(p, p', l)] false

% *L'action de rompre un contrat est sans effet, lorsque cette rupture concerne un logement que l'on n'a pas loué;*

$\forall^E$ p, p', l [rompre-contrat(p, p', l)]$\neg$ Est-loué-par(l, p)

% *En cas de rupture de contrat, le logement n'est plus loué;*

$\forall^E$ p, p', l, m [payer-loyer(p, p', l, m)] A-payé-loyer(p, l, m)

% *quand on exécute l'action payer-loyer, le loyer est effectivement payé;*

$\forall^E$ p, p', l, m $\neg$A-payé-loyer(p, l, m) $\Rightarrow$ [(-payer-loyer(p, p', l, m))] $\neg$A-payé-loyer(p, l, m)

% Tant qu'on n'a pas exécuté l'action payer-loyer, le loyer n'est pas payé;

CONTRAINTES DEONTIQUES

$\forall^E$ p, p', l locataire-principal(p) $\Rightarrow$ **P**(sous-louer(p, p', l, m))

% *Un locataire principal peut sous-louer son appartement;*

$\forall^E$ p, p', l, m [louer(p, p', l, m)] [Clock $^{(30)}$] **O** (payer-loyer(p, p', l, m) $_{(\leq 7J)}$)

$\forall^E$ p, p', l, m **O** (payer-loyer(p, p', l, m) $_{(\leq 7J)}$) $\Rightarrow$ ([Clock $^{(30)}$]Est-loué-par(p, l)
$\Rightarrow$ **O** (payer-loyer(p, p', l, m+1) $_{(\leq 7J)}$))

% *Le loyer doit être payé au début du mois, une semaine au plus tard; cette obligation reste valable tant qu'on est locataire;*

$\forall^E$ p, l ($\exists$ m_1, m_2, $m_1 \neq m_2$

V: payer-loyer(p, p', l, m_1) $_{(\leq 7J)}$ $\wedge$V : payer-loyer(p, p', l, m_2)$_{(\leq 7J)}$)

$\Rightarrow$ **O** (payer-amende(p, a_1))

% *Après deux retards de paiement du loyer, le locataire doit verser une amende;*

$\forall^E$ p, p', l, m [louer(p, p', l, m)] **F**(rompre-contrat(p, p', l))

% *Il est interdit de rompre un contrat de bail;*

$\forall^E$ p, p', l locataire(p) $\wedge$ V: (- rompre-contrat(p, p' l))$\Rightarrow$ **O**(payer-amende(p, a_2))

% *La rupture d'un bail par le locataire oblige ce dernier à payer au propriétaire trois*
% *mensualités à titre d'indemnités;*

$\forall^{E}$ p, p', l propriétaire(p') $\wedge$ V: (- rompre-contrat(p', p, l))$\Rightarrow$ **O**(payer-amende(p', a_2))

% La rupture du bail par le propriétaire oblige celui-ci à verser 6 mensualités au locataire

$\forall^{E}$ p, p', l, m [louer(p, p', l, m)] **F**(endommager-mobilier(p, l))

% Il est interdit au locataire d'endommager le mobilier mis à sa disposition;

$\forall^{E}$ p, l V: (-endommager-mobilier(p, l)) $\Rightarrow$ **O**(payer-amende(p, a_3) $_{(\leq 5J)}$

% Toute dégradation de mobilier donne lieu à une amende, qui doit être réglée, dans les plus
% brefs délais (5 jours au plus tard) ;

$\forall^{E}$ p, p', l propriétaire(p')$\wedge$ ($\exists$ m_1, m_2, m_3, $m_1 \neq m_2$, $m_1 \neq m_3$, $m_2 \neq m_3$

V : payer-loyer(p, p', l, m_1) $\wedge$ V : payer-loyer(p, p', l, m_2) $\wedge$ V : payer-loyer(p, p', l, m_3))

$\Rightarrow$**P**(rompre-contrat(p', p, l))

% Si le locataire accuse trois mois de loyer impayés, le propriétaire peut rompre le bail sans
% aucune indemnité;

CONCLUSION

Utilité de cette démarche

Nous avons mis en évidence le fait que les contraintes d'intégrité ont un rôle proscriptif: elles empêchent la base de données d'atteindre un état incohérent; c'est pourquoi, outre de l'aspect descriptif, nous attribuons à la modélisation de la dynamique des données un rôle prescriptif : ainsi elle renseigne sur les actions qu'il faut entreprendre pour que la base de données évolue d'état cohérent à état cohérent.

Cette démarche est importante dans la conception centrée sur la production de système d'information actif; en effet les systèmes d'information implémentés sont traditionnellement des systèmes transactionnels. Ils sont composés d'une base de données et d'un ensemble de programmes paramétrés, appelés transactions qui peuvent être déclenchés à la demande des usagers et exécutés sur le champ. De tels systèmes sont passifs. Aucun changement ne se produit tant qu'un intervenant extérieur n'a pris l'initiative de déclencher l'exécution d'une transaction. Ainsi la décision de provoquer le changement d'état du système d'information est manuelle. Les utilisateurs ont alors la possibilité, dans le cadre de leurs habilitations, de définir une transaction et de l'exécuter. Ainsi les risques d'erreur et d'introduction d'incohérences sont grands. Le système d'information enregistre des changements sans pouvoir contrôler s'ils sont corrects. Pourtant, dans la grande majorité des cas, les conditions dans lesquelles une transaction peut et doit être exécutée sont totalement connues et peuvent être définies lors de la conception, et être automatiquement reconnues lors de l'exécution.

Si un système d'information connaît à l'avance les situations dans lesquelles un changement réel doit être pris en compte, il peut décider du déclenchement des transactions et contrôler leurs exécutions. En agissant ainsi, il devient un système actif, dont le rôle est de reproduire dans le système d'information les changements qui sont intervenus dans la réalité.

Un système d'information actif est un système intégrant un ensemble de mécanismes de synchronisation de la totalité des actions qui s'exécutent en son sein. Ce système actif est muni d'une interface avec le monde réel qui lui permet être renseigné sur les changements d'état qui s'y produisent. Le rôle du système actif est alors reconnaître le changement (ou de le refuser dans le cas contraire) et de le répercuter sur le système d'information . Mais pour remplir ce rôle, le système actif doit disposer d'une description complète de la dynamique du système opérant dans le monde réel.

Le système d'information actif est donc un système de contrôle, qui s'apparente à un système d'exploitation. Comme tout système d'exploitation, le système actif déclenche et contrôle l'exécution des transactions à un instant t, en tenant compte des règles de la dynamique, établies pendant le processus de conception et traduites par les modèles.

Nous avons commencé par établir une distinction entre deux types de contraintes d'intégrité dans les bases de données et utilisant une conception particulière de la notion de modèle, nous montrons comment il est possible de représenter des règles de comportement pour un univers de discours; ce qui permet non seulement d'utiliser la base de données, comme la représentation d'une certaine réalité mais aussi d'un comportement attendu, ainsi que des mesures à adopter lorsque ce qui a été prévu n'est pas conforme au comportement actuel de l'univers du discours.

Bibliographie

[AB81] C.A Alchourron & E. Bulygin, The Expressive Conception of Norms, in: R. Hilpinen (ed.), *New Studies in Deontic Logic*, Reidel, Dordrecht / Boston, 1981, pp. 125-148.

[AM81] C.A. Alchourron & D. Makinson, Hierarchies Of Regulations and Their Logic, in: R. Hilpinen (ed.), *New Studies in Deontic Logic*, Reidel, Dordrecht / Boston, 1981, pp.125-148.

[And 58] A. R. Anderson, A Reduction of Deontic Logic to Alethic Modal Logic, *Mind* 67, 1958, pp.100-103.

[And 67] Anderson, A.R. : Some Nasty Problems in Formalisation of Ethics, *Noús* 1, 1967, pp. 345-360.

[Aqv 67] L. Aqvist, Good Samaritans, Contraty-to-Duty-Imperatives, and Epistemic Obligations, *Noús* 1, 1967, pp.361-379.

[Cas 81] Castaneda, H.N., the Paradoxes of Deontic Logic in : R. Hilpinen (ed.); *New Studies in Deontic Logic*, Reidel, Dordrech /Boston, 1981, pp37-85.

[Che 80] Chellas, B.F., *Modal logic : An introduction*, Cambridge University Press, 1980

[Chis 63] Chisholm, "Contrary-to-duty imperatives and Deontic Logic", *Analysis* 24, 1963.

[FM 92] Fiadeiro, J. and Maibaum, T.S.E, " Temporal reasoning over deontic specifications" *Journal of logic and computation,* 1 (3), 1991.

[Gab 76] Gabbay, D. M, Investigation in Modal Tense Logics with application to problems in philosophy and linguistics, Reidel, Dordrecht/Boston, 1976.

[Harr 84] Harel, D. : Dynamic logic, in : D.M . Gabbay and F. Guenthner (eds), *Handbook of philosophical logic, vol.2.,* Reidel ,1984.

[HuCr 68] Hughes, G.E and Cresswell, M.J, *An intoduction to Modal Logic* (Methuen, 1968).

[Hi88] Hilpinen, R. Deontic Logic: *Introductory and Systematic Readings* (Reidel, 1988).

[Khos 88] Khosla,S. , *System Specification: A Deontic Approach,* Ph.D Thesis, Imperial College, London, 1988.

[KM 87] S. Khosla & T.S.E. Maibaum, The Prescription and Description of State Based Systems, in: B. Banieqbal, H. Barringer & A. Pnueli (eds.), *Temporal Logic in Specification*, Lect. Notes in Comp. Sc. 398, Springer, 1987, pp 243-294.

[Le88] Lee, R.M., Bureaucracies as deontic systems, *ACM Trans. Office Information Syst.* 6, 2 (1988) pp.87-102.

[Meyer 87] Meyer J.J.Ch., A Simple Solution to the "Deepest" Paradox in Deontic Logic, *Logique et Analyse* 117-118, 1987, pp. 81-90.

[Meyer 88] Meyer J.J.Ch., A Different Approach to Deontic Logic : Deontic Logic Viewed as a Variant of Dynamic Logic, *Notre Dame Journal of Formal Logic* 29 (1), 1988, pp.109-196.

[Meyer89] Meyer J.J.Ch., Using Programming Concepts in Deontic Reasoning, in R. Bartsch, J. Van Benthem et P. Van Emde Boas (eds), *Semantics and Contextual Expression,* FORIS Publications, Dordrecht/ Riverton, 1989, pp. 117-145.

[MW 93] Meyer J.J. Ch., & Wieringa, Deontic Logic in computer science, Wiley, 1993.

[NiGa 78] Nicolas, J.M and H. Gallaire, H. Databases : Theorie vs interpretation, in Gallaire and Menker, 1978, pp. 33-54.

[JoSe94] Jones, A. and Sergot, M. Deontic Database Constraints and the Characterisation of Recovery, in *Proceedings of the Second Workshop on Deontic Logic in Computer Science (Deon'94),* Oslo, 1994.

[Rei87] R. Reiter, Nonmonotonic Reasoning, *Annual Reviews of Comp. Sc.* 2, 1987, pp. 147-187.

[Ross 41] Ross, A. Imperatives and Logic, *Theoria* 7, 1941, pp. 53-71.

[RyLe 92] Ryu, Y.U and Lee, R.M, A formal representation of deontic reasoning approach, Research Monograph R-M-1992-08-1, Eramus University Research Institute for Decision and Information Systems, Rotterdam, the Netherlands, 1992.

[Se 85] Searle, J.R. & Vanderveken, D., *Foundations of Illocutionary Logic* (Cambridge University Press, 1985).

[VW63] von Wright,G.H., *Norms and Action* (Routledge and Kegan Paul, 1963).

[VW65] von Wright, G.H., A correction to a new system of deontic logic, Danish Handbook of Philosophy 2 , 1965, pp. 103-107.

[VW68] von Wright, G.H, An essay in deontic logic and the general theory of action, *Acta Philosophica Fennica*, Fasc.XXI (North- Holland1968).

[WR 88] Wieringa, R.J. & R.P. van de Riet, Algebraic specification of object dynamics in knowledge base domains, in *Proc. IFIP TC2/WG8.1 Working Conf. on the Role of Artificial Intelligence in Database and Information Systems*, Canton, China (4-8 [¨[July,1988).

[WIER 89] Wieringa, R.J, Meyer, J.J., et Weigand, H., Specifying dynamic and deontic integrity constraints in *Data & Knowledge Engineering* 4 (1989) North-Holland.