Manuscrit de thèse pour l'obtention du grade de docteur
Erika Valencia
Directeur : Jean-Paul Sansonnet

Outils de topologie algébrique pour la gestion de l'hétérogénéité sémantique entre agents dialogiques

---

LIMSI-CNRS, Bt 508
Université Paris-XI
91403 Orsay cedex
http://www.limsi.fr/Individu/erika

Présentation du document

Ce document expose les travaux de recherche menés au LIMSI (Laboratoire d'Informatique, pour la Mécanique et les Sciences de l'Ingénieur), laboratoire CNRS de l'Université Paris-Sud à Orsay. Ces travaux s'inscrivent dans le cadre de la thèse de doctorat de Erika Valencia, effectuée sous la direction de Jean-Paul Sansonnet DR CNRS au LIMSI, et co-encadrée par Jean-Louis Giavitto CR CNRS au LAMI (Laboratoire de Méthodes Informatiques) de l'Université d'Evry.

La thèse s'insère dans le projet InterViews du LIMSI, dirigé également par Jean-Paul Sansonnet et sur lequel travaillent trois autres doctorants : Sébastien Gérard, Nicolas Sabourret et Guillaume Pitel. Le projet InterViews étudie et met en oeuvre l'interaction de type dialogique entre des composants logiciels via le Web. Ainsi, la thèse a pour sujet d'étude la gestion de l'hétérogénéité sémantique entre de tels composants logiciels. Pour cela, nous proposons : 1) de formaliser ce problème en logique de descriptions, 2) d'utiliser la notion d'adaptation issue des travaux menés en raisonnement par analogie, et d'élaborer des algorithmes en utilisant un formalisme de représentation des connaissances original et puissant basé sur des notions de topologie algébrique.

Remerciements

Remerciements à Gérard Sabah qui m'a donné les outils et les moyens de travailler, à Jean-Paul Sansonnet et Jean-Louis Giavitto, qui m'ont encadrée scientifiquement.

Remerciements aux membres du Jury : Alain Mille, Jean-Pierre Müller, Marie-Christine Rousset, Daniel Kayser, et mes co-encadrants.

Table des matières

.0.0.0.a Chapitre V : Outils de topologie algébrique pour la représentation des connaissances 18

I Formaliser le problème de l'hétérogénéité sémantique entre agents dialogiques 21

I La problématique de l'hétérogénéité sémantique entre agents dialogiques dans le projet InterViews 22

I.1 Le projet InterViews 23

I.1.1 Présentation 23

I.1.1.1 Une problématique émergente 23

I.1.1.2 Objectifs 24

I.1.1.3 Contraintes 24

I.1.1.4 Axes de recherche du projet 24

I.1.2 Description scientifique du projet 24

I.1.2.1 Choix du modèle agent 24

I.1.2.2 Représentation et raisonnement : le langage de conception VDL 25

I.1.2.3 Communication dialogique : le langage de requêtes VQL . . . 25

I.1.2.4 Réalisations sur le domaine applicatif 26

I.2 La notion d'agent dialogique 26

I.2.1 Composition d'un agent dans InterViews 27

I.2.2 Communication et partage entre agents 27

I.2.3 Dialogisme entre agents 28

I.2.3.1 Idée intuitive 28

I.2.3.2 Définition d'une interaction dialogique 29

I.2.3.2.a Un potentiel de sens 29

I.2.3.2.b Un suivi de conversation 29

I.2.4 Interaction dialogique à base de requêtes 29
I.2.4.1 Composition d'une requête 29
I.2.4.2 Réponse associée à la requête 30
I.2.4.3 Accès à une connaissance lors de la réception d'une requête . . 30
I.2.5 Types d'interaction dialogiques 31
I.3 Hétérogénéité sémantique entre agents dialogiques 32
I.3.1 Origines de l'hétérogénéité sémantique dans InterViews 32
I.3.2 Degrés de décalage sémantique dans une interaction 32
I.3.3 Les types d'hétérogénéité sémantique 34
I.3.4 Proposition pour la gestion d'une interaction hétérogène 34
I.4 Conclusions du chapitre 36

II Les Logiques de descriptions comme formalisme de représentation des connaissances 37

II.1 Origines et introduction 38
II.1.1 Les réseaux sémantiques 38
II.1.1.1 Idée de base : réseaux de mots 38
II.1.1.2 Problèmes posés par les réseaux sémantiques 38
II.1.2 Les réseaux SIN 40
II.1.2.1 Idée de base : donner une vraie sémantique aux réseaux sémantiques 40
II.1.2.2 Les concepts dans les réseaux SIN 40
II.1.2.3 Les relations : les rôles 40
II.1.2.4 Le système KL-ONE 41
II.1.2.5 Les SIN et les graphes conceptuels 42
II.1.3 Les domaines de recherche en logiques de descriptions 42
II.1.3.1 Plusieurs langages 42
II.1.3.2 Intégrer du raisonnement 43
II.2 Représentation terminologique 43
II.2.1 Syntaxe de TF 43
II.2.1.1 Les concepts 44
II.2.1.2 Les rôles 45
II.2.1.3 Les constructeurs 45
II.2.1.3.a Les constructeurs de concepts 46

II.2.1.3.b Les constructeurs de rôles . . . 46
II.2.1.4 Syntaxe complète de TF . . . 46
II.2.2 Sémantique dénotationnelle de TF . . . 46
II.2.2.1 Dessiner des ensembles . . . 46
II.2.2.1.a Concepts et rôles . . . 47
II.2.2.1.b Constructeurs . . . 49
II.2.2.2 Formalisation . . . 50
II.2.2.2.a Fonction d'extension . . . 50
II.2.2.2.b Concepts et rôles . . . 51
II.2.2.2.c Constructeurs . . . 51
II.2.2.2.d Autres relations . . . 52
II.2.2.2.e Notion de cycle . . . 53
II.2.3 A propos des notations . . . 53
II.2.3.1 Syntaxe abstraite . . . 53
II.2.3.2 Sémantique . . . 53
II.2.4 Illustration par l'exemple . . . 54
II.2.4.1 Terminologie informelle . . . 54
II.2.4.2 Représentation en réseau SIN . . . 54
II.2.4.3 Définition à l'aide de TF . . . 54
II.3 Bases de connaissances en LD . . . 56
II.3.1 Connaissances intensives et extensives . . . 56
II.3.1.1 Hypothèses sur le monde . . . 56
II.3.1.2 Interprétation ensembliste . . . 56
II.3.1.3 Modèle pour une terminologie . . . 56
II.3.1.4 Les Tbox . . . 58
II.3.2 Langage assertionnel AF . . . 58
II.3.2.1 Syntaxe de AF . . . 59
II.3.2.2 Sémantique formelle de AF . . . 60
II.3.2.3 Les Abox . . . 60
II.3.2.4 Illustration par l'exemple . . . 60
II.3.3 Classifieur pour la base . . . 61
II.4 Les langages de LD plus complexes . . . 62
II.4.1 Famille de référence . . . 62

II.4.2 Tous les constructeurs répertoriés 62
II.4.2.1 Les constructeurs de concepts 62
II.4.2.2 Les constructeurs de rôles 65
II.4.3 Les langages de la famille AL 66
II.4.3.1 Le langage TF de ce document 66
II.4.3.2 Langage FL^{-} 66
II.4.3.3 Langage AL 67
II.4.3.4 Formation des autres langages 67
II.4.4 Les langages mixés 68
II.4.4.1 Nécessité de raisonner sur des individus 68
II.4.4.2 Introduire des constructeurs 68
II.4.4.3 Raisonner avec les langages mixés 69
II.4.4.3.a Assertions implicites : 69
II.4.4.3.b Mixage des assertions et de la terminologie : 69
II.4.4.3.c Abstraction : 70
II.4.4.3.d Restrictions de nombre : 70
II.4.4.3.e Complément : 70
II.4.4.3.f Connecteurs logiques : 70
II.4.4.4 Difficultés 71
II.4.4.5 Traitement de O et B dans les systèmes de LD 71
II.4.4.5.a CLASSIC[10] 71
II.4.4.5.b BACK[99] 71
II.4.5 Introduire des attributs 71
II.4.5.1 Constructeurs d'attributs 72
II.4.5.2 Syntaxe induite dans la Tbox 72
II.4.6 Comparaison des LD avec la logique du premier ordre 72
II.5 Conclusion du chapitre 72

III Formaliser l'hétérogénéité sémantique en logique de descriptions 73

III.1 Agents et interaction : définitions 74
III.1.1 Modéliser les agents dialogiques 74
III.1.1.1 Rappels sur les agents dialogiques 74
III.1.1.2 Ontologie modélisée par une Kbase en logique des descriptions 74

III.1.1.3 Hypothèses sur la Kbase d'un agent 75
III.1.2 Interaction dialogique entre agents 75
III.1.2.1 Pas d'interaction 75
III.1.3 Requête et réponse 76
III.1.3.1 Typage des variables libres 76
III.1.3.2 Notation 76
III.1.3.3 Cohérence avec la Kbase de l'envoyeur 76
III.2 Un pas d'interaction entre deux agents hétérogènes 77
III.2.1 Notion de point de vue pour la gestion 77
III.2.1.1 Point de vue global 77
III.2.1.2 Point de vue local 77
III.2.2 “Résoudre” *versus* “gérer” l'hétérogénéité sémantique 78
III.2.2.1 Point de vue global : résolution 78
III.2.2.2 Point de vue local : gestion 78
III.2.2.3 Qu'est-ce que gérer l'hétérogénéité sémantique ? 79
III.2.3 Echec d'une interaction sémantiquement hétérogène 79
III.2.3.1 Not-understanding problem 79
III.2.3.2 Misunderstanding problem 80
III.2.3.3 Gérer l'hétérogénéité à l'échelle d'une requête 81
III.2.4 Une interaction sémantiquement hétérogène 81
III.2.4.1 Définitions 82
III.2.4.2 Union inconsistante des Kbases en interaction dialogique . . . 82
III.2.4.2.a Les Kbases doivent être différentes 83
III.2.4.2.b Les Kbases doivent partager la connaissance d'au moins un terme 83
III.2.4.2.c Cas d'interaction sans communication 83
III.2.4.3 Conséquence sur l'intersection des Kbases 83
III.2.5 Implications sur les domaines d'interprétation des Kbases hétérogènes . 84
III.2.5.1 Intersection de domaines 84
III.2.5.1.a a - Ontologie globale 84
III.2.5.1.b b - Pas de communication 84
III.2.5.1.c c - Une Kbase commune : la Base 84
III.2.5.1.d d - Une Kbase initiatrice : le Germe 86

III.2.5.2 Différences de domaines 86
III.3 Etude d'un exemple d'interaction hétérogène entre deux Kbases 87
III.3.1 Préliminaires : gestion de l'interaction 87
III.3.1.1 Algorithmes 87
III.3.1.1.a A1 : Gestion de l'interaction. 88
III.3.1.1.b A2 : Gestion d'un pas d'interaction 88
III.3.1.1.c A3 : Gestion de la requête 88
III.3.1.1.d A4 : Gestion de l'hétérogénéité sémantique 88
III.3.1.2 Requêtes possibles 88
III.3.2 Exemple abstrait : deux agents dialogiques 89
III.3.2.1 Hétérogénéité sémantique terminologique 89
III.3.2.2 Requête non-hétérogène : réduction des mondes possibles . . . 90
III.3.2.3 Requête hétérogène 90
III.3.2.4 Proposition pour gérer l'hétérogénéité de termes 91
III.3.3 Hétérogénéité de type référentielle 92
III.3.3.1 Identifieurs globaux 92
III.3.3.2 Identifieurs locaux 92
III.4 Conclusions du chapitre 92

II Un modèle topologique pour l'adaptation dans l'analogie 93

IV Le raisonnement à base de cas et l'adaptation en Intelligence Artificielle 94
IV.1 Le raisonnement à partir de cas 94
IV.1.1 Principe et définitions 95
IV.1.2 Domaines d'application en l'Intelligence Artificielle 96
IV.1.2.1 Recherche d'information 96
IV.1.2.2 Bases de données 97
IV.1.3 Description scientifique du RàPC 98
IV.1.3.1 Le cycle du RàPC 98
IV.1.3.2 Importance de la représentation pour la formulation 98
IV.1.3.2.a Attributs-valeurs : 99
IV.1.3.2.b Arbres de décision : 99
IV.1.3.2.c Prédicats de contraintes : 99

IV.1.3.2.d Règles : 100
IV.1.3.2.e Représentation pour les domaines d'application du RàPC 100
IV.1.3.3 Similarité et sélection des cas 100
IV.1.3.4 Adaptation d'un cas 101
IV.2 Adaptation et analogie 101
IV.2.1 Raisonner par analogies 101
IV.2.1.1 Les contributions de la philosophie 101
IV.2.1.2 Les contributions de la psychologie 102
IV.2.1.3 Définition 103
IV.2.2 Le système ANALOGY proposé par Evans 104
IV.2.3 SME : système basé sur le modèle proposé par Gentner 104
IV.2.3.1 Présentation du modèle théorique SMT 104
IV.2.3.1.a Préférence pour les relations : 105
IV.2.3.1.b Principe de systématicité : 105
IV.2.3.2 Le système SME correspondant à SMT 105
IV.2.4 Le modèle de Holyoak et Thagard : ACME 106
IV.2.5 Chalmers, French, Mitchell, Hofstadter : Le système Copycat 107
IV.3 Conclusions du chapitre 108

V Outils de topologie algébrique pour la représentation des connaissances 109
V.1 Éléments de topologie algébrique 110
V.1.1 Objet d'étude 110
V.1.2 Simplexes et complexes 111
V.1.3 Complexes et graphes 111
V.1.4 Chemin polygonal 113
V.2 Q-analyse 113
V.2.1 Présentation 113
V.2.2 Représenter des relations binaires 114
V.2.3 Représentation duale 115
V.2.4 Q-connectivité 115
V.3 Représentation de prédicats 117
V.3.1 Exemples et analyse 119
V.4 Un formalisme simplicial de représentation des connaissances 121

V.4.1 Construction par catégorisation 121
V.4.2 Propriétés 122
V.4.2.1 Représenter un concept 122
V.4.2.2 Hiérarchies de classes 123
V.4.2.3 Traitement des exceptions 125
V.4.2.4 Plasticité 125
V.4.2.5 Abstraction 126
V.5 Conclusions du chapitre 126

VI Le système ESQIMO **128**
VI.1 Les problèmes du type test de Q.I. 129
VI.1.1 Définition et présentation 129
VI.1.2 Le système ANALOGY 130
VI.2 Étude expérimentale 131
VI.2.1 Le protocole 132
VI.2.2 Résultats et commentaires 133
VI.3 Présentation du système ESQIMO 136
VI.3.1 Modélisation des données du problème 136
VI.3.1.1 Les données du problème 136
VI.3.1.2 L'ensemble des éléments Ω 137
VI.3.1.3 Représentation des propriétés 137
VI.3.2 Transformations de représentations simpliciales pour l'adaptation 141
VI.3.2.1 Figures simples à un seul élément 141
VI.3.2.2 Appliquer une transformation à un simplexe quelconque 142
VI.3.3 Procédure générale de résolution d'analogies 142
VI.3.3.1 Figures générales A et B non élémentaires 142
VI.3.3.2 Appariements des sommets de C_A et C_B 144
VI.3.3.3 Application de la transformation à C_C, et construction de la solution C_D 146
VI.3.4 Algorithme de ESQIMO et remarques sur l'implémentation 148
VI.3.4.1 Représentations 149
VI.3.4.2 Appariements 149
VI.3.4.3 Les chemins 149
VI.3.4.4 Les déformations élémentaires 149

VI.3.4.5 Résolution générale 150
VI.4 Exemples commentés 150
VI.4.0.6 Exemple 1 150
VI.4.0.7 Exemple 2 151
VI.4.0.8 Exemple 3 152
VI.4.0.9 Exemple 4 153
VI.4.0.10 Exemple 5 154
VI.4.0.11 Exemple 6 154
VI.4.0.12 Exemple 7 155
VI.4.0.13 Exemple 8 156
VI.5 Discussion de l'algorithme ESQIMO 157
VI.5.1 Description d'une configuration 157
VI.5.2 Le choix de l'unité de transformation 158
VI.5.3 Détermination du domaine d'une transformation 158
VI.5.4 Le choix d'un chemin 159
VI.5.5 La transformation associée à un chemin 159
VI.5.6 La formalisation de la similitude des chemins 159
VI.5.6.0.a Homotopie de chemins. 160
VI.5.6.0.b Cobordisme. 160
VI.6 Conclusions sur ESQIMO 160

III L'adaptation pour la gestion de l'hétérogénéité sémantique entre bases de connaissances en logique de descriptions. 162

VII Les Complexes Simpliciaux pour Représenter les Relations d'une Base en Logique de Descriptions 163

VII.1 La logique des prédicats du premier ordre 164
VII.1.1 Un formalisme de représentation logique 164
VII.1.2 Les entités manipulées en LPO 164
VII.1.2.0.c Remarque : 165
VII.1.3 Sémantique classique de la LPO 165
VII.1.3.1 Modèle 165
VII.1.3.2 Sémantique dénotationnelle 166
VII.1.3.3 Interpréter une formule 166

VII.1.3.4 Base de connaissances en LPO 166
VII.1.4 Recensement des interprétations ensemblistes de structures de la LPO . 166
VII.1.4.1 Constantes : domaine d'interprétation 166
VII.1.4.2 Variables 167
VII.1.4.3 Quantifications 167
VII.1.4.4 Domaine de validité d'un prédicat 168
VII.1.4.5 Domaine de validité des formules composées 168
VII.2 Modèles pour la LPO en CS 169
VII.2.1 Représentation des constantes et variables 170
VII.2.2 Représentation des quantificateurs 170
VII.2.2.1 Quantification universelle 170
VII.2.2.2 Quantification existentielle 171
VII.2.3 Représentation des prédicats du premier ordre 171
VII.2.3.1 Une première proposition : représenter avec des simplexes . . . 171
VII.2.3.2 Une deuxième proposition : représenter avec des complexes . . 172
VII.2.4 Connecteurs 172
VII.2.4.1 Connecteur ¬ 172
VII.2.4.2 Degré de complémentarité entre CS 172
VII.2.4.2.a Proposition a) 173
VII.2.4.2.b Proposition b) 173
VII.2.4.2.c Synthèse des propositions a) et b) 173
VII.2.4.2.d Les degrés pour des requêtes différentes 174
VII.2.4.3 Connecteur ∧ 174
VII.2.4.4 Connecteur ∨ 174
VII.2.4.5 Connecteur → 174
VII.2.5 Conclusion 175
VII.3 Exprimer une base de LD en CS 175
VII.3.1 Les éléments d'une LD 175
VII.3.2 Les graphes de description 175
VII.3.2.1 Définition d'un graphe de descriptions 176
VII.3.2.2 Sémantique 177
VII.3.3 Un Modèle pour les ABox en CS 177
VII.3.3.1 Représenter la relation d'instanciation avec des CS 178

VII.3.3.2 Analyse de la représentation en CS d'une ABox. 179
VII.3.4 Une représentation des TBox en CS 179
VII.3.4.1 Liens entre CS et GD 179
VII.3.4.2 Les relations part-of en LD 180
VII.3.4.3 Les agrégats 180
VII.3.4.4 Toutes les relations d'une Tbox 183
VII.3.4.5 Une représentation en CS par type de problème 184
VII.4 Conclusions du chapitre 184

VIII Analyse pour une intégration des outils et exemples **185**
VIII.1 Procédure générale de résolution d'une interaction hétérogène 186
VIII.1.1 Traitement général : rappels 186
VIII.1.1.1 Gestion d'un cycle d'interaction. 186
VIII.1.1.2 Gestion d'un pas d'interaction 187
VIII.1.1.3 Gestion de la requête 187
VIII.1.1.4 Gestion des patterns de la requête 188
VIII.1.1.4.a Appariement et adaptation pour le traitement d'un pattern 188
VIII.1.1.5 Substitution valide et production d'une réponse 189
VIII.2 Extraction de complexes à partir d'une base de connaissances en LD 189
VIII.2.1 Les différentes fonctions d'extraction 189
VIII.2.2 Extraction des complexes d'inclusion par $A_{X:\leq}$. 190
VIII.2.2.1 La relation d'inclusion $\leq$. 190
VIII.2.2.1.a Hiérarchies de classes 190
VIII.2.2.1.b Inclusions de concepts 191
VIII.2.2.2 Algorithme de $A_{X:\leq}$ 191
VIII.2.2.2.a Définition 191
VIII.2.2.2.b Premier exemple 192
VIII.2.2.2.c Deuxième exemple 192
VIII.2.2.2.d Troisième exemple 192
VIII.2.2.3 Procédure d'extraction automatique des complexes d'inclusion 193
VIII.2.2.4 Illustration de l'algorithme $A_{X:\leq}$ 194
VIII.2.2.4.a Le Système $Incl_e$ 195
VIII.2.2.4.b Le Système $Incl_r$ 195

VIII.2.2.5 Utilisation de constructeurs dans les relations d'inclusion . . . 197
VIII.2.3 Extraction des complexes d'équation avec $A_{\doteq}$ 198
VIII.2.3.1 Les connecteurs logiques 198
VIII.2.3.1.a Traitement de la conjonction 198
VIII.2.3.1.b Traitement de la négation 199
VIII.2.3.2 Les restrictions 199
VIII.2.3.2.a Restriction de valeur $A_{x:\doteq:\forall}$ 199
VIII.2.3.2.b Les restrictions de nombre $A_{X:\doteq:n}$ 201
VIII.2.4 Extraction des complexes d'extension par A_{role} et $A_{\in}$ 201
VIII.2.4.1 Les complexes d'instanciation de concepts 201
VIII.2.4.2 Les complexes de rôles 202
VIII.2.5 Conclusion sur les extractions de complexes 203
VIII.3 Gestion d'une requête non hétérogène 203
VIII.3.1 Toutes les requêtes qu'il est possible de poser à une Kbase en LD . . . 204
VIII.3.1.1 Rappels de syntaxe 204
VIII.3.1.2 Sémantique intuitive des requêtes 204
VIII.3.1.3 Opérations simpliciales correspondantes 205
VIII.3.1.3.a Requêtes sur des concepts en relation d'inclusion . . . 205
VIII.3.1.3.b Les requêtes sur les extensions 206
VIII.3.1.3.c Les requêtes sur les équations 207
VIII.3.2 Traitement des requêtes 208
VIII.3.2.1 Deux cas de figures 208
VIII.3.2.2 Résolution par matching direct 208
VIII.3.2.2.a Requête sur un concept 208
VIII.3.2.2.b Requête sur un individu 209
VIII.3.2.2.c Comment lire une substitution sur une représentation simpliciale ? 210
VIII.3.2.3 Cas de matching avec déduction 211
VIII.3.2.4 Requête sur un concept 211
VIII.4 Traitement des requêtes hétérogènes 212
VIII.4.1 Préliminaires 212
VIII.4.2 Un problème de compréhension 213
VIII.4.3 L'algorithme d'adaptation d'ESQIMO 213

VIII.4.3.1 Représenter toutes les données 214
VIII.4.3.2 Appariement structurel 214
VIII.4.3.3 Transformer le cas source 214
VIII.4.4 Représentation simpliciale de la requête 215
VIII.4.5 Algorithme d'adaptation $A_{\div}$ de la requête avec le complexe d'inclusion extrait de Kb_r 216
VIII.4.5.1 Appariements dans ESQIMO ESQ_{app} 216
VIII.4.5.2 Adaptation des termes hétérogènes $A_{\div}$ 216
VIII.4.5.3 Fonction de test A_{test} 217
VIII.4.5.4 Fonction d'identification des candidats à l'adaptation A_{id} . . . 217
VIII.4.5.5 Application à l'exemple 218
VIII.4.6 Production de substitutions Valides 219
VIII.4.7 Autres exemples 221
VIII.4.7.1 Requête hétérogène sur un individu 221
VIII.4.7.2 Requête hétérogène sur un rôle 221
VIII.5 Conclusions sur l'intégration des outils 222

Introduction de la thèse

La thèse se place dans le cadre du projet InterViews du LIMSI. Ce projet est proposé pour aborder les nouveaux domaines de recherche découlant de l'évolution du Web sur Internet. Dans ce cadre, nous nous intéressons au problème de l'hétérogénéité sémantique entre agents dialogiques. Un agent dialogique est un composant logiciel qui est capable d'interagir à un haut niveau sémantique avec d'autres agents (logiciels ou humains). Pour cela, il doit être capable de se représenter son fonctionnement et de raisonner dessus pour recevoir des requêtes, fournir des explications et participer à des tâches d'assistance.

Notre travail a commencé par l'identification d'un nouveau problème entre agents interagissant par dialogisme : *l'hétérogénéité sémantique*. Ce problème, reconnu comme fondamental aujourd'hui par la communauté des systèmes multi-agents, doit être distingué du problème de l'intégration de bases de données hétérogènes. Il a donc fallu délimiter, définir et formaliser ce problème nouveau. Pour cela nous avons utilisé la *logique des descriptions*.

Nous avons proposé une solution nouvelle et originale qui est d'utiliser un raisonnement à partir de cas afin de trouver une interprétation aux termes hétérogènes dans une requête. Cela nous a conduit à étudier plus spécifiquement l'étape *d'adaptation* dans le raisonnement par analogie, pour laquelle nous avons fourni un modèle fondé sur des outils de la *topologie algébrique*. Ce modèle a mené à l'implémentation d'un système de résolution automatique d'analogies simples.

Nous avons ensuite montré pourquoi et comment il est possible d'intégrer tous ces outils dans un système complet et cohérent pour le traitement de l'hétérogénéité sémantique dans une interaction dialogique.

Ce travail, qui se veut précurseur, mérite d'être poursuivi afin de le mener vers des étapes plus implémentatoires ; en particulier, nous souhaiterions : implémenter les différents algorithmes présentés dans le langage développé pour la plate-forme InterViews, puis les tester lors de l'interaction dialogique d'agents.

Partie 1

Dans cette partie, nous présentons le projet InterViews et nous proposons une formalisation du problème de l'hétérogénéité sémantique en Logique des Descriptions.

Chapitre I : La problématique de l'hétérogénéité sémantique entre agents dialogiques dans le projet InterViews

De plus en plus d'applications sont distribuées et interagissent au travers du Web. Par ailleurs, de nouvelles contraintes imposent une recherche pour une interaction plus naturelle entre l'utilisateur de telles applications et les applications elles-mêmes. Enfin, les applications concernées contiennent de plus en plus de composants dynamiques.

Ainsi, le projet InterViews vise l'étude et la mise en oeuvre de l'interaction dialogique entre plusieurs composants logiciels via le Web. Pour cela, il paraît intéressant et naturel de modéliser ce type de problématique comme un système multi-agent dont les agents sont des composants logiciels dynamiques disposant d'une représentation des connaissances et de capacités de raisonnement et d'interaction. L'interaction entre de tels agents est alors dialogique, dans le sens où elle est un sous-ensemble de la langue naturelle et où elle permet la communication à un haut niveau sémantique. Ce champ applicatif se répartit alors en plusieurs domaines de recherche : l'interaction dialogique dans un système multi-agents ; la représentation des connaissances ; le raisonnement sur le fonctionnement.

Dans ce cadre, un problème qui se pose immédiatement est l'hétérogénéité sémantique lors de l'interaction dialogique entre deux agents. Le problème de l'Hétérogénéité Sémantique (HS), qui a depuis été reconnu comme crucial par la communauté agent, est à différencier de l'hétérogénéité logicielle et matérielle (problèmes qui ont déjà été abordés et pour lesquels des solutions efficaces sont déjà mises en oeuvre), et de l'intégration de bases de données hétérogènes (domaine où l'on vise à homogénéiser deux sources présentant une hétérogénéité sémantique).

En effet, la subtilité de la gestion du problème de la HS dans InterViews est de gérer l'hétérogénéité lors d'une interaction pour rendre la communication possible, tout en préservant les différences et en évitant le plus possible une homogénéisation, qui est toujours difficile à mettre en oeuvre en environnement contrôlé, et impossible à atteindre dans les cas généraux des systèmes *ouverts* [49] de l'Internet. Pour ces raisons, il est délicat et important de bien délimiter et bien définir le problème de la HS entre agents avec un formalisme rigoureux et ayant fait ses preuves en représentation des connaissances.

Chapitre II : Les logiques de descriptions comme formalisme de représentation des connaissances

La question est alors de savoir avec quels outils de formalisation nous pourrions proposer une définition formelle de ce problème. Le plus naturel semble de choisir un formalisme de représentation des connaissances qui possède une sémantique bien définie. Ainsi nous pouvons modéliser la base de connaissances (Kbase) de chaque agent et poser le problème de l'hétérogénéité sémantique qui existe entre eux.

On remarque que la gestion d'une telle HS passe par la gestion de la HS au niveau *terminologique*. Il est alors encore plus normal d'utiliser un formalisme adapté à la représentation des terminologies.

Le formalisme choisi est donc la Logique des Descriptions (LD), que nous présentons historiquement, syntaxiquement et sémantiquement dans ce chapitre.

Chapitre III : Formaliser l'hétérogénéité sémantique en logique de descriptions

Dans un premier temps, nous reprenons la définition des agents dialogiques formulée dans le chapitre I, que nous formalisons en LD. Nous en déduisons la définition en LD d'une interaction entre agents dialogiques.

Ensuite, nous posons les hypothèses de la *gestion* de l'hétérogénéité sémantique entre agents dialogiques modélisés en LD, par opposition à sa *résolution* qui mènerait à l'homogénéisation des Kbases. Compte tenu de ces contraintes, nous définissons ensuite formellement la HS entre agents dans InterViews

Enfin, nous synthétisons les deux principaux types de problèmes de HS qui peuvent émerger entre agents en LD : la HS terminologique, et la HS référentielle. Nous les illustrons par des exemples d'interactions dialogiques, et nous proposons une stratégie de gestion pour chacun d'eux : *l'adaptation* pour la HS terminologique et *l'accès par le contenu* pour la HS référentielle.

Partie 2

Pour gérer la HS que nous avons définie formellement, nous avons montré qu'il faut passer par l'étape de *gestion* d'une requête *hétérogène* à *un* seul pas d'interaction par un seul agent, celui qui *reçoit* la requête. On restreint alors notre problème à ce sous-problème.

Nous proposons alors une idée originale : interpréter une requête en LD qui contient des termes inconnus (hétérogènes) reviendrait en fait à faire du raisonnement à partir de cas (RàPC). En effet, la requête contenant des termes inconnus constitue un cas nouveau, et elle sera résolue (interprétée) en utilisant les expériences passées, c'est-à-dire, les termes connus, pour donner un "sens" à cette requête.

Chapitre IV : Le raisonnement à partir de cas et l'adaptation en Intelligence Artificielle

Nous sommes alors amenés à nous intéresser au raisonnement à partir de cas. Le RàPC se décompose en cycles de raisonnement qui incluent les étapes suivantes : 1) stocker les cas, 2) retrouver un cas (sélectionner), 3) adapter la solution, et enfin 4) l'apprentissage du cas avec sa solution.

Afin de transposer ce type de raisonnement à une Kbase écrite en LD, nous nous intéressons à l'implémentation de ces étapes dans un système informatique d'écriture de Kbases en LD. Grâce aux moteurs de classification fournis par les systèmes de LD, comme LOOM par exemple, nous disposons d'algorithmes pour les étapes de stockage et recherche de cas. Une étape devient alors cruciale et nouvelle : c'est celle de *l'adaptation* de la solution.

Notre idée se transforme alors en : comment effectuer l'étape d'adaptation structurelle d'un cycle RàPC avec une base écrite en LD en vue de traiter des termes nouveaux à l'aide de termes connus dans une requête hétérogène ?

Nous restreignons alors notre étude à l'adaptation : pour l'étudier, nous nous intéressons au raisonnement simple dit par analogie. Nous en donnons une présentation, avec les modèles pour l'analogie développés en Intelligence Artificielle.

A ce stade, nous voudrions pouvoir avoir un algorithme qui nous permette d'effectuer une adaptation sur des définitions structurelles (car c'est la caractéristique intéressante d'une représentation avec les LD). On considère alors l'analogie comme un appariement structurel, ce qui nous permet d'utiliser une métaphore spatiale pour engendrer l'algorithme en question.

.0.0.0.a Chapitre V : Outils de topologie algébrique pour la représentation des connaissances La métaphore spatiale nous mène à l'utilisation de la la topologie algébrique que l'on présente comme formalisme de représentation des connaissances dans ce chapitre.

Nous introduisons les concepts élémentaires de topologie algébrique utilisés par la suite : les complexes simpliciaux (CS). Nous présentons une méthode de représentation de relations binaires avec les CS : la Q-Analyse. Nous proposons ensuite un moyen d'utiliser cette Q-analyse pour représenter les prédicats et les concepts. Nous donnons des exemples pour argumenter notre proposition et la comparer à d'autres systèmes. Cette méthode de représentation des connaissances est appelée CAT pour Combinatorial Algebraic Topology. Nous fournissons également une processus de construction d'une telle représentation, par catégorisation, pour un agent disposant de détecteurs assimilés à des prédicats unaires.

Chapitre VI : Un système topologique pour la résolution automatique d'analogies : Le système ESQIMO

Avec le formalisme de représentation des connaissances CAT, basé sur la topologie algébrique, nous proposons un algorithme d'adaptation de structures pour l'analogie et nous l'illustrons par la programmation d'un petit système de résolution automatique d'analogies simples : ESQIMO. ESQIMO a été prototypé en Mathematica puis programmé en Caml, et testé avec succès sur des exemples simples.

Dans ce système, nous choisissons l'application paradigmatique des tests de QI géométriques, que nous testons d'abord *sur des humains* afin de trouver des heuristiques de résolution intéressantes à inclure dans l'algorithme. Nous modélisons ensuite les connaissances pertinentes d'un tel problème à l'aide de complexes simpliciaux. Enfin nous exposons dans le détail l'algorithme lui-même qui modélise une adaptation comme une déformation, c'est-à-dire une transformation composée de transformations élémentaires d'un simplexe en un autre le long d'un chemin polygonal, le tout plongé dans une représentation simpliciale plus grande représentant l'espace des possibles lié au problème.

Partie 3

En troisième partie, nous voulons appliquer l'algorithme topologique développé dans la partie 2 à la gestion d'une requête hétérogène par un seul agent, puis à l'interaction dialogique entre deux agents dialogiques dans InterViews.

Chapitre VII : Les complexes simpliciaux pour représenter les relations d'une base de connaissances écrite en logique de descriptions

Le but de ce chapitre est de comparer le formalisme CAT des Complexes Simpliciaux à la LD pour éventuellement les proposer comme représentation des connaissances, alternative ou complémentaire, afin de résoudre les deux types de HS entre agents. Pour cela, nous utilisons le fait qu'une base LD peut être exprimée en Logique des Prédicats du Premier Ordre (LPO), il nous reste alors à savoir si une base en LPO peut s'exprimer sous la forme des CS.

Notons qu'une fois la correspondance entre LPO et CS établie, nous ne cherchons pas à calculer une correspondance entre la LD et les CS, en passant par la LPO. Nous utilisons ce résultat seulement pour confirmer qu'il est possible d'exprimer des connaissances contenues dans une Kbase écrite en LD, à l'aide de complexes simpliciaux. Nous nous attacherons à montrer comment on peut alors exprimer les relations d'une Tbox et d'une Abox. En d'autres termes, la LPO sert ici pour une comparaison en termes d'expressivité.

Nous présentons quelques rappels de LPO, en particulier sur les aspects de sémantique dénotationnelle. Nous proposons ensuite une correspondance avec le formalisme CAT introduit au chapitre V. Nous verrons que cette correspondance se traduit soit par une extraction de complexes à partir directement des relations d'inclusion de la Tbox, ou alors par une interprétation des formules de LD en termes de complexes simpliciaux (qui sont alors les extensions).

Chapitre VIII : Analyse pour une intégration des outils et exemples.

Nous illustrons dans ce chapitre l'intégration des outils proposés aux chapitres III, V et VI, en nous appuyant sur les correspondances établies au chapitre VII. Pour cela, nous détaillons toutes les étapes du traitement d'une requête (hétérogène ou pas) par un agent récepteur : depuis l'extraction d'une représentation simpliciale jusqu'à la production d'une substitution valide.

Pour cela, nous nous concentrons dans un premier temps sur l'illustration de tous les processus d'extraction des complexes simpliciaux en fonction des relations de la Kbase écrite en LD. Une fois ces processus illustrés, nous pouvons utiliser ces représentations simpliciales pour traiter explicitement les différentes requêtes qu'il est possible de poser à un agent. Pour cela, nous listons toutes les requêtes syntaxiquement correctes, et nous nous intéressons aux opérations simpliciales qu'il faut effectuer pour y répondre.

Finalement, nous essayons d'élargir notre illustration en sortant du cadre d'un seul pas d'interaction et en traitant une interaction entre deux applications réelles de InterViews.

Première partie

Formaliser le problème de l'hétérogénéité sémantique entre agents dialogiques

Chapitre I

La problématique de l'hétérogénéité sémantique entre agents dialogiques dans le projet InterViews

Introduction

De plus en plus d'applications sont distribuées et interagissent au travers du Web. Par ailleurs, de nouvelles contraintes imposent une recherche pour une interaction plus naturelle entre l'utilisateur de telles applications et les applications elles-mêmes. Enfin, les applications dont il s'agit contiennent de plus en plus de composants dynamiques.

Ainsi, le projet InterViews vise l'étude et la mise en oeuvre de l'interaction dialogique entre plusieurs composants logiciels via le Web. Pour cela, il paraît intéressant et naturel de modéliser ce type de problématique comme un système multi-agent dont les agents sont des composants logiciels dynamiques disposant d'une représentation des connaissances et de capacités de raisonnement et d'interaction. L'interaction entre de tels agents est alors *dialogique*, dans le sens où elle est un sous-ensemble de la langue naturelle et où elle permet la communication à un haut niveau sémantique. Ce champ applicatif se répartit alors en plusieurs domaines de recherche : l'interaction dialogique dans un système multi-agents ; la représentation des connaissances ; le raisonnement sur le fonctionnement.

Dans ce cadre, un problème qui se pose immédiatement est l'hétérogénéité sémantique lors de l'interaction dialogique entre deux agents. Le problème de l'Hétérogénéité Sémantique (HS), qui a depuis été reconnu comme crucial par la communauté agent, est à différencier de l'hétérogénéité logicielle et matérielle (problèmes qui ont déjà été abordés et pour lesquels des solutions efficaces sont déjà mises en oeuvre), et de l'intégration de bases de données hétérogènes (domaine où l'on vise à homogénéiser deux sources présentant une hétérogénéité sémantique).

En effet, la subtilité de la gestion du problème de l'hétérogénéité sémantique dans InterViews est de gérer l'hétérogénéité lors d'une interaction pour rendre la communication possible, tout en préservant les différences et en évitant le plus possible une homogénéisation, qui est toujours difficile à mettre en oeuvre en environnement contrôlé, et impossible à atteindre dans les cas généraux des systèmes *ouverts* de l'Internet. Pour ces raisons, il est délicat et important de bien délimiter et de bien définir le problème de la HS entre agents avec un formalisme rigoureux et ayant fait ses preuves en représentation des connaissances.

Résumé du chapitre

Dans ce chapitre, nous introduisons le cadre applicatif et scientifique du projet InterViews, au sein duquel a été développé notre travail. Nous présentons le projet dans sa globalité, et décrivons les différents volets scientifiques qu'il recouvre. Le projet InterViews a pour objet l'étude et la mise en oeuvre des interactions entre applications actives via Internet. Nous modélisons ces interactions à l'aide du paradigme agent : les applications sont modélisées par des agents dialogiques et sont appelées des *vues*. Un agent dialogique est défini comme étant un composant logiciel qui intègre dans une représentation sémantique unifiée une fonctionnalité *opératoire* (la partie active de l'agent) et une fonctionnalité *explicatoire* (la partie interactive de l'agent).

Dans le volet du projet concernant l'interaction et plus précisément le dialogisme, nous introduisons le problème de l'hétérogénéité sémantique. Nous définissons cette notion dans le cas particulier du projet InterViews, en fonction de nos contraintes et objectifs. Cette définition nous servira dans le chapitre III pour formaliser le problème de l'hétérogénéité sémantique entre agents dialogiques à l'aide des outils formels introduits au chapitre II.

I.1 Le projet InterViews

I.1.1 Présentation

I.1.1.1 Une problématique émergente

Le Web est une des sources d'information électronique les plus importantes actuellement. Cependant, les *interfaces* et les services de raisonnement disponibles pour y accéder sont très limités [32].

De plus, les pages Web sont appelées à devenir de plus en plus le lieu d'interaction *standard* entre les *utilisateurs* et les services en ligne. Elles ont donc un rôle de plus en plus actif et par delà le simple accès aux services et aux informations. Ainsi, en plus de *l'accès* elles doivent désormais aussi assurer *l'accueil*, *l'assistance* et la *formation* pour une classe de plus en plus vaste d'utilisateurs *non-experts* (appelés aussi utilisateurs-citoyens).

Par ailleurs, les *documents* disponibles sur le Web sont de plus en plus enrichis par des applications multi-média qui fournissent de la connaissance sous diverses formes de représentation (texte, graphiques, images animées, vidéo, sons, réalité virtuelle, etc.). Les liens hypertexte entre différents documents représentent des relations entre des entités de connaissance.

I.1.1.2 Objectifs

Le projet InterViews vise ce domaine applicatif et propose d'exploiter les outils de l'Intelligence Artificielle afin d'élaborer une plate-forme d'interaction entre pages incluant des composants actifs et utilisateurs-citoyens au travers d'Internet.

I.1.1.3 Contraintes

Ainsi, les contraintes principales qui ont été fixées pour cette plate-forme sont :

- La communication doit être la plus *naturelle* possible avec les utilisateurs ;
- La communication doit être homogène entre les utilisateurs et les pages, mais aussi entre les pages : choix du mode d'interaction par *dialogisme* ;
- Les pages doivent avoir une *représentation* de leur *contenu* (structure) mais aussi de leur *fonctionnement* ;
- L'interface doit être auto-*explicative* ;
- Les capacités explicatoires incluent des services de raisonnement.

I.1.1.4 Axes de recherche du projet

La mise en oeuvre d'une telle plate-forme se répartit sur trois axes de recherche :

1. Choix et utilisation d'outils de représentation et de raisonnement ;
2. Module d'assistance en langue naturelle ;
3. Application aux systèmes d'information distribués.

Nous allons maintenant détailler cette approche dans chacun de ces axes d'étude, ainsi que les choix de modélisation dans une description plus scientifique du projet.

I.1.2 Description scientifique du projet

I.1.2.1 Choix du modèle agent

Le paradigme des systèmes multi-agents est de plus en plus utilisé pour décrire, étudier et développer des applications faisant entrer en interaction différents *composants logiciels*. En effet, au delà de la simple gestion de bases hétérogènes, il y a un besoin grandissant de prendre en compte les aspects dynamiques et évolutifs des applications, et donc de les considérer comme des *agents* disposant d'un niveau de représentation des connaissances et de capacités de raisonnement pour entrer en interaction avec leur environnement.

Notre étude se place dans le domaine des recherches sur la *communication* entre agents logiciels. Notons que dans ce cadre, un premier pas avait été établi avec la proposition des actes de discours [80], et les langages d'interaction qui s'en inspirent comme KQML [34]. Cependant, afin de ne pas réduire l'interaction à un médium de transport d'information "à la Shannon", la *conversation* [83] a ensuite été considérée comme une suite d'interactions co-construites et non pas indépendantes. C'est dans cette lignée que s'est inscrite l'introduction de la notion de *dialogisme* [25]. Le dialogisme est un mode de communication entre un humain et un composant logiciel et peut être étendu à un mode de communication entre deux agents logiciels, nous définirons plus précisément cette notion dans la section I.2.3.

Ainsi, le projet InterViews a pour principal objet, l'étude des *Agents Dialogiques* : un agent dialogique modélisera donc un composant logiciel et *intègre* ainsi dans une représentation sémantique *unifiée* :

- une fonctionnalité opératoire (la partie active de l'agent),
- et une fonctionnalité "explicatoire" (la partie interactive de l'agent).

Contrairement à un composant logiciel classique, un agent dialogique est donc capable de se représenter son fonctionnement, de raisonner dessus et d'interagir à un haut niveau sémantique pour : recevoir des requêtes, fournir des explications et participer à des tâches d'assistance ce qui correspond aux contraintes posées au paragraphe précédent.

Nous soulignons le fait que dans InterViews, les outils de conception des agents intègrent les caractéristiques entre les agents. Cette approche est de plus en plus reconnue comme nécessaire dans la communauté des systèmes multi-agents [74].

I.1.2.2 Représentation et raisonnement : le langage de conception VDL

Une communication à un haut niveau sémantique requiert de prendre en considération le comportement et les intentions des agents : pour cela nous avons recours aux actes de langage [80]. Le dialogisme est alors basé sur les actes de langage mais inclut aussi l'explication du fonctionnement et la gestion de l'hétérogénéité sémantique. Un agent dialogique est donc modélisé comme un composant actif cohérent contenant la description de sa structure et de son fonctionnement ainsi que ses capacités de communication.

Un agent doté de telles capacités est appelé une *Vue* dans InterViews. Pour décrire de tels composants, nous proposons un langage de représentation des connaissances propre au projet InterViews : VDL (pour View Design Language). Ce langage doit permettre d'exprimer la structure et le fonctionnement d'une vue, qui définissent sont vocabulaire.

Nous voulons que le modèle de représentation soit *homogène* et permette de produire *génériquement* les réponses aux questions posées aux vues. De plus, le traitement de la *référence* doit recevoir une attention particulière puisqu'il est primordial dans les fonctionnalités d'assistance. L'ensemble du système doit être commandable à partir de *requêtes* en langue *naturelle*, et donc qu'une application puisse être décrite sans écrire une ligne dans un langage formel. La conception d'une vue est basée sur un principe de transparence qui permet d'approcher une programmation par requêtes (et donc basée sur l'interaction elle-même), définie par le langage VQL (View Query Language), décrit ci-dessous.

Enfin, nous proposons de séparer les représentations et définitions de concepts et d'individus dans les vues permettant une dissociation qui est nécessaire pour le traitement de la référence dans l'interaction en langue pseudo-naturelle.

I.1.2.3 Communication dialogique : le langage de requêtes VQL

Pour répondre aux requêtes qui lui sont adressées, le programmeur d'une vue doit pouvoir indiquer à quelles unités de sens sont reliées les images (et mots-clés) qu'il utilise pour définir son interface. Chaque entité définie comme phénomène (manifestation) d'une vue écrite en VDL doit ainsi être accompagnée d'une sémantique afin d'être interopérable via VQL, le langage de requêtes de InterViews.

L'accès aux vues avec VQL se fait classiquement à l'aide des trois classes d'actes de langage : *ask* qui demande une *information* à la vue, *tell* qui effectue une *assertion* et *achieve* qui envoie une *commande.*

La référence est résolue dans VQL grâce à la distinction entre la référence intrinsèque (incluant les notions d'attributs, et de sous-parties) et la référence extrinsèque (incluant les relations génériques, les niveaux de composition, les relations spatiales et temporelles).

Une requête contient plusieurs informations : les mots de langue naturelle qui sont des déclancheurs de l'action ; les motifs typiques utilisés pour préciser l'action ; les tests de pertinence. Dans VQL, ces trois notions sont notées par les mots-clés respectifs WORDS, RULES et CHECK qui définissent les trois composantes d'une requête.

I.1.2.4 Réalisations sur le domaine applicatif

Dans le but de rendre nos travaux consultables sur le Web, il a été nécessaire de mettre en place une architecture client/serveur. Les outils utilisés par les membres du projet InterViews sont les langages Mathematica, Perl et Java. Le système ainsi obtenu est robuste et pleinement fonctionnel, il est disponible sur le site d'InterViews[1], où l'on peut consulter et interagir avec des vues simples :

- Mike : c'est une interface de contrôle pour un four qui se situe à distance.
- Coco : c'est une vues très simple contenant un compteur dont on peut accélérer ou freiner la vitesse, changer le sens d'incrémentation (ou décrémentation) et le pas, et le stopper.
- Jojo : est une grille de 3×3 cases qui peut contenir des figures géométriques que l'ont peut créer dans une réserve puis placer ou enlever.
- Momo : est un jeu simple de morpion sur une grille 3×3
- ELIZA en français : une application ELIZA, inspirée du ELIZA de Weizenbaum, en français.
- Un monde de cubes : où l'on peut bouger des objets sur des emplacements, en suivant des lois simple de gravité (empilements, dépilements et surface nécessaire pour poser).
- Le jeu des tours de Hanoï, à un seul joueur (l'utilisateur), où l'on doit empiler des disques par ordre de taille en n'utilisant que trois axes d'empilement pour tous les mouvements.
- Le jeu de Nim à deux joueurs (utilisateur et machine), où il faut retirer jusqu'à 4 pièces à la fois sur des axes d'alignement, le joueur retirant la dernière pièce gagnant le jeu.

Notons que plusieurs vues utilisent les concepts (objets et/ou actions) définis dans d'autres vues plus élémentaires. C'est le cas de Momo par rapport à Jojo ou bien du jeu des tours de Hanoï ou de Nim par rapport au monde de cubes.

I.2 La notion d'agent dialogique

Nous avons vu que la notion d'agent dialogique est centrale dans InterViews. Après avoir introduit le projet avec ses objectifs et contraintes ainsi que les axes de recherche qu'il induit,

[1]Site InterViews `http ://www.limsi.fr/Individu/jps/interviews.htm`.

nous allons maintenant développer cette notion centrale pour le projet. Pour cela, nous allons étudier chaque notion clé à savoir les agents, le dialogisme mais aussi l'interaction à base de requêtes et les types d'interaction possibles afin d'en proposer des définitions qui seront la base de notre travail dans les chapitres suivants.

I.2.1 Composition d'un agent dans InterViews

Un composant logiciel incorporé dans une page Web est appelé une vue dans la plate-forme InterViews ; il est entièrement défini par sa représentation en VDL, qui est donc un programme précisant les trois items suivants :

1. La corporalité de l'entité définie : les concepts sont définis en extension et en intension et sont reliés par des liens *exécutables* (de type composant/intégralité).
2. Le fonctionnement de l'entité définie : la partie exécutable est manifestée dans son comportement, elle est éventuellement guidée par un ou plusieurs buts (ou motivations).
3. Les interactions de cette entité : nécessité de définir la syntaxe et les protocoles d'interaction, dépendant des items 1 et 2.

Ces composants sont modélisés comme des agents en interaction au travers d'Internet. Ainsi, les agents A_i dans InterViews sont composés de :

- Une *ontologie* o_i : qui est une spécification explicite d'une *conceptualisation* [43]. Une conceptualisation est une *abstraction* à l'aide de concepts de l'univers connu de l'agent. Elle est une *spécification explicite* en ce qu'elle est représentée à l'aide d'un formalisme de représentation des connaissances. La question de la *création* de l'ontologie (émergence, apprentissage, catégorisation), n'est pas abordée puisque les agents dialogiques sont définis et construits par des utilisateurs.
- Une *mécanique* de fonctionnement f_i : qui est un ensemble de données *exécutables*.
- Un *comportement* c_i : qui est la *manifestation* de l'exécution des procédures de l'agent en utilisant les connaissances représentées dans l'ontologie.

I.2.2 Communication et partage entre agents

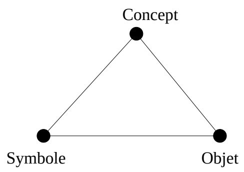
Un système multi-agent (SMA) est composé de plusieurs agents qui opèrent dans un environnement commun [33], [68]. Ces agents peuvent aussi partager des informations et des ressources logicielles dans certains types de SMA. On peut ainsi classer les différents SMA selon le type de ressources partagées par les agents.

Par ailleurs, dans un SMA, les agents interagissent par le biais d'une communication, c'est-à-dire un échange de connaissance à propos de concepts.

Le triangle du sens [81] donné par la figure I.1, nous montre un objet, qui est une *entité* qui *existe* dans l'environnement, par exemple "une pomme" ; un symbole est un *identifiant* qui se *réfère* à cet objet, par exemple le mot "pomme" ; et le concept qui est le *sens* qui est porté par le mot fournissant une *connexion sémantique* entre le symbole et l'objet.

Pour qu'un agent intelligent communique un concept à un autre agent, il doit :

Fig. I.1 – Triangle du sens [81].



1. sélectionner un symbole
2. transmettre le symbole.

L'agent récepteur doit alors :

1. percevoir le symbole
2. interpréter ce symbole : pour en trouver son propre sens.

Quand on dit que deux agents partagent une ressource c'est qu'ils ne passent pas par le biais d'une communication explicite de haut-niveau telle que décrite ci-dessus pour transmettre des connaissances [36]. Les humains, par exemple, ne sont donc pas capables de partager des ressources, tandis que certains types d'agents artificiels le peuvent.

Dans InterViews, nous voulons idéalement que les agents ne partagent pas de ressources. L'interaction entre eux ne peut donc se faire que par une communication de haut niveau sémantique, et nous verrons qu'elle sera dialogique.

I.2.3 Dialogisme entre agents

I.2.3.1 Idée intuitive

Comme nous l'avons déjà mentionné, le dialogisme est un mode de communication entre un humain et un composant logiciel et peut être étendu à un mode de communication entre deux agents logiciels. La langue utilisée par l'humain pour s'adresser au composant artificiel sera un sous-ensemble de la langue naturelle pour plusieurs raisons :

- son interlocuteur n'est pas humain : il ne possède pas de corps humain, n'évolue pas strictement dans le même environnement, possède une expérience réduite et profondément différente, a été conçu dans un but particulier, est dédié à un domaine, se donne à voir à l'humain ;
- les modes d'interaction ne sont pas les mêmes : la conversation a lieu par écrit, au travers d'un clavier, par exemple ;
- l'environnement est différent : les objets à propos desquels il est possible de discourir sont différents et souvent limités en nombre.

La principale implication est la réduction des problèmes issus de subtilités (flexibilité) de la langue naturelle par rapport à l'étude du dialogue entre humains. Ainsi, les agents communicants connaissent leur rôle, leurs devoirs et leurs droits, et les potentiels exploitables chez les autres.

Notons que dans un tel environnement, les communications ont souvent lieu par *nécessité* uniquement et pour atteindre un but factuel, c'est-à-dire effectuer la tâche pour laquelle ils ont été crées ; ce qui a pour conséquence de réduire le domaine possible d'interventions et de styles de dialogue par rapport à la langue naturelle entre humains.

Idéalement, le dialogisme ne devrait pas être soumis à une modalité : ce n'est ni de la langue parlée, ni de l'écrit, c'est encore autre chose, dont la principale caractéristique est la spontanéité [9], il ne suffit pas de réutiliser un langage de requêtes trop pauvre.

En conclusion, nous dirons que le dialogisme est un type de dialogue se restreignant à l'utilisation d'un sous-ensemble de la langue naturelle, dans un environnement délimité, avec des agents artificiels, mais ne pouvant être ramené à un dialogue artificiel, et encore moins à un protocole informatique de transmission de données et/ou de délégation de tâches.

I.2.3.2 Définition d'une interaction dialogique

Plus formellement, le dialogisme doit être considéré comme présentant deux principales caractéristiques [25] :

I.2.3.2.a Un potentiel de sens Le *sens* d'une requête dialogique est co-construit par les inter-actants. Un message n'a pas un sens *a priori*, mais seulement un *potentiel* de sens. En effet, l'agent émetteur exprime un message porteur d'une signification locale, mais qui n'est pas universellement définie. Le récepteur doit alors en trouver une *interprétation* selon ses connaissances locales sur lui-même, sur l'émetteur et sur le contexte. Le sens interprété par le récepteur peut alors être différent de celui que l'émetteur avait affecté à son message.

I.2.3.2.b Un suivi de conversation C'est le mécanisme qui permet de maintenir la compréhension dans l'interaction malgré les différences potentielles de sens accordées au même message par deux inter-actants. En effet, l'émetteur doit s'assurer que le sens qu'il a affecté à son message et celui qui a été trouvé par le récepteur sont compatibles. Il s'agit alors pour les inter-actants de détecter les *incohérences*, révélatrices d'incompréhensions.

Nous retenons de ces caractéristiques la définition suivante :

Definition | *Une interaction dialogique est une interaction à un haut niveau sémantique. Une interaction dialogique est une suite d'échanges de requêtes ayant un potentiel de sens. Cette suite d'interaction peut permettre de résoudre les incohérences.*

I.2.4 Interaction dialogique à base de requêtes

I.2.4.1 Composition d'une requête

Interagir c'est échanger de l'information par des requêtes, cette information se situe à plusieurs niveaux :

1. Dans l'acte d'interagir ;

2. Dans le contexte de l'interaction ;
3. Dans le type d'acte de language, le type de requête (exemples *: ask*, *tell*, *achieve*) ;
4. Dans le contenu des requêtes échangées.

Nous dirons qu'une requête est un *acte* de *langage* [80], accompagné d'un *contenu sémantique*, ayant lieu dans un *contexte* donné [68]. L'information échangée entre deux agents A_1 et A_2 interagissant, se distribue sur ces trois éléments de la requête. Les actes de langage sont des informations dont la sémantique est partagée par toutes les entités dans InterViews. Le contexte dépend de l'interaction, et il est perçu de façon différente, mais son existence est un fait objectif et partagé.

Enfin, d'après la définition donnée aux requêtes dialogiques, un agent A_1 ne peut formuler des requêtes $Q_{1\to 2}$ à un agent A_2, que dans le vocabulaire qu'il connaît, le sien. Ce postulat est très important et nous l'utiliserons pour la formalisation du problème de l'hétérogénéité sémantique au chapitre III, mais aussi pour sa gestion au chapitre VIII.

I.2.4.2 Réponse associée à la requête

Soit une requête $Q_{1\to 2}$ formulée par un agent dialogique A_1 vers un agent dialogique A_2. Elle est envoyée par A_1 dans *l'intention* de provoquer une *réponse* de la part de A_2. Cette réponse peut se traduire par :

- Un *fonctionnement* déclenché chez A_2 : c'est-à-dire une action que A_2 doit exécuter ;
- Une réponse sous forme de *requête* : véhiculant à son tour une contenu sémantique, avec potentiel de sens, et se justifiant par rapport à la suite d'échanges de requêtes (contenus, effets et ordre d'émission des requêtes) qui constitue l'interaction dialogique globale ;
- Ou *rien* : ce qui est, en soi, une information.

Que ce soit pour déclencher un fonctionnement, ou une requête, A_2 *doit accéder* à une information dans sa base de connaissances pour produire une action ou délivrer un contenu. On dit que c'est A_1 qui déclanche l'accès à cette information chez A_2.

Le déroulement de l'interaction entre composants logiciels à l'aide d'un langage de requêtes respecte habituellement un *protocole*.

I.2.4.3 Accès à une connaissance lors de la réception d'une requête

Étant donné une information nécessaire à A_2 pour construire la réponse $R_{2\to 1}$ attendue à une requête donnée, le problème est maintenant pour A_2 *d'accéder* à cette information. Ce problème d'accès à l'information se décompose en deux sous-tâches : d'abord caractériser l'information à rechercher, ensuite, à partir de cette caractérisation, y accéder. Nous verrons qu'après cette tâche, commence le vrai problème dans le cas où l'information ainsi caractérisée est inaccessible, inexistante, ou intutilisable.

Pour l'instant, une requête se ramène à un accès dans la représentation de l'autre. La caractérisation puis la recherche de cette information se fait à l'aide d'une *adresse*. Une adresse, prise au sens large peut être sous l'une des formes suivantes :

1. Adresse physique dans une représentation : c'est un *chemin*, à la manière des chemins d'accès à un fichier dans un système de gestion de fichiers qui les stocke dans une arborescence (exemple : le système Unix) ;
2. Le nom de l'information, son étiquette : c'est un *symbole* ;
3. La description de l'information, sa structure : c'est son *contenu*.

Ces trois modes de désignation, d'accès à une information chez l'autre vont être plus ou moins efficaces en fonction du type d'information recherchée, mais aussi pour des agents, en fonction des connaissances qui sont partagées :

- Partager des adresses, c'est partager des patterns particuliers dont la racine et même le début du chemin peuvent être les mêmes. Cela suppose un système de référence commun ;
- Partager des éléments de *contenu*, c'est partager des *symboles avec leur sens*. En effet, il ne suffit pas de partager des symboles, s'ils ne désignent pas les mêmes objets.

Dans tous les cas, nous ramenons l'ensemble des connaissances partagées à un ensemble de patterns de représentation, étant donné que le formalisme de représentation dans VDL est lui-même de nature arborescente.

I.2.5 Types d'interaction dialogiques

Deux types d'interactions sont à considérer dans InterViews :

1. L'interaction Humain-Vue : Ce type d'interaction est à la base de l'introduction d'agents assistants (*help*) dans l'interface, notamment dans des systèmes d'EIAO. Un type particulier de help, est celui appelé SH (*shifted* help) car il décrit des cas d'interaction avec *décalage* sémantique lors d'une demande d'aide. En effet, le plupart des systèmes d'aide reviennent à une consultation d'index, et nécessitent des requêtes directes de la part de l'humain. C'est l'humain qui s'adapte au formalisme proposé par l'agent assistant. Le fait de traiter des nuances dues à la langue naturelle dans l'interaction, revient à obliger l'assistant à posséder un module de traitement lui permettant de s'adapter à l'utilisateur. En allant encore plus loin, une symétrisation des rôles d'aide entre un humain et un agent d'interface peuvent nécessiter une véritable gestion d'un dialogue inspiré des dialogues en langue naturelle. La gestion de l'interaction humain-vue passe par l'introduction du dialogisme. Dans cette étude nous ne nous penchons pas sur le problème du traitement syntaxique et sémantique des phrases exprimées en langue naturelle (module séparé), mais à la gestion du dialogisme dans le language pivot VDL, et aux problèmes de décalage sémantique qui en découlent.
2. L'interaction Vue-Vue : Ce type d'interaction est totalement symétrique. C'est un renouvellement du problème des extraterrestres, introduit par Minsky [65], car les vues possèdent une autre corporalité et peuvent partager des choses différentes par rapports aux humains. Ainsi, cette interaction donne aussi obligatoirement lieu au problème des agents sémantiquement hétérogènes, puisque dans InterViews tout est exprimé en VDL. Cependant, un décalage peut se glisser au moment de la conception (fonction, vocabulaire, structure, définition, but, connaissances, ...), et il en découle également une hétérogénéité au niveau sémantique.

I.3 Hétérogénéité sémantique entre agents dialogiques

I.3.1 Origines de l'hétérogénéité sémantique dans InterViews

Les agents dialogiques présentés et définis dans la section précédente modélisent les composants logiciels réels en interaction sur la plate-forme InterViews. Les applications dont il est question peuvent être définies :

- par des utilisateurs différents,
- dans des buts différents,
- avec des vocabulaires différents,
- à des instants différents,
- etc...

A cause de toutes ces sources de *différences*, il émerge obligatoirement une *hétérogénéité,* au niveau *sémantique* entre les agents dialogiques lorsqu'ils veulent *interagir* avec des requêtes *dialogiques*. En effet, une interaction dialogique est définie comme une interaction à un haut niveau sémantique.

Dans *World and Object*, Quine [73] présente le problème de *Gavagaï*. Ce problème résulte de la *rencontre* d'un éthnologue avec une tribu dont le langage lui est inconnu. Si un natif dit "Gavagaï" tout en pointant "un lapin blanc qui court", il n'y a pas de moyen immédiat pour l'ethnologue de savoir ce que Gavagaï veut dire. En effet, cela pourrait vouloir dire "blanc", "lapin", "courir", "partir", ou "blanc lapin", ou "faim", etc. Notre étude se concentre sur la gestion de ce type d'hétérogénéité, due au phénomène de potentiel de sens : c'est ce que nous appelons *l'hétérogénéité sémantique.*

C'est le même problème qui est en fait abordé par Minsky lorsqu'il s'intéresse à la recontre extra-terrestre [65], que nous avons déjà mentionnée plus haut. Il va plus loin que Quine, dans le sens où il propose que cette hétérogénéité sémantique soit surmontée en se ramenant à ce que nous aurions en commun avec les extra-terrestres. Il ne parle pas de connaissances communes mais d'expériences communes, puisque nous partagerions le même environnement et un certain nombre de perceptions communes de celui-ci. En effet, il ne peut y avoir une interaction (au sens physique) que si nous partageons ces éléments.

La gestion du suivi de conversation peut être un élément utile à la résolution de l'hétérogénéité sémantique, comme par exemple dans le jeu du "guessing game" du projet Talking Heads [82].

I.3.2 Degrés de décalage sémantique dans une interaction

Un des objectifs de l'interaction est de réduire *l'écart* cognitif initial entre les interlocuteurs et d'instaurer une convergence sémantique et référentielle [98]. Dans le problème des agents dialogiques, l'écart dont il s'agit est ramené à un décalage au niveau sémantique. La figure I.2 montre une échelle de décalages sémantiques, rapportée au segment continu [0,1].

Le décalage nul : Dans le cas de l'interaction vue-vue, il correspond à une situation idéale d'interaction avec soi-même. Dans le cas d'une interaction homme-vue, il correspond à

FIG. I.2 – Décalages sémantiques entre deux agents dialogiques ramenée au segment continu [0,1]. Si le décalage est nul, ce qui est représenté par un décalage égal à 0, la gestion de l'hétérogénéité sémantique n'est pas nécessaire. Si le décalage est total, ce qui est représenté par un décalage égal à 1, la gestion de l'hétérogénéité sémantique est impossible.

une requête directe non ambigüe, de type *"straight* query". Il n'est pas *nécessaire* de gérer les décalages dans ce cas. Dès que l'on s'éloigne du cas des requêtes *straight*, on se retrouve dans un cas d'hétérogénéité sémantique.

Le décalage maximal : Il correspond au problème de l'hétérogénéité au sens *fort*. Dans le cas du problème des extra-terrestres, cela correspondrait à interagir avec des êtres à deux dimensions par exemple : il n'est pas *possible* de gérer les décalages sémantiques qui en découlent, car aucune expérience commune ne peut servir de *passerelle*. Ainsi, dès que l'on s'écarte trop du décalage nul, on se retrouve dans un problème intraitable.

Les seuls cas interessants dans cette étude se situent donc entre les deux bornes extrêmes. Il s'agit maintenant de les *caractériser*, les *classer* et les *résoudre* en proposant des outils effectifs que l'on inclura dans le modèle de requêtes d'InterViews.

Gérer les décalages, c'est alors rendre *possible* une interaction, avec partage et échange effectif d'information entre deux entités en position de décalage sémantique l'une par rapport à l'autre. Cette gestion n'est possible que s'il existe une ensemble non vide d'éléments *communs* entre les entités, ce que nous avons appelé une *passerelle*.

Nous allons considérer le problème de la gestion des décalages dans InterViews par le biais de la définition d'un modèle de *requêtes* pour VQL. Ainsi, les agents partagent au moins VQL et donc le mode d'interaction par requêtes. Des éléments différents pourraient être partagés par les agents et conduiraient à des interactions de type différent : la syntaxe, le corps, l'environnement direct (par exemple utilisateur ayant défini la vue, ou application de la vue, ...), vocabulaires, fonctionnalités, types de motivations (buts ou moteurs), existence d'une curiosité, etc.

Traiter le problème de l'hétérogénéité sémantique dans l'échange de requêtes dialogiques entre agents, correspond à ne considérer l'hétérogénéité sémantique entre agents dialogique que au niveau des *mots*. De plus, si l'on fait la distinction entre mots fonctionnels et mots à contenu, les termes hétérogènes dont il est question ici ne correspondent qu'aux mots à *contenu*. En effet, les mots fonctionnels sont donnés par le langage VQL et sont donc partagés.

Nous allons étudier les phénomènes d'hétérogénéité sémantique au niveau des définitions de sens pour les termes connus par les agents, c'est-à-dire entre les bases de connaissances ontologiques. Ainsi, pour traiter les problèmes d'hétérogénéité sémantique entre agents dialogiques, nous proposons de les considérer au niveau de la *représentation* des *bases* de *connaissances* des agents [101].

I.3.3 Les types d'hétérogénéité sémantique

Les sortes de *différences* entre ontologies ont déjà été analysées par Campbell [19] qui distingue entre les différences de :

Nommage : hétérogénéité provenant de l'utilisation d'un même nom pour des objets différents ou de noms différents pour un même objet. Ce problème est recouvert par la question plus générale de la *référence*. Nous parlerons maintenant d'hétérogénéité sémantique de type référentielle.

Taxonomie : hétérogénéité provenant de la *structuration* en hiérarchie du domaine. Nous parlerons désormais d'hétérogénéité sémantique de type taxonomique. Rappelons que d'après la définition donnée aux requêtes dialogiques, un agent A_1 ne peut formuler des requêtes $Q_{1 \to 2}$ que dans le vocabulaire qu'elle connaît, le sien ; et que le vocabulaire connu par un agent A_i est défini dans sa terminologie.

Structure : hétérogénéité provenant de la *syntaxe*. Nous ne traiterons pas ce type d'hétérogénéité car nous utilisons le même langage, VDL, pour décrire et définir les connaissances de tous les agents, ainsi que pour exprimer les requêtes et les réponses (en VQL, qui est une sous-partie de VDL, voir I.1.2.3).

Expressivité : hétérogénéité provenant du langage de *représentation*. Comme pour la structure, nous écartons ce type d'hétérogénéité grâce à l'utilisation d'un même langage de formalisation, VDL, pour les connaissances de l'agent, et VQL pour les requêtes échangées.

I.3.4 Proposition pour la gestion d'une interaction hétérogène

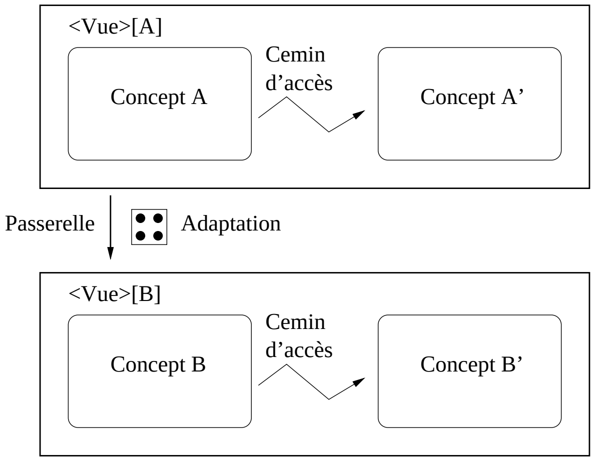
Lors d'une interaction dialogique entre deux agents dans InterViews, la vue A envoie une requête écrite en VQL à la vue B qui peut contenir ou pas des *termes* hétérogènes. La vue B pourra interpréter ces termes hétérogènes en fonction de :

1. La connaissance autour de ce terme hétérogène contenue dans la requête, provenant de A et trahissant de la connaissance à propos de A ;
2. Ses propres connaissances qui lui permettent de transformer le terme hétérogène en fonction des autres termes connus dans sa base.

Nous appellerons l'ensemble d'éléments "communs", présents dans la base de connaissances de la vue B, et utilisables par celle-ci pour interpréter la requête reçue de A, la *passerelle* entre les deux représentations. Pour *accéder* à la connaissance de l'autre vue, celle qui produit la requête, B doit utiliser cette connaissance commune et donc "emprunter" cette passerelle.

Précisons ici ce que nous entendons par *connaissance commune*. Il est évident qu'il n'y a pas d'agent superviseur qui peut assurer à la vue A et à la vue B qu'ils ont *effectivement* des connaissances communes. On ne peut être *sûr* d'avoir des connaissances communes *exactement*, que si on connait les symboles de connaissances (un modèle syntaxique) et les entités que ces symboles *désignent* dans un monde *effectif* (une interprétation). Or, la vue B n'a pas accès à la définition des connaissances de A, elle ne peut donc pas *savoir*, sans l'aide d'un superviseur, ce que désigne exactement chaque symbole utilisé dans une requête. Un tel superviseur étant exclu

FIG. I.3 – Schéma représentant le carré analogique : la vue A envoie une requête à la vue B qui peut contenir des termes hétérogènes. La vue B pourra interpréter ces termes hétérogènes en fonction de : la connaissance autour de ce terme hétérogène contenue dans la requête, provenant de A et trahissant de la connaissance à propos de A , ses propres connaissances qui lui permettent de transformer le terme hétérogène en fonction des autres termes connus dans sa base.



par hypothèse dans InterViews, B n'aura jamais la *garantie* de partager des connaissances avec A. Cependant, B aura un point de vue *subjectif* sur les symboles contenus dans la requête issue de A : si elle contient des symboles définis dans sa propre base de connaissances, elle *comprendra* ces symboles par rapport à leur sémantique *locale*, "chez elle". Ce n'est que si de telles interactions mènent à des succès que les vues pourront *supposer* partager des connaissances sans jamais en avoir une garantie. Ce type de partage sans garantie est le même que nous retrouvons dans les interactions humaines.

La désignation d'une connaissance dans une requête, un accès à une information, n'aura donc de sens pour B que si elle peut reformuler la requête reçue en fonction de la passerelle qui la relie à la vue A. La figure I.3 schématise ce processus, où des concepts sont définis dans les vues A et B les uns *par rapport* aux autres (relations) et sont donc liés par des "chemins d'accès". L'existence d'une passerelle permet l'utilisation de certains mots par A (dans ses requêtes envers B), et sera utile à B pour relier certaines connaissances inconnues issues de A à certaines connaissances connues chez elle, définies dans B. Le fait de reformuler une requête, en une autre, *le long* d'une passerelle, revient à *adapter*, déformer des *structures* par un mécanisme de résolution *d'analogies*.

Nous ne donnons ici que l'intuition de ma démarche que nous proposons d'adopter pour gérer le problème de l'hétérogénéité sémantique entre agents dialogiques. En effet, beaucoup d'hypothèses implicites sont déjà présentes dans cette présentation, comme par exemple l'utilisation d'une métaphore spatiale pour représenter des structures de connaissances, et les liens entre ces connaissances au sein d'une vue. Nous verrons dans la deuxième partie de cette thèse toutes ces notions en détail, et nous donnerons des arguments en faveur de l'utilisation de la métaphore spatiale et de la notion d'adaptation pour traiter ce problème précis.

I.4 Conclusions du chapitre

Dans ce chapitre, nous avons exposé le cadre du projet InterViews dans lequel s'insère notre travail de thèse. Pour cela, nous avons introduit les axes de recherche du projet et décrit scientifiquement nos objectifs et objets d'étude. Nous avons ensuite introduit et défini la notion centrale d'agent dialogique, ainsi que le dialogisme lui-même.

Dans ce cadre, nous avons présenté les conditions de l'émergence d'une hétérogénéité sémantique entre ces agents dialogiques et défini ce que nous entendons par hétérogénéité sémantique entre agents dialogiques.

Le problème que nous traitons dans ce document est donc ramené à la gestion d'une interaction sémantiquement hétérogène entre *deux* agents selon les contraintes imposées par le projet InterViews.

Pour gérer ce problème, nous proposons tout d'abord de le formaliser afin d'en dégager les caractéristiques, et proposer une solution adaptée. La formalisation que nous proposons se base sur un formalisme de représentation des connaissances très répandu dans le domaine des bases de connaissances en Intelligence Artificielle : ce sont les logiques de descriptions. Nous introduisons ce formalisme au chapitre suivant, puis nous l'utilisons pour formaliser l'hétérogénéité sémantique entre agents dialogiques dans le chapitre III.

Chapitre II

Les Logiques de descriptions comme formalisme de représentation des connaissances

Introduction

Nous avons présenté la problématique de l'hétérogénéité sémantique entre agents dialogiques dans le cadre du projet InterViews. Afin de traiter ce problème et d'en élaborer des solutions algorithmiques, nous proposons de fonder notre étude sur une modélisation du problème en logique des descriptions.

En effet, un système basé sur la représentation des connaissances doit distinguer entre différents types de connaissances. Un de ces types est la *connaissance terminologique*, c'est-à-dire à propos du *vocabulaire* (technique en général, mais ce n'est pas obligatoire). Ce type de connaissance est représentable avec des formalismes dits "centrés-objets". Ce genre de formalisme pour la représentation des connaissances a débuté avec l'introduction des *Réseaux Sémantiques* (RS) par Quillian en 1967 [72], puis par la proposition des *Frames* par Minsky en 1975 [64]. Le point commun essentiel de ces formalismes est que la connaissance terminologique y est exprimée uniquement en termes d'introduction de définitions, c'est ce qu'on appelle un mode *définitoire*. De plus, la connaissance est définie par des expressions qui la *structurent* [14]. Brachman a ensuite proposé les *Structural Inheritance Networks* (SIN) en 1977 [15], qui donneront plus tard les *logiques de description*. Le système le plus connu implémentant le formalisme SIN, s'appelle KL-ONE [16] ; on considère aujourd'hui les logiques de description comme les descendants de ce système.

Plan du chapitre

Ce chapitre se veut une introduction aux bases de la logique de descriptions. Il présente, d'une part, l'origine des logiques de descriptions, afin de motiver leur utilisation en tant que formalisme de représentation des connaissances dans le problème particulier qui nous intéresse ; et d'autre part, la syntaxe pour un langage de description simple (appelé TF). Notons que dans la suite du document, pour la formalisation du problème d'hétérogénéité, nous utiliserons un autre langage de description (appelé AL) simple également, mais plus courant dans la communauté, qui présente quelques différences par rapport à TF et que nous introduisons en fin de chapitre. Enfin, nous précisons les éléments de *sémantique* dénotationnelle propres aux logiques de descriptions, qui en font justement un formalisme très approprié pour les problémes d'hétérogénéité *sémantique*.

Enfin nous montrons comment les chercheurs en Intelligence Artificielle utilisent le formalisme des logiques de descriptions pour modéliser des bases de connaissances puisque c'est l'aspect qui nous intéressera le plus dans la modélisation de nos agents dialogiques dans le chapitre suivant.

II.1 Origines et introduction

II.1.1 Les réseaux sémantiques

II.1.1.1 Idée de base : réseaux de mots

Quillian a proposé les réseaux sémantiques pour donner un modèle de la *mémoire* humaine [27]. En effet, il s'est intéressé au stockage de *mots* avec leur sens en mémoire, et a proposé leur modélisation par un graphe à la suite de tests portant sur les temps de réponse à des tâches terminologiques sur des humains. Ces tests mesuraient les temps de réponse à des questions portant sur des définitions de mots usuels. Les temps de réponse étaient interprétés par Quillian comme des *distances sémantiques* entre les mots dans une représentation de la taxonimie qui serait *spatiale*.

L'idée a donc été d'introduire des *graphes* pour traduire cette spatialité de la représentation : les *noeuds* du graphe modélisent les mots, et les *arcs* reliant ces noeuds modélisent les *relations* (liens) entre les concepts. Les *inférences* opérées dessus sont alors modélisées par des propagations d'activation sur ce réseau (cette propagation se fait à une certaine vitesse, et on comprend que la propagation vers des concepts éloignés dans un graphe est plus longue et se traduit donc par un temps de réponse plus long). Un concept est ainsi défini par les concepts *atteignables* depuis son noeud. Les réseaux sémantiques mettent en oeuvre l'idée intuitive d 'espace des concepts.

II.1.1.2 Problèmes posés par les réseaux sémantiques

Les désavantages de ce formalisme se résument essentiellement dans la *confusion* de connaissances de divers types, représentées par les mêmes entités. En effet, les arcs ne différencient pas deux types de liens pourtant différents :

a) les liens *assertionnels* : par exemple le fait que "`Marie est une Femme`" est une assertion, qui se traduit par un lien entre le concept individuel `Marie` et le concept générique `Femme`, ce lien correspond au symbole d'appartenance $x \in E$.

b) des liens *structurels* : ceux-ci devraient être distingués des liens assertionnels, mais il faudrait en plus lever la confusion entre les différents types de liens structurels :

1. structure d'héritage de propriétés entre deux concepts, par exemple le fait que le concept `Mammifère` soit relié au concept `Animal`, lui donne en héritage des propriétés comme la `respiration`, ce lien se traduit par le symbole d'inclusion $A \subseteq E$
2. structure de composition : en termes de *parties* par exemple, le fait qu'un `Humain`, `Marie` en particulier possède une `tête`, ce lien se traduit par la relation $Part\ of$ (partie de, voir [75]).
3. structure d'équivalence,
4. structure de contraste,
5. structure de particularisation,
6. structure de relations spatiales,
7. structure de succession, entre des ordinaux par exemple,
8. structure de fonction (voir une liste dans [75] pages 210-212) ;

Par ailleurs, les noeuds ne différencient pas :

a) les concepts de *classe* : comme celui de `Femme` dans l'exemple précédent

b) et les *individus* : comme `Marie` dans l'exemple précédent ;

la confusion entre les concepts et les individus au niveau des noeuds est un problème dual de la confusion des différents liens au niveau des arcs. Les liens d'appartenance d'une sous-classe à une classe ou d'un individu à une classe sont tous les deux étiquetés par "`sorte-de`". Ainsi, `Marie` est une `sorte-de Humain`, et `Brun` est un sous-concept de `Humain` , donc une `sorte-de Humain` également.

De plus, les valeurs ne différencient pas :

a) les valeurs *possibles* : pour un attribut, comme l'`âge` de quelqu'un, compris entre `0` et `100` par exemple (notion de domaine de définition),

b) et les valeurs *effectives* : `Marie` a effectivement `27 ans`.

Enfin, il est difficile de savoir si l'existence d'un noeud dans le réseau correspond avec *l'*existence *effective* du concept correspondant dans le monde modélisé. En effet, le fait de représenter dans un RS, un noeud `Marie`, lié au noeud `Femme`, veut-il dire que :

- `Marie` existe effectivement en tant qu'individu du monde représenté,
- ou bien que l'on représente un monde où tout individu qui s'appelerait "Marie" est obligatoirement une `femme` ?

Des exemples plus compliqués de problèmes dans les RS sont consultables dans le livre de G. Sabah [75] (en particulier, confusions entre classe et instances de `voiture`, entre liens `possède` et `sorte-de`, menant à une situation ambigüe où deux individus `Humains` différents possèdent la même `voiture`) ; en effet, une combinaison de chacune de ces confusion mène à des situations très ambigües.

La solution à ces problèmes est la définition d'une *sémantique* formelle. Une manière classique de donner la sémantique formelle d'un formalisme de représentation des connaissances est de le comparer avec la Logique du Premier Ordre (LPO). C'est ce qu'ont proposé d'autres chercheurs avec par exemple les *Réseaux Partitionnés* (Hendrix 1979 par exemple [47]). Mais cette façon de définir la sémantique des réseaux sémantiques les réduit à une simple *variante* notationnelle de la LPO.

II.1.2 Les réseaux SIN

II.1.2.1 Idée de base : donner une vraie sémantique aux réseaux sémantiques

Brachman (1977) essaie de proposer un formalisme de représentation de terminologies [14], [15], qui ne présente plus les inconvénients cités plus haut à propos des réseaux sémantiques. Pour cela, il part de la constatation qu'ils sont utilisés à différents niveaux :

- *implémentatoire* : en tant que structures de données,
- *logique* : comme variante notationnelle de LPO,
- *conceptuel* : avec des liens conceptuels, dans des graphes de dépendance conceptuels,
- *linguistique* : avec des mots pour les primitives (c'est très dépendant du langage).

Brachman propose alors l'introduction d'un niveau supplémentaire entre le niveau logique et le conceptuel : le niveau *épistémologique*. Ce niveau comporterait des description *intensionnelles* de concepts ; ainsi les concepts ne sont plus vus comme des blocs atomiques mais comme des *définitions complexes*. Cela a pour conséquence que l'héritage entre concepts ne porte pas seulement sur les propriétés de ceux-ci mais également sur leur *structure* (interne). Le système formel introduit par Brachman est appelé *semantical inheritance network* (SIN).

II.1.2.2 Les concepts dans les réseaux SIN

Un réseau SIN représente lui aussi une terminologie d'un domaine. Cette représentation est structurée et définitoire et représente les concepts de la terminologie comme des noeuds, reliés entre eux par des relations. Cependant, dans un SIN, la distinction entre concepts *génériques* et *individuels* est clairement introduite :

- Les concepts *génériques* peuvent être primitifs (préexistants, on les note avec un $*$ devant, par exemple $*Man$) ou définis par l'utilisateur (en fonction des concepts primitifs ou définis précédemment).
- Les *individus* sont considérés comme des sous-classes réduites à un singleton.

Cependant, chaque individu introduit (défini) ne correspond pas forcément à un objet réel de l'univers décrit. Pour parler explicitement de l'existence d'individus, et asserter à leur sujet, il faut ajouter une machinerie extensionnelle au niveau de représentation épistémologique (qui lui, est intensionnel). Ainsi, on dispose d'une représentation intensionnelle des concepts d'une terminologie ; et d'un système autorisant des assertions à propos d'individus membres des concepts (pris comme des classes) de la terminologie représentée.

Le sens d'un concept est déduit des super-concepts auxquels il est attaché (et dont il hérite) et des informations attachées à lui-même (définitoires). En particulier, il ne dépend pas des sous-concepts ou des individus qui en héritent (ce choix peut être soumis à discussion !).

II.1.2.3 Les relations : les rôles

En plus des concepts, Brachman introduit la notion de *rôles*. Ces rôles sont des relations binaires *potentielles* entre les individus de différentes classes.

Soit T, une terminologie, soient C_1, C_2 deux concepts de T ; un rôle r reliant ces deux concepts est une relation binaire $r(C_1, C_2)$ entre deux classes. Ce rôle complète la définition du concept

C_1 en précisant que toute instance x de C_1 devra entretenir une relation r avec un individu y de la classe C_2. L'individu y est appelé un *role-filler* pour le rôle r dans la définition de C_1.

Prenons un exemple pour préciser cette dernière notion. Soit :

$$C_1 = Equipe \text{ et } C_2 = Humain$$

supposons que nous ayons introduit une *Equipe* comme étant un $Ensemble$ (qui est par exemple un concept prédéfini), on veut alors ajouter à cette définition le fait qu'une *Equipe* est constituée de membres. Pour cela, on introduit le rôle $membre$ qui mettra en relation le concept *Equipe* avec le concept $Humain$. Ainsi, toute instance d'*Equipe* (par exemple l'équipe E_1) devra entretenir la relation $membre$ avec un individu de type $Humain$ (par exemple joe).

- Lorsque l'on contraint la relation $membre$ à opérer sur un humain, on dit que l'on a effectué une *restriction* de *valeur*. En effet, le restriction porte sur le “type”, donc la valeur de l'individu y, le role-filler.
- Lorsque l'on contraint le nombre d'individus du concept C_1 qui vont vérifier cette relation, on dit que l'on a effectué une *restriction* de *nombre*. Cette restriction peut être un maximum (max) ou un minimum (min).

On peut également opérer une restriction particulière qui s'appelle *différenciation*. Une différenciation s'opère entre deux rôles et permet de différencier un rôle par rapport à un autre, c'est-à-dire de *profiter* de la définition d'un autre rôle pour définir le nouveau rôle en *différence*. Par exemple, on peut introduire le rôle $chef$ dans la définition de *Equipe* et le définir comme étant un membre avec une propriété spéciale “en plus”. Cela revient à établir un lien *is-a* (voir définition dans [75]) entre le rôle $membre$ et le rôle $chef$. Un chef sera alors automatiquement un $Humain$ mais aura en plus les propriétés du rôle $chef$ données en différence (être une $Femme$ par exemple).

Les services de *raisonnement* possibles sur une telle structure de représentation des connaissances sont restreints : les connaissances représentées étant terminologiques, l'essentiel des opérations se résume aux activités d'un *classifieur*.

II.1.2.4 Le système KL-ONE

Le système KL-ONE [16], correspond à la première implémentation des SIN (Brachman 1978). Il a connu un grand succès et a été utilisé notamment comme classifieur pour le développement de plusieurs systèmes d'Intelligence Artificielle (en particulier de la Langue Naturelle). Le succès de ce système peut s'expliquer par les points suivants :

- KL-ONE ne tente pas de répondre à toutes les exigences du domaine des représentations des connaissances : il adresse seulement la définition de concepts ;
- il reste proche des réseaux sémantiques et donc d'un modèle de mémoire ;
- les outils de raisonnement se réduisent à ceux d'un classifieur ;
- ces outils sont intégrés au système et donc utilisables directement ;
- il est muni d'une sémantique puisqu'il a été comparé à la LPO.

En résumé, KL-ONE est à la hauteur de ses ambitions, qui sont bien *délimitées*. Afin d'augmenter le pouvoir de raisonnement des systèmes basés sur les SIN, il est nécessaire d'élargir les capacités d'expression des connaissances *assertionnelles*. La solution proposée est d'introduire en plus des connaissances terminologiques, des connaissances *assertionnelles*, représentables avec un langage particulier (cet aspect est très superficiellement traité dans KL-ONE).

II.1.2.5 Les SIN et les graphes conceptuels

Bien qu'introduits plus tard par Sowa [81], et n'ayant pas participé au développement des logiques de descriptions, les Graphes Conceptuels présentent beaucoup de similarités avec les SIN et méritent qu'on s'y intéresse un peu.

L'inspiration de Sowa vient de l'étude de la *perception* : elle consiste pour lui en l'élaboration d'un *modèle* à partir d'images *partielles* (les percepts) que l'on *combine*. Le modèle de travail est formalisé par un *graphe conceptuel*, qui utilise des graphes *canoniques* (c'est-à-dire, de base) issus de la mémoire à long terme pour sa construction. Précisons que ce formalisme, contrairement à ceux dont il est question dans ce document, n'est pas exclusivement terminologique, c'est un formalisme qui se veut *général*. D'ailleurs Sowa prétend que tout autre formalisme peut être réécrit sous la forme de graphes conceptuels. Les connaissances de la mémoire à long terme qui sont utilisées comprennent entre autres bien sûr un réseau sémantique pour les connaissances terminologiques. C'est pourquoi certains aspects des GC vont êtres apparentables aux RS et aux SIN.

Plus formellement, les connaissances terminologiques sont représentées dans un *treillis de types*, dont les deux extrémités sont le concept *universel* $\top$ et le concept *absurde* $\perp$. La relation d'ordre partiel de ce treillis représente la relation *d'héritage* des réseaux sémantiques. Cependant, à chaque définition intensionnelle des concepts dans ce treillis, est ajoutée une représentation en *extension*. Ainsi, on attache à chaque noeud n, un ensemble E_n représentant tous les *référents possibles* pour le concept n dans le treillis T. Ainsi, un concept n est défini par son intention (son type) t_n et son extension (ses référents) E_n, et on le note $n(t, E_n)$. C'est par ce biais que sont *distingués* les concepts *génériques* des *individus*. En effet, un concept générique aura pour extension une *variable*, marquée par un $*$, et un individu aura pour extension un élément, marqué par le signe $\#$.

D'autres notations ont été introduites pour représenter des phrases (assertions) mais elles posent des problèmes que nous n'aborderons pas dans le cadre des représentations terminologiques. Notons simplement que pour représenter des assertions sur les concepts du treillis, les graphes disposent aussi de relations n-aires. Ces relations sont représentées avec des graphes, reliés au nombre de concepts correspondant à leur arité et du type imposé par la relation (notion proche de la restriction dans les SIN). Ces graphes représentant les relations sont des éléments de base, appelés graphes simples. Enfin, signalons que l'expressivité des GC est équivalente à la LPO.

II.1.3 Les domaines de recherche en logiques de descriptions

II.1.3.1 Plusieurs langages

De nos jours, plusieurs variations sur le thème de la logique de descriptions ont été proposées. Ainsi, on parle plutôt des *logiques* de *descriptions* (LD) ou de *langages terminologiques*. Ainsi, les LD sont aujourd'hui un outil commun de représentation des connaissances (KR) en IA, pour exprimer des *taxonomies* (appelées aussi terminologies) et opérer des inférences dessus. Les LD fournissent un langage pour construire et traiter ces terminologies, et sont pour cette raison parfois aussi appelées des *langages conceptuels*. Ce formalisme est muni d'une *sémantique dénotationnelle* définie dans la théorie des ensembles.

II.1.3.2 Intégrer du raisonnement

En analysant les *descriptions* des concepts, le système organise ces derniers en *hiérarchies*. Cette hiérarchie est la base pour effectuer des services de *raisonnement*. Les raisonnements sont effectués par l'intermédiaire de *requêtes* sur la base de connaissances. *L'efficacité* du raisonnement a souvent été le premier argument pour motiver le choix de tels formalismes de représentation des connaisances. En effet, le fait de formaliser le langage facilite le calcul de la complexité des raisonnements d'inférence, ce qui a donc beaucoup été étudié [29] et amélioré.

II.2 Représentation terminologique

Le formalisme introduit dans les SIN, est celui sur lequel s'appuie KL-ONE et ses descendants comme KRYPTON [17] ou NIKL [79]. D'autres formalismes ont aussi adopté les idées clé de KL-ONE comme BACK [99], LOOM [61], CLASSIC [10]. Tous ces systèmes utilisent des formalismes de représentation de terminologies qui peuvent paraître différents en surface mais qui reposent tous sur les mêmes bases.

Une des principales caractéristique des LD est de *séparer* les connaissances *terminologiques* (concernant les concepts) des connaissances *assertionnelles* (concernant les individus), cette distinction présente des avantages du point de vue de la représentation des connaissances et de la complexité.

Le langage terminologique présenté dans ce document est appelé TF, il est assez proche de celui introduit par Brachman et Levesque en 1984, mais utilise un peu plus de constructeurs, il est tiré de la thèse de Nebel [71]. Nous verrons ce que cela veut dire exactement dans la section II.4 où nous parlerons plus amplement de tous les langages terminologiques, en les comparant entre eux et avec TF.

II.2.1 Syntaxe de TF

Le langage terminologique permet de définir une *terminologie*. Cette terminologie est la représentation structurée et déclarative d'un vocabulaire d'un domaine. Une terminologie est définie comme une séquence de *déclarations*, donc d'introduction de définitions, de *termes* ou de *restrictions* sur des définitions déjà déclarées :

$$\langle terminologie \rangle ::= \{\langle termeIntroduction \rangle | \langle restriction \rangle\}^*$$

Où le symbole $*$ signifie une répétition de 0 ou *plusieurs* fois ce terme. En effet à tout instant, on peut soit introduire un nouveau terme de vocabulaire, soit restreindre les caractéristiques d'un terme existant déjà dans la terminologie, pour le *spécialiser* et créer ainsi un nouveau terme.

Un **terme** est introduit *une et une seule fois*, et il peut être l'introduction soit d'un *rôle,* soit d'un *concept* :

$$\langle termeIntroduction \rangle ::= \langle conceptIntroduction \rangle | \langle roleIntroduction \rangle$$

Ainsi, la structure d'une terminologie n'est pas une liste ni un multi-ensemble, c'est un *ensemble de définitions.*

II.2.1.1 Les concepts

Un **concept** exprime la notion de *classe* en tant que groupe d'individus, il correspond à un prédicat *unaire* en LPO. On définit de nouveaux concepts par une introduction de concept définie comme suit dans TF :

$$\langle conceptIntroduction\rangle ::= \langle atomicConcept\rangle \doteq \langle concept\rangle |$$
$$\langle atomicConcept\rangle \leq \langle concept\rangle | \langle atomicConcept\rangle \leq ANYTHING$$

Ainsi, on note que l'on peut introduire des concepts de trois manières :

- En définissant ($\doteq$ pour la *définition*) un nouveau concept atomique $\langle atomicConcept\rangle$ avec un détail de ses caractéristiques dans $\langle concept\rangle$;
- En introduisant un nouveau concept atomique $\langle atomicConcept\rangle$ et en le positionnant par rapport au reste de la terminologie ; c'est-à-dire en précisant de quel concept $\langle concept\rangle$ il est un sous-concept ($\leq$ pour la relation *d'inclusion*) ;
- En introduisant un nouveau concept atomique $\langle atomicConcept\rangle$ et en le positionnant par rapport au plus grand concept possible de la terminologie ($ANYTHING$) ;

Introduire un concept atomique revient à introduire simplement un nouvel *identifieur* :

$$\langle atomicConcept\rangle ::= \langle identifieur\rangle$$

Pour introduire un concept complexe, c'est-à-dire une combinaison de concepts, on dispose d'opérateurs de construction. Les opérateurs de construction de TF sont :

- AND, pour exprimer le "et" logique, il permet d'aligner plusieurs termes pour définir un concept : on l'utilise comme ça : (AND $description_1$ $description_2$). On peut bien sûr emboîter plusieurs AND et créer une conjonction à plus de 2 termes. Par exemple une moto est un objet-à-deux-roues et un engin-à-moteur, cela s'écrit
  $moto ::= (AND(deuxRoues, enginMoteur))$.
- ALL pour exprimer une restriction de *valeur* sur le role-filler d'un rôle. Rappelons qu'un rôle est la représentation d'une relation binaire $R(x, y)$, dans cette notation, le rôle-filler est y (c'est l'argument). Restreindre la valeur de ce filler revient à donner un *type* (un domaine) obligatoire pour y. On utilise le constructeur ALL comme ça : (ALL $role$ *concept*). Cela correspond à dire que tous les concepts reliés au concept défini par le lien $role$ doivent obligatoirement être du type *concept*. Par exemple, si on écrit (ALL $chef$ $Femme$) cela veut dire que tous les concepts reliés par le lien $chef$ doivent être des $Femme$.
- $ATLEAST$ et $ATMOST$, pour exprimer des restrictions de *nombre*, on l'utilise comme ça : ($ATLEAST$ n $role$). Cela veut dire que le concept doit avoir au moins n concepts reliés à lui par le lien $role$. Par exemple on peut écrire ($ATLEAST$ 4 *Leg*) cela veut dire qu'on définit un concept de choses qui ont au minimum 4 *Leg*. C'est pareil pour $ATMOST$, pour "au plus".

Voici la syntaxe générale correspondante pour ces constructeurs :

$$\langle concept\rangle ::= \langle atomicConcept\rangle | (AND\langle concept\rangle^{+}) | (ALL\langle role\rangle\langle concept\rangle) |$$

$$(ATLEAST\langle nombre\rangle\langle role\rangle)|(ATMOST\langle nombre\rangle\langle role\rangle)$$

Enfin, chaque définition de concept peut reposer soit sur un concept prédéfini, soit sur un concept déjà défini par l'utilisateur.

Par ailleurs, on peut introduire la notion de concepts *disjoints* pour deux concepts ayant un même sur-concept :

$$\langle restriction\rangle ::= (DISJOINT\langle atomicConcept\rangle\langle atomicConcept\rangle)$$

Par exemple si on définit :

$Man \leq Human$

$Woman \leq Human$

ensuite, on veut préciser que tout individu Man ne peut être $Woman$ et vice-versa, on écrit :

$(DISJOINT\ Man\ Woman)$

II.2.1.2 Les rôles

Un **rôle** est l'entité qui exprime une relation entre deux membres d'une paire d'individus, c'est donc un prédicat *binaire* en LPO. Rappelons que les LD sont historiquement issues des SIN qui sont des graphes. Ainsi, un rôle est la représentation d'un arc de graphe, il est donc limité à deux extrémités et les rôles sont donc limités à exprimer des relations binaires. Certains chercheurs se sont d'ailleurs penchés sur cette limitation, et il a été prouvé que toute relation n-aire peut s'exprimer à l'aide de relations binaires ; mais il n'a pas été dit que cela soit naturel, agréable, expressif, etc. Le débat reste ouvert dans la communauté.

De même que pour les rôles, un rôle atomique est en fait un simple identifieur, de plus on peut définir un rôle par :

- son introduction en tant que *sous-rôle* d'un autre rôle : $\langle atomicRole\rangle \leq \langle role\rangle$;
- son introduction en tant que sous-rôle du rôle quelconque, qui *subsume* tous les rôles ($ANYROLE$) : $\langle atomicRole\rangle \leq ANYROLE$;
- son *identification* (signe $\doteq$) avec une expression : $\langle atomicRole\rangle \doteq \langle role\rangle$.

II.2.1.3 Les constructeurs

Les opérateurs de construction de concepts cités plus haut, sont appelés aussi plus simplement des *constructeurs*. Ils sont écrits dans cette section sous leur forme concrète. En effet, le fait d'écrire le nom du constructeur en toutes lettres suivi de ses arguments s'appelle la forme concrète, elle est plus pédagogique. On voit souvent une forme plus abstraite qui se base sur des notations de la LPO. On distingue les constructeurs de concepts, et de rôles.

II.2.1.3.a Les constructeurs de concepts

- De *formation* de *concepts* : la conjonction (AND), la restriction de valeur (ALL) la restriction de nombre ($ATLEAST$, $ATMOST$). Mais il en existe d'autres pour construire des concepts, qui ne sont pas utilisés dans TF (par exemple l'égalité,...).
- D'*introduction* de concepts : *définition* d'un concept atomique ($\doteq$) en l'identifiant à une expression, ou bien *spécification* ($\leq$) comme sous-concept d'un autre. Dans les deux cas, on peut utiliser des concepts complexes, construits avec les constructeurs de formation de concepts, pour l'expression de droite dans ces définitions. Un concept apparaissant à gauche de tels symboles est appelé : *primitif* s'il est suivi d'un $\leq$ et *défini* s'il est suivi d'un $\doteq$. Un concept apparaissant à droite de tels symboles est appelé *atomique*.

II.2.1.3.b Les constructeurs de rôles

- De *formation* de *rôles* : par exemple, inverser le sens de la relation, restreindre la plage de valeurs de l'un des deux individus pris dans la relation, composition de rôles. Nous n'avons pas parlé de ces constructeurs car il ne sont pas autorisés dans TF.
- D'*introduction* de rôles : *définition* d'un rôle atomique ($\doteq$) en l'identifiant à une expression, ou bien *spécification* ($\leq$) comme sous-rôle d'un autre. Dans les deux cas, on peut utiliser des rôles complexes, construits avec les constructeurs autorisés.

Dans ce document, nous insistons sur le fait que TF se *restreint*, à des rôles non composés, c'est-à-dire qu'il n'y a aucun constructeur de formation de role autorisé. Le fait d'introduire d'autres opérateurs, comme par exemple, ceux du deuxième type ci-dessus, permet de définir des *extensions* du langage TF : cela débouche sur d'autres logiques de descriptions.

Ainsi, l'ensemble des constructeurs est différent selon la LD considérée, et un langage de description L donné est *complètement défini* par la donnée des constructeurs qu'il autorise dans sa syntaxe (cet aspect est développé dans la section II.4).

II.2.1.4 Syntaxe complète de TF

La syntaxe complète de TF est donnée par la table II.1.

II.2.2 Sémantique dénotationnelle de TF

Le sens *intuitif* d'un formalisme de représentation des connaissances doit être appuyé par un modèle *théorique* de sémantique. Un moyen classique d'y arriver est de donner une traduction du formalisme en termes de *logique* du *premier ordre*. Cependant, une sémantique *directe* est plus utile pour les calculs de complexité par exemple. Nous allons donc donner une sémantique *dénotationnelle* pour le langage TF.

II.2.2.1 Dessiner des ensembles

La sémantique de TF s'appuie sur la notion *d'ensembles*. Ainsi, pour associer un sens intuitif à une terminologie, on commence par se donner un ensemble D qui représente tous les individus effectifs d'un univers (voir figure II.1).

TAB. II.1 – Syntaxe de TF

$$
\begin{array}{rcl}
\langle terminologie\rangle & ::= & \{\langle termeIntroduction\rangle|\langle restriction\rangle\}^* \\
\langle termeIntroduction\rangle & ::= & \langle conceptIntroduction\rangle| \\
 & & \langle roleIntroduction\rangle \\
\langle conceptIntroduction\rangle & ::= & \langle atomicConcept\rangle\doteq \langle concept\rangle| \\
 & & \langle atomicConcept\rangle\leq \langle concept\rangle| \\
 & & \langle atomicConcept\rangle \leq ANYTHING \\
\langle roleIntroduction\rangle & ::= & \langle atomicRole\rangle\doteq \langle role\rangle| \\
 & & \langle atomicRole\rangle\leq \langle role\rangle| \\
 & & \langle atomicRole\rangle \leq ANYROLE \\
\langle concept\rangle & ::= & \langle atomicConcept\rangle| \\
 & & (AND\langle concept\rangle^+)| \\
 & & (ALL\langle role\rangle\langle concept\rangle)| \\
 & & (ATLEAST\langle nombre\rangle\langle role\rangle)| \\
 & & (ATMOST\langle nombre\rangle\langle role\rangle) \\
\langle role\rangle & ::= & \langle atomicRole\rangle \\
\langle restriction\rangle & ::= & (DISJOINT\langle atomicConcept\rangle\langle atomicConcept\rangle) \\
\langle nombre\rangle & ::= & \langle entierNonNegatif\rangle \\
\langle atomicRole\rangle & ::= & \langle idetinfieur\rangle \\
\langle atomicConcept\rangle & ::= & \langle identifieur\rangle
\end{array}
$$

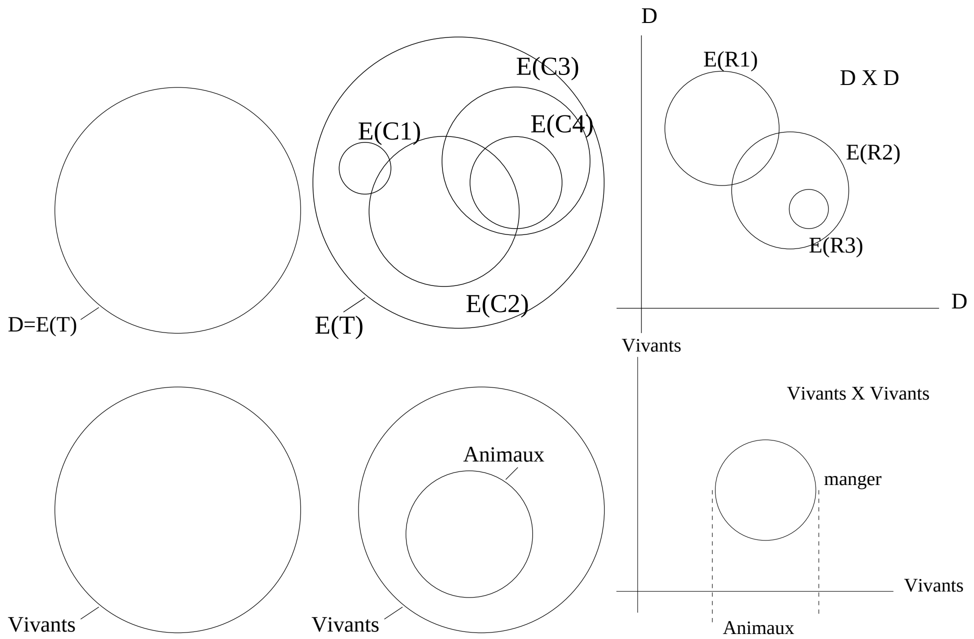
Supposons par exemple, que nous soyons en train d'examiner la sémantique d'une terminologie sur les objets vivants. Alors l'ensemble D de tous les objets de l'univers sera tout simplement l'ensemble de "tous les vivants possibles" (plantes, animaux, etc).

Cette façon de faire est proche de la notion d'interprétation que l'on utilise pour donner une sémantique à des systèmes de formules de LPO. A partir de cet univers, nous voulons donner un sens à toutes les structures qui interviennent dans la terminologie.

II.2.2.1.a Concepts et rôles Les notions auxquelles on s'intéresse dans TF étant les *concepts* et les *rôles*, nous allons donner le sens intuitif d'un concept et d'un rôle dans le formalisme des ensembles. On considère que :

- Un concept est *défini* en *intension* dans la terminologie, grâce à des expressions de LD. Par exemple, toujours dans la terminologie des animaux, on peut vouloir définir le concept de Animal. On définira alors ce concept comme "un vivant qui mange" par exemple (ce qui s'écrit en TF :
  $Animal \doteq (AND\ Vivant\ (ATLEAST\ 1\ mange))$).
- Un concept a pour *sémantique* un *sous-ensemble* de D correspondant à l'*extension* du concept (voir figure II.1). L'extension du concept *Animal* sera alors un sous-ensemble des vivants contenant tous les individus dont on sait que ce sont des animaux.
- Un rôle étant une relation entre deux individus, on va dire dans ce cas, qu'il représente une relation entre deux éléments de D. L'extension d'un rôle sera donc non pas un sous-ensemble de D mais un sous-ensemble de $D \times D$ (voir figure II.1). Ainsi l'extension du rôle *mange* est un sous-ensemble de $D \times D$ tel qu'il n'englobe que des points dont l'abscisse fait partie de l'ensemble des animaux.

FIG. II.1 – Sémantique d'une terminologie. a) La sémantique d'une terminologie est un ensemble D, appelé l'univers. b) La sémantique d'un concept dans une terminologie est un sous-ensemble de l'ensemble D représentant la terminologie. C'est aussi l'extension du concept. En effet, son intension est sa définition dans la terminologie. c) La sémantique d'un rôle dans une terminologie est un sous-ensemble de $D \times D$. En effet, un rôle est une relation binaire d'éléments de D. d) Exemple d'univers dénotant une terminologie sur le monde des vivants. e) Exemple de concept dénotant les Animaux comme sous-ensemble des vivants. f) Exemple d'extension de rôle dénotant la relation manger.



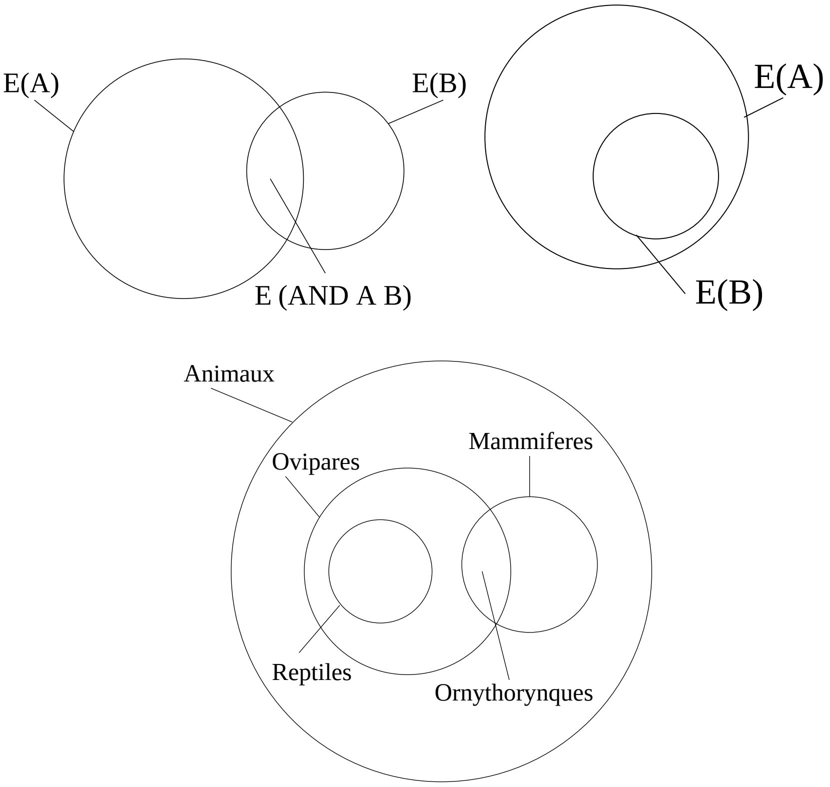
II.2.2.1.b Constructeurs En étudiant les relations entre les termes d'une terminologie, on peut analyser les relations entre les extensions de ces termes ; c'est-à-dire entre les ensembles que l'on dessine. Ainsi, on peut dessiner les configurations qui correspondent à l'utilisation de chaque constructeur de TF (pour les illustrations, se reporter à chaque fois à la figure II.2) :

- La conjonction (AND) : la conjonction de deux concepts est l'intersection de deux sous-ensembles. Si le sous ensemble $E(B) \subset D$ dénote le concept A et le sous-ensemble $E(B) \subset D$ dénote le concept B, alors le sous-ensemble $E(AND\ A\ B) = (E(A) \cap E(B)) \subset D$ dénote le concept $(AND\ A\ B)$. Par exemple, le concept d'ornithorynque peut être défini comme $(AND\ Ovipares\ Mammiferes)$ dans les *Animaux* (voir figure II.2).
- Définition d'un concept ($\doteq$) : Le fait de définir un concept revient à étiqueter un concept. Sa sémantique est donc l'étiquetage d'un sous-ensemble. Par exemple, on étiquette l'intersection précédente par *Ornithorynques* pour dénoter le fait d'écrire *Ornithorynques* $\doteq$ $(AND$ $Ovipares\ Mammiferes)$.
- Introduction d'un concept primitif ($\leq$) : La sémantique de l'introduction d'un sous-concept est l'inclusion ensembliste. Si A est un sous-concept de B, c'est-à-dire qu'on a $A \subset B$, alors l'extension de A, notée $E(A)$ est *incluse* dans l'extension de B, notée $E(B)$. Par exemple le concept *Reptiles* peut être un sous-concept de *Ovipares* : *Reptiles* $\leq$*Ovipares* (voir figure II.2).
- Introduction ou définition d'un rôle ($\doteq, \leq$) : De même, la sémantique de l'introduction d'un sous-rôle est aussi l'inclusion ensembliste. Si $R1$ est un sous-rôle de $R2$, c'est-à-dire qu'on a $R1 \subset R2$ dans $D \times D$, alors l'extension de $R1$, notée $E(R1)$ est *incluse* dans l'extension de $R2$, notée $E(R2)$ dans $D \times D$. Par exemple le rôle *manger* aurait pu être un sous-rôle de *ingerer* qui, dans ce cas, peut concerner aussi les plantes qui reçoivent de l'eau, etc.

Examinons maintenant le cas des restrictions utilisées pour définir un concept. Nous les traitons à part, car une restriction qu'elle soit de valeur ou de nombre s'applique en fait à un rôle utilisé pour définir un concept, sa sémantique est donc plus subtile. Nous avons :

- La restriction de valeur (ALL) : Cette restriction s'applique à un rôle R qui entre dans la définition d'un concept. Par exemple, considérons le concept $Herbivores \doteq (AND\ Animaux$ $(ALL\ manger\ Plantes))$. Cette restriction de valeur indique que pour être un herbivore, il faut être un animal et en plus, pour tout couple d'éléments (x, y) de l'extension du rôle *manger* où x est un animal, alors on sait qu'obligatoirement l'élément y de ce couple sera dans l'extension du concept *Plantes* (lui-même inclus dans $Vivants$ bien sûr). La figure II.3, montre la restriction du rôle *manger* comme un sous-ensemble de l'ensemble associé à *manger* ; le concept *Herbivores* est bien inclus dans le concept *Animaux* sur l'axe des abscisses, et le concept *Plantes* est quelquepart dans $Vivants$ sur l'axe des ordonnées. Ainsi, une restriction de valeur porte sur l'axe des *ordonnées* d'un rôle dans la représentation ensembliste.
- La restriction de nombre ($ATLEAST$ par exemple) : Soit la définition du concept $HerbivoreVaries \doteq (Herbivores\ (ATLEAST\ 3\ manger))$.
  C'est une type d'herbivores qui ont au moins trois liens de nourriture (pas forcément différentes ?), leur régime est donc un peu varié. Le concept $HerbivoresVaries$ sera donc bien sûr un sous-concept de *Herbivores*. La restriction $(ATLEAST\ 3\ manger)$ se traduit en disant que le nombre d'éléments y formant des couples (x, y) dans l'extension de *manger*, avec x élément du domaine *Herbivores*, doit être supérieur à 3 pour pouvoir dire que x est aussi dans $HerbivoresVaries$. En termes de graphes, cela veut dire que le concept $HerbivoresVaries$ doit avoir au moins trois arcs de type *manger* qui sortent de son noeud.

FIG. II.2 – Sémantique des constructeurs simples de concepts. a) La sémantique du constructeur AND est l'intersection ensembliste. b) La sémantique du constructeur $\leq$ est l'inclusion ensembliste. c) Illustration de ces deux sémantiques par l'exemple des animaux. Le concept de $Reptiles$ est inclus dans celui d'$Ovipares$, tandis que le concept de $Ornithorynques$ est défini comme l'intersection des concepts $Mammiferes$ et $Ovipares$.



Cela n'est représentable en termes d'ensembles que comme l'inclusion de $HerbivoresVaries$ dans $Herbivores$.

II.2.2.2 Formalisation

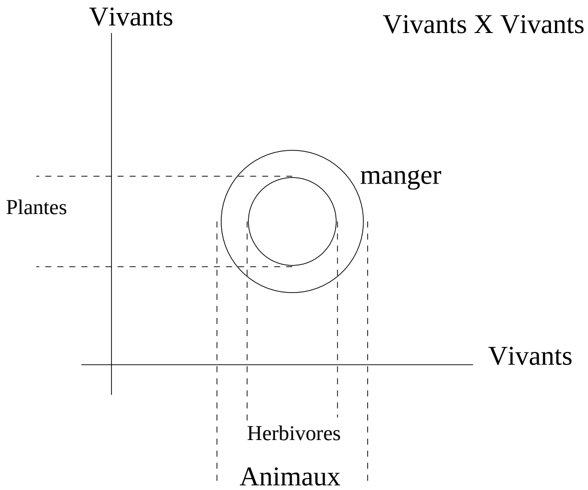
II.2.2.2.a Fonction d'extension Pour formaliser la sémantique que l'on donne pour chaque élément de la terminologie, on introduit la fonction d'extension ε telle que cette fonction appliquée à un concept de la terminologie renvoie l'extension que dénote ce concept dans le domaine D, ainsi :

$$\varepsilon(concept) = extension_D(concept)$$

Pour définir ε plus formellement, nous introduisons d'abord les notations suivantes :

N_c : ensemble des *concepts atomiques* de la terminologie,
N_r : ensemble des *rôles atomiques* de la terminologie,

FIG. II.3 – Sémantique des restrictions sur les rôles dans la définition de concepts. La restriction de valeur sur le rôle *manger* permet de définir le nouveau concept *Herbivores*. Cette restriction porte sur l'axe des ordonnées de ce rôle, ici l'élément de l'axe des ordonnées doit être inclus dans le concept *Plantes*.



N : ensemble général des termes *atomiques* il contient donc les concepts atomiques et les termes atomiques,
TF_c : ensemble de *tous les concepts*, pas forcément atomiques,
TF_r : ensemble de *tous* les *rôles*, pas forcément atomiques,
TF_T : ensemble de *tous les termes* de la terminologie.

II.2.2.2.b Concepts et rôles La sémantique des concepts et des rôles est donnée par la fonction d'extension ε, définie par :

$$\varepsilon : \begin{cases} TF_c \rightarrow 2^D \\ TF_r \rightarrow 2^{D\times D} \end{cases}$$

$$\varepsilon(ANYTHING) = D$$

$$\varepsilon(ANYROLE) = D \times D$$

En effet, l'extension du plus grand concept correspond au domaine D en entier, et l'extension de la plus vaste relation correspond à $D \times D$ puisque les relations ont pour extension des couples de valeurs dans D.

II.2.2.2.c Constructeurs Nous donnons maintenant la sémantique de chacun des *constructeurs* de TF. L'extension correspondant à une conjonction de concepts (AND) est donnée par *l'intersection* des extensions de ces concepts. Si un concept c est instance du concept C_1 et (AND) du concept C_2, alors son extension sera l'intersection $C_1 \cap C_2$:

$$\varepsilon\left[AND(c_1...c_n)\right] = \cap_{i=1}^{n}\varepsilon(c_i)$$

L'utilisation du constructeur $(ALL\ r\ c)$ dans une définition, correspond à une restriction de valeur sur le concept c auquel aboutit le rôle r dans le définition du concept courant. Cette restriction de valeur indique que pour tout couple d'éléments (x, y) de l'extension de ce rôle r, on sait qu'obligatoirement l'élément y de ce couple sera dans l'extension du concept c :

$$\varepsilon[(ALL\ r\ c)] = \{x \in D | \forall y : \langle x, y \rangle \in \varepsilon(r) \Rightarrow y \in \varepsilon(c)\}$$

L'utilisation du constructeur $(ATLEAST\ n\ r)$ dans une définition de concept, correspond à une restriction de nombre $(n \in N,\ n > 0)$ sur les role-filler du rôle r. Cela se traduit en disant que le nombre d'éléments y formant des couples (x,y) dans l'extension de r (avec x élément quelconque du domaine D), doit être supérieur à n. On note le cardinal par des barres verticales ($\|\|$) :

$$\varepsilon[(ATLEAST\ n\ c)]= \{x \in D | \| \{y \in D | \langle x, y \rangle \in \varepsilon(r)\} \| \geq n\}$$

On utilise le même principe pour donner une sémantique au constructeur $ATMOST$, sauf que cette fois-ci le nombre d'éléments y doit être inférieur à n :

$$\varepsilon[(ATMOST\ n\ c)]= \{x \in D | \| \{y \in D | \langle x, y \rangle \in \varepsilon(r)\} \| \preceq\leq n\}$$

Si de plus on ajoute la contrainte que ε respecte la restriction $DISJOINT$, c'est-à-dire que deux extensions sont disjointes quand leurs intensions sont exclusives (un concept a n'est jamais du type b et vice-versa), cela impose la structure suivante :

Pour tout terme atomique a, et C_1, C_2 primitifs, et tout terme arbitraire t de la terminologie, on a les trois propriétés suivantes

- Les extensions de deux termes égaux de T sont égales : $\varepsilon(a) = \varepsilon(t)\ \forall (a = t) \in T$;
- L'extension d'un terme atomique qui est sous-concept d'un terme quelconque est incluse dans l'extension de ce dernier : $\varepsilon(a) \subseteq \varepsilon(t)\ \forall (a \leq t) \in T$;
- L'intersection des extensions de deux concepts disjoints dans T est vide : $\varepsilon(c_1) \cap \varepsilon(c_2) = \emptyset$ $(DISJOINT\ c_1\ c_2) \in T$.

Lorsque l'on définit cette structure pour la fonction d'extension on parle alors d'une structure sémantique, notée $\langle D, \varepsilon \rangle$: elle dépend de la fonction ε mais aussi du domaine choisi D.

Par ailleurs, la subsomption $(\leq)$ entre termes se traduit naturellement par l'inclusion d'ensembles entre les extensions dans la sémantique introduite. Ainsi on peut dire que :

> t est *subsumé* par t' dans T ssi quelle que soit la structure sémantique $\langle D, \varepsilon \rangle$ de T, $\varepsilon(t) \subseteq \varepsilon(t')$, noté $t \preceq_T t'$.

où la relation $\preceq_T$ est *transitive* et *réflexive.*

II.2.2.2.d Autres relations Les autres relations pertinentes sont *l'équivalence* et *l'incohérence* définies ci-dessous. En effet, deux termes sont dits *équivalents* si pour toute structure sémantique (un domaine et une fonction d'extension) ces deux termes ont la *même* extension :

> On dit que t et t' sont *équivalents*, et on le note $t \approx_T t'$ dans T ssi quelle que soit la structure sémantique $\langle D, \varepsilon \rangle$ de T, on a $\varepsilon(t) = \varepsilon(t')$.

Enfin, on dira qu'un concept est *incohérent* dans une terminologie si pour toute structure sémantique de cette terminologie on a les deux restrictions de nombre contradictoires :

On dit que qu'un concept c est *incohérent* quelque soit la structure sémantique $\langle D, \varepsilon \rangle$ de T, si on a $AND(ATLEAST(1,r)ATMOST(0,r))$ sur un rôle r attaché à t.

En effet, si un concept contient dans sa définition un rôle qui doit être rempli au moins une fois mais au plus 0 fois, cela pose un problème !

II.2.2.2.e Notion de cycle Enfin, étant donné que la *sémantique* d'un concept dépend de la sémantique de ses composants et de celle des constructeurs utilisés pour les lier, il est nécessaire de ne pas introduire de *cycles* dans les définitions. En effet, définir un premier concept en fonction d'un deuxième, lui-même défini en fonction du premier revient à ne jamais définir le contenu. Si une chaise est une *sorte-de* siège et qu'un siège est *par-exemple* une chaise, on ne peut pas exploiter ces connaissances en tant que définitions. Pour éviter les cycles dans les définitions de concepts, nous commençons par cerner cette notion, en donnant une définition formelle d'un *cycle terminologique* :

On dit que a utilise *directement* a' dans T ssi le côté droit de l'introduction de a mentionne a'. On dit que a_0 utilise a_n ssi $\exists$ une chaîne d'introductions $a_0, ..., a_n$ dans T telle que a_i utilise directement a_{i+1}. Alors on dit que T contient un *cycle terminologique* si a s'utilise lui-même dans T.

II.2.3 A propos des notations

II.2.3.1 Syntaxe abstraite

La syntaxe de TF est facile et parlante, nous avons déjà mentionné qu'elle était *concrète*, mais ce n'est pas toujours celle qui est employée dans le domaine. Il arrive souvent que les constructeurs soient écrits en toutes lettres et placés en première position d'une liste entre parenthèses (comme c'est le cas ici), mais bien souvent on adopte pour chacun d'eux une notation très proche de la logique, c'est alors une notation *abstraite*. En effet, à chaque constructeur correspond en général (au moins intuitivement) un connecteur de LPO et on utilise donc ces notations. Cependant, cela implique que pour les constructeurs qui n'ont pas d'équivalents en LPO on continue à les écrire en toutes lettres, et cela donne des notations assez hybrides et confuses. Nous reviendrons sur ces notations dans la section II.4.2.1.

II.2.3.2 Sémantique

De même, dans la plupart des articles du domaine, la sémantique est donnée avec une notation un peu plus *concise*, et donc plus hermétique. La notation choisie dans cette introduction est plus parlante et pédagogique. En général, il n'est pas question d'une fonction d'extension mais d'interprétation :

A la fonction d'extension définie ci-dessus correspond une fonction d'interprétation, notée $.^I$

Au domaine D correspond toujours un domaine, mais il est noté Δ^I

Le fait d'indicer ces deux notions par la lettre I vient du fait que l'ont parle *d'interprétation* de manière *globale* pour désigner le couple $\langle \Delta^I . ^I \rangle$ et on le note alors tout simplement I, on a donc : $I = \langle \Delta^I . ^I \rangle$.

II.2.4 Illustration par l'exemple

II.2.4.1 Terminologie informelle

Voici la description informelle d'une terminologie en *français* :

- Un Homme est un Humain ;
- Une Femme est un Humain ;
- Aucun Homme n'est une Femme et vice versa ;
- Une Équipe est un Ensemble avec au moins 2 membres qui sont des Humains ;
- Une Petite-Equipe est une Equipe avec au plus 5 membres ;
- Une Equipe-Moderne est une Equipe avec au-plus 4 membres et au-moins 1 chef qui est un membre et tous les chefs sont des Femmes.

II.2.4.2 Représentation en réseau SIN

Représenter la terminologie précédente avec un réseau SIN à la manière de KL-ONE donne le schéma de la figure II.4. Sur cette figure, les relations d'héritage sont schématisées par un trait double, les relations possibles matérialisées par des rôles sont dessinées avec des traits simples coupés au milieu par une boule. Cette boule sert lorsque l'on veut pointer sur le rôle pour lui même, c'est sa *localisation*. En effet, les restrictions de valeur et les différenciations de rôles portent sur des rôles (nous verrons cela justement à l'aide de cet exemple dans quelques lignes).

On distingue pour cet exemple, deux sous-hiérarchies (qui seraient contenues dans ANYTHING) : l'une concernant les équipes et l'autre concernant les concepts liés au concept Humain. Ces deux hiérarchies interviennent dans la terminologie par le lien des rôles nécessaires à la définition des concepts de Petite-Equipe et de Equipe-Moderne. Ces rôles sont de différentes sortes :

- Restriction de valeur : le lien entre Equipe et Humain (trait simple) constitue la définition du rôle membre ; de même, le lien entre Equipe-Moderne et Femme constitue la définition du rôle chef ; ce type de restriction relie toujours deux concepts.
- Restriction de nombre : le lien entre Petit-Equipe et membre, et le lien entre Equipe-Moderne et membre sont des restrictions de nombre ; elles restreignent le nombre d'individus (de role-fillers) ; cette restriction s'applique donc toujours entre un concept et un rôle.
- Différenciation : le seul exemple ici est le lien entre chef et membre ; la différenciation s'applique toujours entre deux rôles.

II.2.4.3 Définition à l'aide de TF

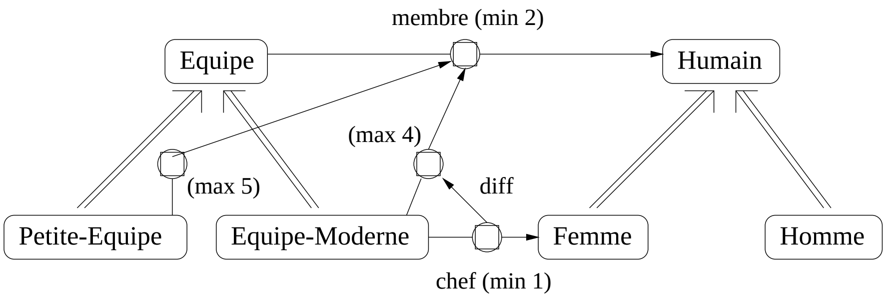
Définissons maintenant cette terminologie à l'aide d'introduction de termes avec la syntaxe TF. Nous introduisons les concepts primitifs :

$Humain \leq ANYTHING$

$Homme \leq Humain$

$Femme \leq Humain$

FIG. II.4 – Représentation en réseau SIN de la terminologie sur les équipes.



$Ensemble \leq ANYTHING$

Où nous introduisons le super concept $ANYTHING$ qui n'apparaissait pas dans la représentation en SIN.

Les deux rôles dont il est question dans cet exemple sont celui de membre et de chef, nous introduisons donc les rôles atomiques :

$membre \leq ANYRELATION$

$chef \leq membre$

Nous précisons que les concepts d'homme et de femme sont disjoints à l'aide du constructeur $DISJOINT$:

$(DISJOINT\ Homme\ Femme)$

Et nous introduisons enfin les concepts atomiques plus complexes par des équations. Une équipe est un ensemble mais moyennant certaines restrictions. Une restriction de valeur concernant le rôle *membre* : il en faut au moins deux et ils doivent tous être du type humain :

$Equipe \doteq (AND\ Ensemble\ (ALL\ membre\ Humain)\ (ATLEAST\ 2\ membre))$

Une petite équipe est alors une équipe moyennant une restriction supplémentaire : il faut au maximum 5 membres :

$PetiteEquipe \doteq (AND\ (Equipe\ (ATMOST\ 5\ membre))$

Enfin, une équipe moderne est une équipe ayant au moins un lien de rôle $chef$ avec un individu du type $Femme$:

$EquipeModerne \doteq (AND\ Equipe\ (ATMOST\ 4\ membre)\ (ATLEAST\ 1\ chef)\ (ALL\ chef\ Femme))$

II.3 Bases de connaissances en LD

II.3.1 Connaissances intensives et extensives

Pour constituer une base de *connaissances* à propos d'un *domaine*, on doit disposer d'un *schéma*[77] définissant les concepts du domaine et précisant leurs relations : c'est ce qui est défini dans la terminologie. Cependant, la base ne portera sur un domaine *réel* que dans la mesure où l'on *instancie* ce schéma, pour permettre l'introduction *d'individus effectifs* à propos desquels on pourra *asserter* des faits.

II.3.1.1 Hypothèses sur le monde

Dans une base de connaissances écrite en LD, toute assertion à propos d'un monde se fonde sur les deux hypothèses suivantes :

- **Unique Name Assumption** (UNA) : un même symbole désigne toujours le même objet ; si deux individus sont différents, alors ils ont obligatoirement des identifieurs différents.
- **Closed World Assumption** (CWA) : comme en LPO, seules les descriptions explicitées sont considérées comme vraies.

II.3.1.2 Interprétation ensembliste

En termes d'ensemble, on peut considérer que définir une terminologie revient à *introduire* des ensembles et à fixer les relations ensemblistes qu'ils vérifient (voir figure II.5). Instancier ce schéma revient alors à introduire des éléments effectifs à *l'intérieur* des ensembles ainsi définis. Un *monde* est alors une *configuration* d'éléments respectant les relations ensemblistes *prédéfinies* dans la terminologie (voir figure II.5).

Ainsi, une base de connaissances se compose de deux *types* de connaissances :

Intensive elle s'exprime à l'aide du langage terminologique, elle est stockée dans ce qui s'appelle la *TBox* (voir figure II.6)

Extensive elle constitue une *instanciation* de la terminologie (une extension effective correspondant aux concepts définis en intension dans la terminologie). Pour exprimer ces assertions, on définit un langage assertionnel. Le langage assertionnel correspondant à TF s'appelle AF, nous le décrivons en page 58. De telles assertions sont stockées dans ce qu'on appelle la $ABox$ (voir figure II.6).

II.3.1.3 Modèle pour une terminologie

Une description Ω est intuitivement le fait de dessiner une situation qui soit une des incarnations possibles de la terminologie T. Cette description repose sur l'introduction d'un ensemble D appelé le domaine. Un modèle M de cette description est alors un couple $\langle D, I\rangle$ où D est le

FIG. II.5 – Interprétation ensembliste de la définition d'une terminologie et des assertions s'y rapportant. a) Une terminologie définit un schéma de relations entre concepts. La sémantique de ces concepts est ensembliste ainsi que leurs relations. Ici nous avons introduit le concept $Humain \leq ANYTHING$, ainsi que $Petit \leq Humain$ et $Maigre \leq Humain$. nous ne précisons rien sur la disjonction entre $Petit$ et $Maigre$, l'intersection peut être non-vide. b) Une instanciation de ce schéma correspond à l'introduction d'individus effectifs (les croix) dans les ensembles de telle façon qu'ils respectent les relations ensemblistes vérifiées par les concepts. c) Autre instanciation possible du même schéma. Même si le concept d'humain petit *et* maigre est introduit dans la terminologie, il n'est pas obligatoire d'introduire un individu qui soit une instance de ce concept.

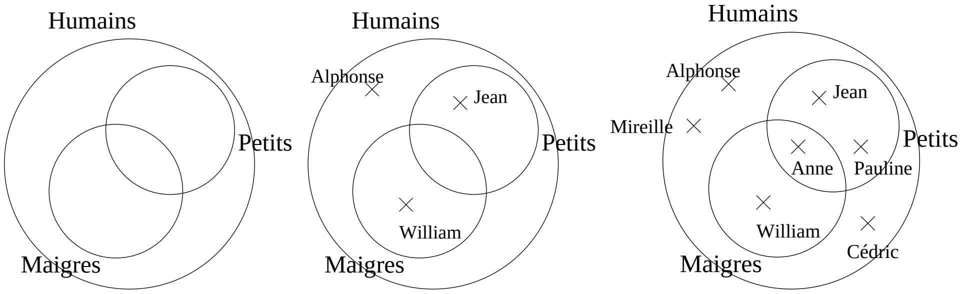
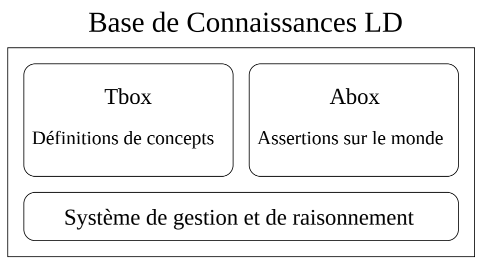


FIG. II.6 – Une base de connaissances en LD : elle se compose d'une Tbox contenant des définitions terminologiques et d'une Abox contenant des assertions sur un monde effectif (une description).



domaine d'éléments et I la fonction d'interprétation qui permet d'associer un sous-ensemble de D à chaque élément de la description, on le note alors $M\langle D, I\rangle$.

D'un point de vue un peu plus formel, nous dirons *qu'interpréter* une terminologie T en une *description* du monde Ω revient à relater un *modèle* M de cette description à la *structure sémantique* ε de T. Ainsi, relater ce modèle à la description de la structure de T revient à ce que M respecte ε, il en découle la définition suivante :

> Un modèle $M\langle D, I\rangle$ de la description Ω respecte la terminologie T ssi $\exists$ une structure sémantique $\langle D, \varepsilon\rangle$ de T telle que la restriction de I et ε à des concepts et rôles atomiques $(I_{/N}, \varepsilon_{/N})$ sont égales.

Où N est l'ensemble des termes atomiques défini page 51.

II.3.1.4 Les Tbox

Une Tbox est une suite d'introductions de termes d'une terminologie et ne contient donc *a priori* que des connaissances intensionnelles. La plupart des sytèmes de LD existants utilisent des Tbox non cycliques, c'est-à-dire où les cylces dans les définitions sont interdits. En général on peut distinguer au moins trois types de Tbox :

Acycliques : ensemble fini d'introductions de termes tels que chaque concept n'est défini qu'une fois et que la Tbox ne contient aucun cycle.

Générales : ensemble fini d'introductions de termes tels que chaque concept n'est défini qu'une fois. Les définitions cycliques sont donc autorisées et augmentent le pouvoir d'expression de la Tbox. On peut par exemple définir le concept d'humain comme étant tous les mammifères dont les deux géniteurs sont humains ! On peut alors considérer au moins 3 sémantiques différentes pour ce type de définition.
a) La *sémantique descriptive* : la sémantique des Tbox acycliques est étendue aux définitions cycliques.
b) La sémantique du *plus petit point fixe* : on restreint les modèles possibles pour la Tbox au plus petit point fixe de l'équation correspondant à l'introduction du concept (dont la définition est cyclique)
c) La sémantique du *plus grand point fixe* : on restreint les modèles possibles pour la Tbox au plus grand point fixe.

Libres (free Tboxes) : cette fois on autorise des concepts généraux à gauche des symboles d'introduction (définition $\doteq$ ou spécification $\leq$). Ici les définitions cycliques sont un cas particulier. Les distinctions entre concepts primitifs, atomiques ou définis ne sont plus possibles. La sémantique est toujours descriptive.

II.3.2 Langage assertionnel AF

On définit un langage assertionel qui permettra d'écrire des assertions à propos d'individus qui respectent une terminologie définie à l'aide de TF. Le langage introduit est noté AF, les expressions écrites en AF sont rangées dans la Abox.

II.3.2.1 Syntaxe de AF

Soient a et b deux individus du domaine D à propos duquel on veut asserter. Une assertion est obligatoirement de la forme (en notation abstraite) :

$$C(a)$$

$$R(a,b)$$

Où C est un concept et R est un rôle. C'est-à-dire que les seules assertions possibles sont :

- l'individu a est une *instance* du concept C, où C est défini dans la terminologie et peut être soit atomique (primitif) soit composé (à l'aide de constructeurs).
- les individus a et b sont en *relation* R, où R est un rôle de la terminologie.

La *syntaxe* exacte de ces assertions dépend bien sûr de la syntaxe du langage *terminologique* choisi. Dans le cadre de cette introduction, nous avons choisi de présenter TF, la syntaxe de AF doit donc s'accorder avec la syntaxe de la table II.1, nous en donnons donc une expression *concrète*. Une assertion dans AF étant une description du monde, elle peut porter sur un objet ou sur une relation effective, ce qui s'écrit :

$$\langle \mathit{description} \rangle ::= (\langle \mathit{objetDescription} \rangle | \langle \mathit{roleDescription} \rangle)^*$$

Une description d'objet est obligatoirement une instanciation d'un concept atomique introduit dans la terminologie, elle se réduit donc à :

$$\langle \mathit{objectDescription} \rangle ::= (\langle \mathit{atomicConcept} \rangle \langle \mathit{object} \rangle)$$

où le $\langle \mathit{atomicConcept} \rangle$ possède la syntaxe donnée par la table II.1 et où $\langle \mathit{object} \rangle$ est un individu. La description de rôle, quand à elle, peut être plus complexe puisqu'elle peut faire intervenir les constructeurs $ATLEAST$ ou $ATMOST$ sur les rôles, on a donc :

$$\langle \mathit{roleDescription} \rangle ::= (\langle \mathit{atomicRole} \rangle \langle \mathit{object} \rangle \langle \mathit{object} \rangle)|$$

$$(\langle \mathit{atomicRole} \rangle \langle \mathit{object} \rangle (ATLEAST \langle \mathit{number} \rangle))|$$

$$\langle \mathit{atomicRole} \rangle \langle \mathit{object} \rangle (ATMOST \langle \mathit{number} \rangle))$$

Ainsi, la syntaxe complète de AF est donnée par la table II.2.

TAB. II.2 – Syntaxe concrète du langage assertionnel AF lié au langage terminologique TF.

$\langle description\rangle$	::=	$(\langle objetDescription\rangle \mid \langle roleDescription\rangle)^*$
$\langle objectDescription\rangle$	::=	$(\langle atomicConcept\rangle\langle object\rangle)$
$\langle roleDescription\rangle$	::=	$(\langle atomicRole\rangle\langle object\rangle\langle object\rangle)\mid$
		$(\langle atomicRole\rangle\langle object\rangle(ATLEAST\langle number\rangle))\mid$
		$\langle atomicRole\rangle\langle object\rangle(ATMOST\langle number\rangle))$

II.3.2.2 Sémantique formelle de AF

La sémantique formelle des langages assertionnels suit celle des langages terminologiques. Ici encore, on s'appuie sur les ensembles. On introduit la notation N_o pour l'ensemble des objets existants dans Ω , et la fonction d'interprétation I :

$$I : \begin{cases} N_o \to D \\ N_c \to 2^D \\ N_r \to 2^{D\times D} \end{cases}$$

Où N_c et N_r sont définis page 51. On définit alors une *interprétation* à partir des notations abstraites :

> Une interprétation $M = \langle D, I\rangle$ satisfait une description δ, et on le note $\models_M \delta$ ssi on a :
> $\models_M (C(o))$ ssi $I(o) \in I(C)$
> $\models_M (R(o,p))$ ssi $\langle I(o), I(p)\rangle \in I(R)$
> Une interprétation est un *modèle* d'une description d'un monde Ω si elle satisfait à toutes les descriptions de ce monde :
> $\models_M \Omega$ ssi $\Omega \models \delta$

Si nous reprenons les notations concrètes, nous aurons par exemple :

$$\models_M (r\ o\ (ATLEAST\ n)) \text{ ssi } |\{\langle I(o), x\rangle \in I(r)\}| \geq n$$

II.3.2.3 Les Abox

Ce formalisme n'est pas le seul introduit dans la littérature. Par exemple on trouve certaines Abox avec des connecteurs logiques dans les assertions : $\vee$, $\wedge$, $\neg$, etc. On appelle de telles Abox des Abox *complexes*. Cependant de telles assertions peuvent être simulées à l'aide de constructeurs de concept judicieusement employés dans la Tbox [77].

II.3.2.4 Illustration par l'exemple

Revenons à l'exemple de terminologie donnée en fin de section précédente (voir page 54) concernant les équipes modernes. Voici un certain nombre d'assertions (en notation concrète, langage AF) *compatibles* avec la terminologie précédente :

$(EquipeModerne\ \#equipeA)$

$(Homme\ \#joe)$

$(Homme\ \#harry)$

$(Humain\ \#mary)$

$(membre\ \#equipeA\ \#joe)$

$(membre\ \#equipeA\ \#harry)$

$(chef\ \#equipeA\ \#mary)$

On introduit :

- un individu du concept EquipeModerne, appelé A ;
- deux individus (précédés du symbole #) du concept Homme, appelés joe et harry ;
- un individu du concept Humain, appelé Marie (on ne sait donc pas si c'est un homme ou une femme) ;
- deux individus du role-filler membre, qui sont nommés joe et harry et en vertu de l'hypothèse UNA, ils correspondent aux mêmes individus introduits précédemment ;
- un individu du role-filler chef, nommé Marie, de même en vertu de UNA, il correspond à l'individu Marie déjà introduit.

$(membre\ \#equipeA\ (ATMOST\ 3))$

Et on introduit enfin une restriction de nombre supplémentaire portant uniquement sur l'équipe A.

A partir de ces assertions, on peut *déduire* que Marie est une Femme par exemple. En effet, elle est chef d'une EquipeModerne, dont on connaît tous les membres (puisqu'on sait qu'ils sont limités à 3 et on en connaît exactement 3), et dont on sait déjà que deux d'entre eux sont des Hommes.

II.3.3 Classifieur pour la base

Les services de *raisonnement* les plus simples fournis par un système DL, pour une base $B = \langle Tbox, Abox \rangle$ donnée, sont ceux d'un classifieur :

Satisfiabilité *conceptuelle* : vérifier si un *concept* C est *satisfiable* (ce qui s'écrit $C \neq \perp$) par rapport à B, c'est-à-dire existe-t-il un modèle M tel que $M \models c$;

Subsomption *terminologique* : vérifier si un concept C est *subsumé* par un concept C' (ce qui s'écrit $C \sqsubseteq C'$) dans *tous* les modèles de B ;

Consistency (ou *satisfiabilité* de la *base*) : vérifier si la base B est *satisfiable*, c'est-à-dire si elle possède un modèle ;

Instance Checking : vérifier si *l'assertion* $C(a)$ est satisfaite dans tous les modèles de B.

Subsomption *hybride :* vérifier si un concept est subsumé par un autre en *tenant compte* de toute la base B (en particulier des assertions).

La subsomption est centrale pour les tâches plus complexes de classification. La consistency est utilisée pour vérifier qu'une base est *cohérente*. L'importance de ces deux tâches à été étudiée dans [71]. L'instance Checking est utilisée pour *retrouver* des informations dans la base. Notons que les deux premières tâches sont effectuées seulement sur la terminologie alors que les trois dernières utilisent la base d'assertions.

Des services plus complexes peuvent être effectués sur des expressions de concepts. Par exemple, Brachman et Levesque donnent en 1984 [28] un algorithme pour une comparaison *structurelle* entre expressions, en procédant à la comparaison des sous-expressions entre elles (proche de l'appariement de structures pour les analogies).

Une voie de recherche très explorée est celle du calcul de *complexité* pour chacune de ces tâches pour une logique de description donnée. En effet, on choisit généralement les LD car elles possèdent une sémantique formelle, mais il faut ensuite se préoccuper des aspects plus *algorithmiques* pour en faire des sytèmes *implémentables*. Cette mesure est toujours citée lorsqu'une *nouvelle* LD est proposée.

II.4 Les langages de LD plus complexes

II.4.1 Famille de référence

Le langage terminologique présenté dans les sections précédents, appelé TF, est très proche du langage de *base* introduit par Brachman et Schmolze dans [16] mais n'est pas exactement le même. Dans ce document, le langage TF a été présenté car il est simple et clair. Cependant, il ne constitue pas le langage de référence dans le domaine des LD aujourd'hui. Nous allons voir quels sont les langages de référence, comment ils se définissent et comment on peut situer TF par rapport à eux.

La *famille* de langages conceptuels de *référence* dans la littérature actuelle est la famille des AL-Languages. Cette famille inclut la plupart des langages utilisés et étudiés aujourd'hui. Le langage le plus *simple* de cette famille est lui-même appelé AL, c'est en fait une extension du langage FL^-(Frame Language), introduit par Brachman et Schmolze en 1984 [16]. La famille des *AL-Languages* est constituée de tous les langages formés de AL auquel on ajoute des constructeurs supplémentaires.

II.4.2 Tous les constructeurs répertoriés

Avant de nous intéresser aux constructeurs qu'il est possible d'ajouter à AL, intéressons-nous à *tous* les constructeurs qu'il est possible d'imaginer et qui ont été proposés dans la littérature. Certains sont inclus dans AL et d'autres non, mais ce n'est pas ce qui est examiné pour le moment. La liste la plus exhaustive des constructeurs utilisés est constituée à partir de [12][28][71][78][77][6]. Les constructeurs peuvent porter sur la formation de concepts, ou de rôles (notons que nous n'en avions pas introduits dans TF).

II.4.2.1 Les constructeurs de concepts

Nous donnons maintenant dans le détail la description de chaque constructeur proposé dans la littérature. Dans l'ordre nous précisons :

$$\text{Nom [L] (nom, } \int \text{) : explication. } \varepsilon(\int)$$

Où ces symboles désignent :

- Nom : son nom en français.
- [L] : lettre qui sert à nommer ce constructeur dans la littérature lorsqu'il y en a une, sinon des crochets vides.
- nom : expression concrète.
- $\int$: expression abstraite.
- explication : sémantique intuitive.
- $\varepsilon(\int)$: sémantique formelle.

Toutes les sémantiques formelles sont des *expressions d'ensembles*, ces ensembles sont évidement tous constitués d'éléments du domaine D (voir sémantique de TF page 51).Voici les constructeurs possibles pour les concepts :

1. **Nom** de concept [] () : ce constructeur est le plus trivial, il consiste simplement en l'utilisation d'un *identifieur* pour désigner un concept. Sa sémantique est immédiate, elle a déjà été donnée dans l'exposé de TF, elle correspond tout simplement à un sous-ensemble de D.
2. **Top** [] (top,$\top$) : c'est le concept contenant (en tant qu'ensemble) tous les autres. Tout concept introduit est forcément une spécialisation de celui-ci. Son extension est donc le domaine tout entier : $\varepsilon(\top) = D$.
3. **Bottom** [] (bottom,$\perp$) : correspond au concept inclus dans tous les autres. Son extension est l'ensemble vide : $\varepsilon(\perp) = \emptyset$.
4. **Conjonction** [] (and, $\sqcap$) : elle correspond à un ET logique, et s'écrit $\sqcap$. Sa sémantique correspond à une intersection ensembliste. Ainsi l'extension d'un concept défini comme une conjonction est l'intersection des extensions de ses composants. Plus formellement, $\varepsilon(C_1 \sqcap C_2) = \varepsilon(C_1) \cap \varepsilon(C_2)$.
5. **Disjonction** [U] (or, $\sqcup$) : elle correspond à la disjonction usuelle en logique et s'interprète donc aussi en termes d'union d'ensembles. Ainsi, $\varepsilon(C_1 \sqcup C_2) = \varepsilon(C_1) \cup \varepsilon(C_2)$. C'est-à-dire que l'extension d'un concept défini comme étant C_1 OU C_2 est en fait l'union des extensions de ces deux concepts.
6. **Complément** sur les **concepts primitifs** [] (not, $\neg$) : ce constructeur réalise la négation d'un concept *primitif.* C'est-à-dire que l'on n'autorise que la négation sur des noms de concepts, des atomes, et pas sur des expressions définissant des concepts composés. Ainsi, on écrit $\neg C$ et sa sémantique correspond au complément ensembliste de l'extension de C dans le domaine considéré. Plus formellement, $\varepsilon(\neg C) = D \backslash \varepsilon(C)$
7. **Complément** sur les **concepts** en **général** [C] (not, $\neg$) : ce constructeur est le même que le précédent mais il porte sur *n'importe quel* concept de la terminologie, et pas seulement les concepts primitifs.
8. **Quantification universelle** [] (all, $\forall$) : ce constructeur porte sur le type de tout concept lié au concept défini par un lien de rôle R. On écrit $\forall R.C$, ce qui veut dire que tout lien R qui part du concept défini aboutit forcément à un objet de type C. Plus formellement, $\varepsilon(\forall R.C) = \{d_1 \mid \forall d_2 : (d_1, d_2) \in \varepsilon(R) \rightarrow d_2 \in \varepsilon(C)\}$.
9. **Quantification existentielle non qualifiée** [] (some, $\exists$) : la quantification existentielle porte sur un rôle R, elle est notée $\exists R$, et dénote l'ensemble d'objets ayant au moins un

lien de rôle de type R avec un autre objet. Cette sémantique s'écrit plus formellement comme $\varepsilon(\exists R) = \{d_1 \mid \exists d_2 : (d_1, d_2) \in \varepsilon(R)\}$. Elle est non qualifiée en ce sens qu'aucune condition sur d_2 n'est imposée. La seule information imposée est que le rôle soit du type R et qu'il y en ait au moins un, mais on ne dit pas vers quel type d'objets ce lien doit être tendu.

10. **Quantification existentielle qualifiée** $[E]$ (some, $\exists$) : cette quantification est comme la précédente mais elle qualifie le type de l'objet d_2. Ainsi, on écrit $\exists R.C$ ce qui veut dire que l'objet doit avoir au moins un lien de rôle R avec un objet du type C. En termes de sémantique, cela donne $\varepsilon(\exists R.C) = \{d_1 \mid \exists d_2 : ((d_1, d_2) \in \varepsilon(R)) \land (d_2 \in \varepsilon(C))\}$.
11. **Restriction** de **nombre** $[N]$ (atleast, $\geq n$) (atmost, $\leq n$) (exact n) : ce constructeur s'applique à la définition d'un concept mais précise une contrainte sur un *rôle* associé au concept. Il s'écrit $(\geq nR)$ pour dire qu'il faut que le *nombre* de liens de rôle de type R soit supérieur ou égal à n pour ce concept. La sémantique porte donc naturellement sur le nombre *d'éléments* entrant en relation R avec des éléments de l'extension du concept concerné. Plus formellement, nous avons $\varepsilon((\geq nR)) = \{d_1 \mid \{d_2 \mid (d_1, d_2) \in \varepsilon(R)\} \geq n\}$. La notation et la sémantique pour une restriction de nombre portant sur un nombre *maximum* de liens, sont obtenues à partir des précédentes en remplaçant le signe $\geq$ par $\leq$. La restriction appelée *exact*e, quant à elle, a pour sémantique la même expression en remplaçant le signe $\geq$ par un signe $=$.
12. **Restriction** de **nombre qualifiée** [] (atleast, $\geq$) (atmost, $\leq$) (exact $=$) : ce constructeur est le même que précédemment mais il qualifie le domaine du filler de R. On écrit par exemple $(\geq nR.C)$ pour dire qu'il doit y avoir au moins n liens de rôle R vers des objets du type C. La sémantique est alors $\varepsilon((\geq nR)) = \{d_1 \mid \{d_2 \mid ((d_1, d_2) \in \varepsilon(R)) \land d_2 \in (\varepsilon(C))\} \geq n\}$. De même pour le maximum et l'exact.
13. **Egalité** [] (eq, $=$) : ce constructeur pose une contrainte sur les rôles du concept défini. On l'utilise pour stipuler l'égalité entre deux rôles, mais ce n'est pas un constructeur de rôle ! On écrit $R_1 = R_2$ pour décrire tous les objets dont le lien de rôle R_1 est égal au lien de rôle R_2. Par exemple, on peut définir l'ensemble des gens qui vivent dans le pays où ils sont nés, ils ont le même filler pour le rôle de pays-de-naissance et pour le rôle de pays-de-résidence. La sémantique formelle est $\varepsilon(R_1 = R_2) = \{d \mid \varepsilon(R_1(d)) = \varepsilon(R_2(d))\}$.
14. **Inégalité** [] (neq, $\neq$) : comme le précédent, ce constructeur pose une contrainte sur les rôles du concept défini. On l'utilise pour stipuler l'inégalité entre deux rôles, mais ce n'est pas un constructeur de rôle ! On écrit $R_1 \neq R_2$ pour décrire tous les objets dont le lien de rôle R_1 est différent du lien de rôle R_2. La sémantique formelle est $\varepsilon(R_1 \neq R_2)= \{d \mid \varepsilon(R_1(d)) \neq \varepsilon(R_2(d))\}$.
15. **Sous-ensemble**[] (subset, $\subseteq$) : comme les deux précédents, ce constructeur pose une contrainte sur les rôles du concept défini. On l'utilise pour stipuler l'inclusion entre deux rôles, mais ce n'est pas un constructeur de rôle ! On écrit $R_1 \subseteq R_2$ pour décrire tous les objets dont le lien de rôle R_1 est inclus dans le lien de rôle R_2. La sémantique formelle est $\varepsilon(R_1 \subseteq R_2) = \{d \mid \varepsilon(R_1(d)) \subseteq \varepsilon(R_2(d))\}$.
16. **Collection** d'individus $[O]$ (oneof,$\{\}$) : ce constructeur spécifie le *type* d'un concept mais pas en *intension* comme nous en avions l'habitude. En effet, il précise une *liste d'individus* qui sont alors les valeurs *possibles* de ce concept, en fait on précise directement son *extension*. Pour cela il faut introduire un *alphabet* supplémentaire (par rapport à ceux introduits page 50) qui précise la liste d'individus autorisés dans la Tbox pour ce type de

constructeur. Ce nouvel alphabet est noté A. On écrit alors $\{a_1, ..., a_n\}$ où les éléments a_i sont pris dans A. La sémantique est donnée directement par cet ensemble, en effet il suffit de prendre l'union de toutes les extensions de tous les a_i comme l'extension du concept. Plus formellement, $\varepsilon(\{a_1, ..., a_n\}) = \{\varepsilon(a_1), ..., \varepsilon(a_n)\}$. Cela fait alors intervenir la sémantique des individus a_i, afin d'écrire une équation formellement juste. Intuitivement, ces extensions devraient correspondre aux individus eux-mêmes. Mais pour que l'on ait le droit de le dire, il faut d'abord étendre la fonction ε de sorte que les a_i introduits pour l'instant seulement au niveau de la Tbox, soient membres du domaine D et respectent l'hypothèse UNA (voir page 56).

17. **Role-filler** $[B]$ (is, :) : il permet d'imposer une contrainte sur la *nature* du role-filler, mais cette contrainte ne porte plus sur son type, comme le fait le constructeur E, elle impose directement quel individu doit être le role-filler. Ici encore, nous faisons intervenir un individu au niveau de la Tbox, cela implique encore que l'on utilise l'alphabet A introduit ci-dessus (voir constructeur O). On écrit $R : a$ et cela veut dire que pour le concept qui est en train d'être défini par cette expression, l'élément qui est en lien de rôle R avec lui doit forcément être l'individu a. La sémantique plus formelle est donnée par $\varepsilon(R : a) = \{d \mid (d, \varepsilon(a)) \in \varepsilon(R)\}$. Ici encore, nous utilisons l'extension de l'individu a introduit dans la Tbox pour rester formellement exact. Il est évident que cette extension correspond à l'individu lui-même si l'on prend la précaution d'étendre la fonction ε pour inclure a dans D en respectant l'hypothèse UNA (voir page 56).

Remarquons que les constructeurs 16 et 17 concernent des *individus* alors qu'ils agissent sur des concepts de la terminologie, nous reviendrons sur cette particularité dans la section d'évaluation, en page 68.

A cette liste, nous ajoutons les deux constructeurs essentiels qui permettent l'introduction des concepts : *l'inclusion* de concept et *l'équation* de concept. Ils sont souvent présentés à part, mais certains les appellent également des constructeurs. Il est vrai qu'ils ne permettent pas exactement de construire des concepts mais d'associer des définitions à des identifieurs. Ils ont déjà été définis dans le cadre de TF, voici leur notation :

- Inclusion de concept $\leq$
- Equation de concept $\doteq$

II.4.2.2 Les constructeurs de rôles

Nous donnons maintenant tous les constructeurs possibles pour les rôles, les sémantiques formelles sont prises dans $D \times D$.

1. **Nom** de rôle [] : comme pour les noms de concepts, cela revient à introduire un identifieur de rôle. La sémantique est un sous ensemble de $D \times D$.
2. **Conjonction** de **rôles** $[R]$ (and, $\sqcap$) : cette conjonction correspond également à une intersection comme celle des concepts (voir plus haut) mais elle s'applique sur des rôles. On l'écrit toujours à l'aide du symbole $\sqcap$. Ainsi $\varepsilon(R_1 \sqcap R_2) = \varepsilon(R_1) \cap \varepsilon(R_2)$.
3. **Disjonction** $[U]$ (or, $\sqcup$) : elle correspond à la disjonction usuelle en logique et s'interprète donc aussi en termes d'union d'ensembles. Ainsi, $\varepsilon(R_1 \sqcup R_2) = \varepsilon(R_1) \cup \varepsilon(R_2)$.

4. **Complément** sur les rôles[] (not, $\neg$) : ce constructeur réalise la négation d'un rôle. Ainsi, on écrit $\neg R$ et sa sémantique correspond au complément ensembliste de l'extension de R dans $D \times D$. Plus formellement, $\varepsilon(\neg R) = (D \times D) \backslash \varepsilon(R)$.
5. **Chaîne** de rôles [H]
6. **Identité** [] (id,) : ensemble des couples de concepts liés à eux-mêmes. Sémantique $\varepsilon(id) = \{(d, d) \mid d \in D\}$.
7. **Inverse** [I] (inverse,$^{-1}$) : ensemble des couples liés dans les deux sens par R, c'est le rôle réciproque, sémantique $\varepsilon(R^{-1}) = \{(d, d') \mid (d', d) \in \varepsilon(R)\}$.
8. **Restriction** [] (restrict,|) : couples liés entre eux par R mais dont le filler est dans le concept C, sémantique $\varepsilon(R|C) = \{(d, d') \mid ((d', d) \in \varepsilon(R)) \wedge (\varepsilon(d') \in \varepsilon(C))\}$.
9. **Composition** [] (compose,$\circ$) : on écrit $R_1 \circ ... \circ R_n$.
10. **Domaine** de **valeurs** [] (domrange,$\times$) : c'est un produit cartésien de deux concepts, on écrit $C_1 \times C_2$.

II.4.3 Les langages de la famille AL

II.4.3.1 Le langage TF de ce document

Le langage qui a servi à présenter les terminologies dans la section précédente est TF, il utilise bien entendu les deux constructeurs de d'introduction de concepts $\leq$ et $\doteq$. De plus, il utilise trois constructeurs de la liste de la page 63 :

$$TF = \{\sqcap, \forall r, \leq n, \geq n\}$$

- la **conjonction** : dont la notation concrète est AND dans TF. Ecrire $(AND\ C_1\ C_2)$ en TF revient à écrire $C_1 \wedge C_2$ dans les familles de référence.
- la **restriction** de **valeur** : notée ALL dans la section précédente ; ce constructeur correspond exactement à la quantification universelle. Ainsi écrire $(ALL\ R\ C)$ dans TF revient à écrire $\forall R.C$ dans les familles de référence.
- la **restriction** de **nombre** : notées $ATLEAST$ et $ATMOST$ dans la section précédente. Ainsi écrire $(ATLEAST\ n\ R)$ en TF revient à écrire $(\geq nR)$ dans les langages de référence. De même, écrire $(ATMOST\ n\ R)$ en TF revient à écrire $(\leq nR)$ dans les langages de référence.

II.4.3.2 Langage FL^-

Historiquement indépendant de TF, le langage FL^- est défini par les constructeurs suivants :

$$FL^- = \{\sqcap, \forall, \exists\}$$

- La **conjonction** de concepts (and, $\sqcap$).
- La **quantification universelle** sur les rôles (all, $\forall$).
- La **quantification existentielle** non qualifiée des rôles (some, $\exists$).

II.4.3.3 Langage AL

Le langage AL est défini comme une *extension* de FL^-et utilise, les constructeurs suivants :

$$AL = \{\sqcap, \forall, \exists, \neg, \top, \bot\}$$

et donc propose trois constructeurs supplémentaires :

- **Complément** de **concepts primitifs** (not, $\neg$).
- **Top** (top, $\top$).
- **Bottom** (bottom, $\bot$).

II.4.3.4 Formation des autres langages

Des langages plus puissants sont ensuite définis par ajout de constructeurs, mais il faut préciser que plus un langage autorise de constructeurs (plus il est expressif), plus le raisonnement dans les bases que l'on peut définir avec lui est difficile (Dans FL^-, le calcul des subsomptions se ramène à une complexité polynômiale, alors que l'ajout du constructeur de restriction de rôle rend ce calcul co-NP-complet[28]).

Le nom de ces langages est donné par référence à AL et par les noms des constructeurs supplémentaires acceptés (donc tous ceux qui portent des lettres), par exemple : $AL[U][C]$, si on ajoute les constructeurs U, C (disjonction et négation) à AL. La formule générale de constitution des noms de langages de cette famille est :

$$AL[E][U][C][R][N][O][B]$$

On aurait pû aussi donner ces langages par référence à FL^-, avec

$$FL^-[E][U][C][R][N][O][B]$$

mais on ne parle alors que des langages contenant les constructeurs de FL^- et ceux qui portent des lettres, donc on ne peut jamais inclure les trois qui font partie de AL, seulement (complément, top, bottom). Mais c'est une notation qu'il faut connaître car elle est utilisée parfois.

Par ailleurs, notons que certaines expressions sont équivalentes car il est possible de recréer certains constructeurs à partir d'autres. Par exemple, $ALEU$ est le même que ALC, $ALEUC$ et $ALEO$ sont les mêmes que $ALEOB$ (voir [78] pour des détails).

Cependant, le nombre de langages qu'il est possible de construire ainsi est exponentiel, seuls quelques-uns ont réellement été étudiés par la communauté, en fonction des avantages qu'ils présentent. Ainsi, par exemple, les langages $AL[C]$, $AL[U][R]$, $AL[N][R]$, $AL[C][N]$, $AL[E][R]$, $AL[C][N][R]$ sont abordés dans [29] au moins du point de vue de la complexité.

Enfin, il est quelquefois question des langages :

- Purs : il ne contiennent pas de constructeurs de concepts faisant intervenir des individus (nous verrons que cela pose en effet des problèmes, page 68), ce sont les langages sans les lettres O et B.
- Mixés : il contiennent au moins l'un de ces deux constructeurs.

II.4.4 Les langages mixés

II.4.4.1 Nécessité de raisonner sur des individus

La puissance des DL tient principalement au fait de séparer explicitement les connaissances terminologiques sur les concepts des connaissances assertionnelles sur les individus. Malheureusement, cette caractéristique entraîne une limitation du pouvoir d'expression [78] des langages conceptuels. Comme nous l'avons signalé plus haut, les LD séparent les connaissances concernant les concepts des connaissances concernant les individus ; cette distinction impose des contraintes du point de vue de *l'expressivité*.

On distingue dans la littérature essentiellement deux courants de propositions pour améliorer l'expressivité dans la Tbox :

1. On autorisant un peu de *mélange* (faible) entre connaissances intensives et extensives dans la Tbox ; Pour cela, on introduit les constructeurs qui autorisent des références à des *individus* dans des expressions de concepts, il s'agit de O et B décrits page 63. Schaerf étudie les conséquences d'une telle proposition dans [76]. L'enjeu est alors de réussir à rester dans le cadre des LD, c'est-à-dire, de conserver les bonnes propriétés des LD dûes justement à la séparation de ces connaissances !
2. On peut également introduire d'autres langages en introduisant d'autres notions élémentaires aux concepts et rôles ; par exemple, les *attributs*. Cela permet de définir directement les concepts par leur *extension*.

II.4.4.2 Introduire des constructeurs

Un langage mixé est un langage de la famille AL-Languages qui contient un des deux constructeurs suivants :

- Le constructeur de collection d'individus, parfois noté $OneOf$, mais toujours repéré par la lettre O : il permet la construction de concepts à partir d'une *énumération* d'individus. Cela correspond à créer un *domaine* de valeurs. Par exemple, on peut définir le concept de $CouleurArcEnCiel \doteq OneOf(violet, bleu, vert, jaune, orange, rouge, mauve)$ où $violet, ..., mauve$ sont des individus.
- Le constructeur role-filler, parfois noté $Fills$, toujours repéré par la lettre B : il dénote l'ensemble d'objets qui ont un individu particulier pour role-filler pour un rôle donné. Par exemple, on peut ainsi définir le concept de $CitoyensFrancais \doteq (Fills\ citoyennete\ france)$, où $citoyennete$ est un rôle et $france$ est un individu.

Notons que l'on peut exprimer B en termes de O et de E (quantification existentielle qualifiée, voir page 63). Cependant, la conséquence de l'introduction de ces constructeurs est la complexification des services de raisonnement. En effet, nous avons l'équivalence :

$$R : a \equiv \exists R.\{a\}$$

Autrement dit, préciser le role-filler comme étant l'individu a (constructeur B), revient à préciser l'existence d'un rôle vers l'ensemble d'individus constitué seulement de a (utilisation des constructeurs E puis O).

II.4.4.3 Raisonner avec les langages mixés

Une fois ces deux constructeurs introduits et une sémantique ensembliste proposée pour leur utilisation, nous nous interrogeons sur les conséquences de leur utilisation pour les tâches de raisonnement en LD. Considérons les aspects suivants :

II.4.4.3.a Assertions implicites : Les LD permettent d'exprimer de la connaissance incomplète, en particulier avec le constructeur existentiel ($\exists$), il est possible de parler d'un objet inconnu mais dont on sait qu'il existe dans la base. L'utilisation du constructeur O permet en plus de parler au sujet de son appartenance non plus seulement à un concept, mais aussi à un ensemble d'individus.

Une conséquence est alors que l'objet inconnu en question est lié à l'ensemble d'individus, comme en étant un membre. Par exemple, si l'on écrit dans la Tbox :

$$\exists : couleur.(CouleurArcEnCiel \sqcap \{bleu\})$$

où *couleur* est un rôle, $CouleurArcEnCiel$ est un concept et *bleu* est un individu, on dit qu'il y a un lien de rôle de type *couleur* vers un objet qui est dans l'intersection de $CouleursArcEnCiel$ et de l'ensemble $\{bleu\}$.

Cela suppose que *bleu* est aussi membre de $CouleurArcEnCiel$, en fait cela revient à asserter *implicitement* que $CouleurArcEnCiel(bleu)$. Cette assertion implicite peut compliquer les choses lorsque l'opérateur O est utilisé non plus avec un ensemble réduit à un *singleton*, mais contenant plusieurs individus. Par exemple, si on écrit :

$$\exists couleur.(CouleurArcEnCiel \sqcap \{bleu, jaune\})$$

alors on asserte $CouleurArcEnCiel(bleu) \vee CouleurArcEnCiel(jaune)$, et la *disjonction* est très importante. Cela veut dire que tout raisonnement s'appuyant sur cette base devra vérifier la consistance avec l'une *et* l'autre de ces assertions implicites.

II.4.4.3.b Mixage des assertions et de la terminologie : Le raisonnement sur les concepts d'une terminologie n'est en général pas influencé par des connaissances assertionnelles. En effet, une Abox, si elle est *satisfiable*, ne joue aucun rôle dans le raisonnement à propos des subsomptions entre concepts. Rappelons qu'une Abox est satisfiable s'il existe une interprétation qui en est un modèle (comme en LPO). Ainsi, en général, si deux concepts C et D sont en relation de subsomption dans la Tbox :

$$C \sqsubseteq D$$

alors il est équivalent d'utiliser les connaissances assertionnelles pour le déduire :

$$Abox \models C \sqsubseteq D$$

On dit que la relation de subsomption est réduite à la subsomption *terminologique*.

Cette propriété (qui est considérée comme agréable pour l'algorithmique) ne tient plus dès lors que l'on utilise le constructeur O. En effet, en utilisant O on n'a plus la simple subsomption terminologique, elle devient une subsomption *hybride*. Pour préciser le sens de cette subsomption de nature différente, prenons un exemple. Soit le système assertionnel A suivant :

$$CouleurArcEnCiel(bleu)$$

$$CouleurArcEnCiel(jaune)$$

Dans une terminologie qui contient :

$$C_1 \doteq \forall couleur.\{bleu, jaune\}$$
$$C_2 \doteq \forall couleur.CouleurArcEnCiel$$

il est *faux* de déduire que le premier concept C_1 est subsumé par le deuxième C_2 sans savoir que $bleu$ et $jaune$ sont des couleurs de l'arc en ciel ! On est obligés de se servir du système A pour *pouvoir* en déduire cette subsomption. En d'autres termes on n'a pas :

$$\forall couleur.\{bleu, jaune\} \sqsubseteq \forall couleur.CouleurArcEnCiel$$

alors qu'on a :

$$A \models \forall couleur.\{bleu, jaune\} \sqsubseteq \forall couleur.CouleurArcEnCiel$$

Nous sommes alors obligés de faire la distinction entre ces deux types de subsomptions (sans se servir des connaissances assertionnelles) lorsque l'on raisonne.

II.4.4.3.c Abstraction : *L'abstraction* est simplement le fait de retrouver *toutes* les assertions valides à propos d'un *individu* a donné et de les collecter dans la définition d'un *concept*. Ce concept est alors le concept le plus *spécifique* possible du langage tel que l'individu a en soit une *instance*. Cependant, en faisant cette opération, l'information concernant le fait que ce concept est exactement a est perdue. Dans un langage contenant O, il est possible d'opérer une abstraction sans perdre cette information.

II.4.4.3.d Restrictions de nombre : L'opérateur O offre la possibilité d'exprimer des *restrictions* de *nombre* sur les rôles puisqu'il donne des restrictions à l'aide de sous-ensembles dont il suffit de *compter* les éléments.

II.4.4.3.e Complément : Il est également possible d'exprimer explicitement le complément d'un concept par rapport à un concept donné en extension.

II.4.4.3.f Connecteurs logiques : Une Abox interdit en général l'utilisation de connecteurs logiques dans les assertions : les assertions doivent être purement atomiques.

II.4.4.4 Difficultés

Nous pouvons, pour conclure sur les langages mixés, résumer les difficultés que l'on rencontre pour raisonner avec :

- Le constructeur O : l'utilisation d'ensembles avec plus d'un individu implique des assertions implicites disjonctives. Une autre difficulté vient de l'identification d'un objet inconnu avec un individu cité dans la terminologie. La disjonction et l'égalité sont la cause des difficultés posées par cet opérateur pour raisonner.
- Le constructeur B : étant donné que B est déductible de O et E les difficultés propres à B découlent des difficultés de ces deux constructeurs. Les difficultés du connecteur E ne se retrouvent pas dans B [78]. De plus, B concerne toujours un *seul* individu, les problemes dûs à des disjonctions d'assertions implicites (du constructeur O) ne tiennent plus non plus. Le seul problème reste celui de l'égalité entre un objet inconnu et l'individu introduit par B.

II.4.4.5 Traitement de O et B dans les systèmes de LD

Reprenons les trois principaux systèmes implémentant les notions de LD, que nous avons cités en introduction : CLASSIC, BACK et LOOM. Voici, par exemple, comment CLASSIC et BACK traitent les constructeurs O et B :

II.4.4.5.a CLASSIC[10] Les individus apparaissant dans la terminologie sont considérés comme des concepts primitifs disjoints, ce sont alors des sous-ensembles du domaine, et on élimine le problème des assertions implicites. Les assertions sur les individus ne sont pas prises en compte lorsque l'on raisonne sur les concepts. en d'autres termes, c'est toujours la subsomption terminologique qui est examinée.

II.4.4.5.b BACK[99] Ces constructeurs ne sont pas permis dans BACK. Il est possible d'exprimer des collections d'individus, mais ces éléments sont appelés des attributs, ils appartiennent à un alphabet disjoint de celui des individus de la Abox. De plus, le domaine d'interprétation des attributs est disjoint de celui des concepts. Cette méthode passe à côté du pouvoir d'expression de O [78]. Nous examinons maintenant cette seconde façon de faire.

II.4.5 Introduire des attributs

Une autre voie pratiquée est celle de l'introduction d'un alphabet *d'attributs* en plus des concepts et rôles dans la terminologie. Cette solution est celle empruntée par exemple par les systèmes BACK et LOOM.

Le terme d'attribut est souvent utilisé dans les langages objets ou centré concept comme une variante au terme de relation ou de rôle. Dans le cas de LOOM et BACK, on introduit une nouvelle notion qui s'ajoute à celle de concept et de rôle : l'attribut.

II.4.5.1 Constructeurs d'attributs

Soit A_i un attribut, et A l'alphabet d'attributs, dont les extensions sont dans le domaine Δ_a. De même que pour les concepts et rôles, il y a des constructeurs d'attributs. On distingue :

- **Bottom** [] (bottom,$\perp$) : correspond à l'attribut inclus dans tous les autres. Son extension est l'ensemble vide : $\varepsilon(\perp) = \emptyset$.
- **Identité** [] (id,) : ensemble des couples d'attributs liés à eux-mêmes. Sémantique $\varepsilon(id) = \{(d,d) \mid d \in D_a\}$.
- **Conjonction** [] (and, $\sqcap$) : correspond à un ET logique, et s'écrit $\sqcap$. Sa sémantique correspond à une intersection ensembliste. On l'utilise entre un attribut et des rôles, on écrit $A_1 \sqcap R_1 \sqcap ... \sqcap R_n$, ce qui a pour sémantique $\varepsilon(A_1 \sqcap R_1 \sqcap ... \sqcap R_n) = \varepsilon(A_1) \cap \varepsilon(R_1) \cap ... \cap \varepsilon(R_1)$.
- **Restriction** [] (restrict,|) : on écrit $A_1|C$ pour définir un concept d'attribut A_1 mais dont la valeur est dans le concept C.
- **Composition** [] (compose,$\circ$) : pour composer plusieurs attributs, on écrit $A_1 \circ ... \circ A_n$.

II.4.5.2 Syntaxe induite dans la Tbox

Les concepts une fois définis, on peut alors définir des attributs dans la terminologie à l'aide des expressions de la table II.1, mais aussi selon la syntaxe supplémentaire suivante :

- Définition d'attribut par équation : $A_1 \doteq A_2$, sémantique $\varepsilon(A_1) = \varepsilon(A_2)$. Où A_2 peut être un attribut composé à l'aide des constructeurs détaillés ci-dessus.
- Définition d'attribut par spécification (attribut appelé primitif dans ce cas) : $A_1 \sqsubseteq A_2$, de sémantique $\varepsilon(A_1) \subseteq \varepsilon(A_2)$. Où A_2 peut être un attribut composé à l'aide des constructeurs détaillés ci-dessus.

II.4.6 Comparaison des LD avec la logique du premier ordre

Une comparaison entre le pouvoir d'expression des LD avec celui de la LPO est donnée par [12][5]. Elle établit que les LD proposées jusqu'à présent (y compris les complexes) peuvent exprimer *toutes* les notions exprimables dans la LPO avec trois variables *au plus*, et *seulement* ceux-là [12].

Par ailleurs, il existe des résultats dans la littérature stipulant qu'il y a des choses exprimables avec $k+1$ variables qui ne le sont pas avec seulement k variables. Notons que certains auteurs se préoccupent donc de cette limitation compte tenu du fait que les logiques de description ont été largement proposées comme langages de requêtes dans les bases de données.

II.5 Conclusion du chapitre

Nous avons présenté dans ce chapitre le formalisme des logiques de descriptions très utilisé en Intelligence Artificielle pour modéliser des bases de connaissances. Le langage de logique de descriptions que nous emploierons dans le reste du document est le plus simple décrit ici, il s'agit de AL. Nous verrons comment utiliser les notions de base des logiques de description pour modéliser notre problème des agents hétérogènes dans le chapitre qui suit.

Chapitre III

Formaliser l'hétérogénéité sémantique en logique de descriptions

Introduction

Pour traiter les problèmes d'hétérogénéité sémantique (HS) entre agents dialogiques, nous proposons de les considérer au niveau de la *représentation* des *bases* de *connaissances* (Kbase) des agents [101].

Il apparaît que la logique des descriptions (LD) [28], [71] est adaptée pour traiter les problèmes de gestion de *requêtes* entre deux représentations des connaissances sémantiquement hétérogènes. En effet, les LD sont déjà considérées comme un candidat naturel pour décrire les schémas en bases de données et les questionner à l'aide de requêtes.

De plus, elles sont étudiées en tant que formalisme de représentation pour les systèmes de gestion de bases de données [77]. Enfin, nous savons que les LD sont aussi adaptées lorsque les requêtes sont formulées par une personne qui n'est pas familière avec la base interrogée ou bien par une personne qui ne sait pas bien quelles questions sont appropriées [11].

Dans le problème qui nous intéresse, les LD vont nous permettre de *formaliser* les problèmes d'hétérogénéité sémantique entre agents dialogiques afin de proposer des *algorithmes* de résolution pour le projet InterViews. Nous serons amenés à modéliser les agents comme des Kbases LD dans ce chapitre [94].

Résumé du chapitre

Dans un premier temps, nous reprenons la définition des agents dialogiques formulée dans le chapitre I, page 27, que nous formalisons en LD. Nous en déduisons la définition en LD d'une interaction entre agents dialogiques.

Ensuite, nous posons les hypothèses de la gestion de l'hétérogénéité sémantique entre agents dialogiques modélisés en LD, avant de la définir formellement sous ces hypothèsess.

Enfin, nous synthétisons les deux principaux types de problèmes de HS qui peuvent émerger entre agents dialogiques en LD : la HS terminologique, et la HS référentielle. Nous les illustrons par des exemples d'interactions dialogiques, et nous proposons une stratégie de résolution pour chacun d'eux : l'adaptation pour la HS terminologique et l'accès par le contenu pour la HS référentielle.

III.1 Agents et interaction : définitions

III.1.1 Modéliser les agents dialogiques

III.1.1.1 Rappels sur les agents dialogiques

Les agents A_i dans InterViews sont des agents computationnels composés de :

- Une *ontologie* o_i : qui est une spécification explicite d'une *conceptualisation* [43]. Une conceptualisation est une *abstraction* à l'aide de concepts de l'univers connu de l'agent. Elle est une *spécification explicite* en ce qu'elle est représentée à l'aide d'un formalisme de représentation des connaissances.
- Une *mécanique* de fonctionnement f_i : qui est un ensemble de données *exécutables*.
- Un *comportement* c_i : qui est la *manifestation* de l'exécution des procédures de l'agent en utilisant les connaissances représentées dans l'ontologie.

Définition | *Nous définissons un agent A_i par le triplet :*
$A_i = \{o_i, f_i, c_i\}$

III.1.1.2 Ontologie modélisée par une Kbase en logique des descriptions

Pour résoudre l'hétérogénéité au niveau sémantique entre des agents dialogiques, nous nous intéressons à la partie qui définit la sémantique des termes connus de l'agent : son ontologie o_i. Ces ontologies sont modélisées par des bases de connaissances, notées KB_i écrites en LD. La modélisation de l'ontologie est alors composée de :

- une base terminologique, la TBox de la Kbase, notée Tb_i ,
- une base assertionnelle, la ABox de la Kbase, notée Ab_i.

Ainsi, dans ce document, chaque agent dialogique A_i est circonscrit à la Kbase :

$$KB_i = \{Tb_i, Ab_i\}$$

qui modélise son ontologie. Dans ce chapitre, nous parlerons désormais indifféramment de l'agent ou de sa modélisation en Kbase.

La question de la création de cette Kbase ne se pose pas (émergence, apprentissage, catégorisation), puisque les agents dialogiques sont définis et construits par des utilisateurs (voir chapitre I, page 27).

III.1.1.3 Hypothèses sur la Kbase d'un agent

Réciproquement, toute base KB_i écrite par un utilisateur, peut modéliser un agent dialogique moyennant quelques précautions.

Définition *Une Kbase en LD, KB_i, modélise un agent dialogique A_i ssi elle est consistante, c'est-à-dire que l'on ne peut pas démontrer l'absurde dans KB_i*

Ainsi, toute application écrite par un utilisateur est vue comme un agent dialogique pouvant interagir dans InterViews, dès lors qu'elle contient une partie ontologique définie par une Kbase consistante.

Nous verrons quelles conditions supplémentaires il faudra envisager pour que l'agent soit considéré comme entrant en interaction *dialogique* avec d'autres agents. Enfin, nous définirons dans quels cas cette interaction peut être qualifiée de sémantiquement *hétérogène*.

III.1.2 Interaction dialogique entre agents

Définition *Une interaction I est une séquence finie de n pas d'interaction I^i, elle est notée :* $I = [I_1, ..., I^n]$ où $n \in N$ avec $n < \infty$

III.1.2.1 Pas d'interaction

Un pas d'interaction I^i entre deux Kbases KB_1 et KB_2 est le couple $I^i = \{Q^i, R^i\}$ formé d'une requête Q^i et sa réponse *associée* R^i.

La Kbase qui *émet* la requête, est qualifiée *d'envoyeur*. La Kbase qui *reçoit* la requête est qualifiée de *receveur*. C'est le receveur qui émet la réponse associée à la requête. La réponse est *associée* en ce qu'elle constitue le *résultat* de l'action de la requête reçue sur la Kbase receveuse.

Dans la plupart des considérations qui suivent, on étudiera un *seul* pas d'interaction isolé i. Lors d'un pas, il est alors possible de discerner qui est la Kbase qui envoie la requête et qui la reçoit.

Nous noterons alors pour les exemples qui ne réfèrent qu'à un seul pas d'interaction :

- KB_e : pour la Kbase de l'agent *envoyeur*,
- KB_r : pour la Kbase de l'agent *receveur*.

Tandis que pour une interaction générale $I = [I^1, ..., I^n]$ nous conserverons les notations plus symétriques :

- KB_1
- KB_2

Pour le pas i on aura alors la définition suivante :

Définition *Un pas d'interaction I^i est défini par le couple :* $I^i = \{Q^i_{e \to r}, R^i_{r \to e}\}$

III.1.3 Requête et réponse

Dans notre modélisation en LD du problème, la requête et la réponse sont des *expressions* de logique des descriptions, tout comme les descriptions dans les Kbases modélisant les agents. Cependant, la requête contient en plus des *patterns* pour questionner à propos d'une connaissance. Ces patterns sont exprimés à l'aide de *variables libres* , comme dans les travaux de A. Borgida [13].

III.1.3.1 Typage des variables libres

Pour questionner à propos d'un *individu*, un *concept*, ou un *rôle*, les patterns s'écrivent à l'aide de variables libres, notées :

- pour un individu : $\#?x$
- pour un concept : $?X$
- pour un rôle : $?x$

Si bien qu'une requête Q est une expression de LD utilisant l'alphabet usuel de la LD, plus les trois symboles ci-dessus en respectant la syntaxe de la LD choisie (chaque type de variable doit respecter les constructions autorisées pour ce type).

III.1.3.2 Notation

On pourra avoir besoin de préciser la dépendance d'une requête vis-à-vis de ses variables, on le note comme suit :

- $Q(\#x?) = (Humain\ \#?x)$ par exemple si l'on questionne au sujet d'un individu $\#x$ instance du concept $Humain$.
- Ou bien $Q(\#?x, ?X) = (?X\ \#?x)$ pour questionner au sujet de tous les individus en demandant de quel concept ils sont des instances.

On complète cette notation par les indices appropriés désignant le pas d'interaction et les Kbases concernées comme défini plus haut.

III.1.3.3 Cohérence avec la Kbase de l'envoyeur

D'après la définition donnée aux requêtes dialogiques (voir chapitre I), une Kbase ne peut formuler des requêtes que dans le vocabulaire qu'elle connaît. Le vocabulaire connu par un agent A_i est celui qui est défini dans la terminologie Tb_i de la Kbase KB_i qui le modélise. Ainsi, à un pas d'interaction quelconque i, l'envoyeur KB_e ne peut formuler vers le receveur KB_r, que les requêtes $Q^i_{e\to r}$ qui sont *consistantes* avec sa *propre* terminologie.

Définition | *Une requête envoyée par un agent dialogique vérifie :*
$KB_e \models Q^i_{e\to r}$

III.2 Un pas d'interaction entre deux agents hétérogènes

Nous posons et argumentons dans cette section le *cadre* de la *gestion* de la HS entre agents dialogiques, afin de *restreindre* la définition de la HS entre agents seulement à sa gestion :

1. *Locale :* pour ne pas se ramener à un cas d'homogénéité ;
2. Par la Kbase qui *reçoit* la requête : car la requête émise n'est pas hétérogène pour l'envoyeur ;
3. Lors *d'un seul pas* d'interaction : pour ne pas ramener la résolution à des coopérations qui seraient gérées par un protocole.

III.2.1 Notion de point de vue pour la gestion

L'hétérogénéité sémantique entre deux Kbases KB_1 et KB_2 peut se résoudre selon deux approches :

- Utilisation d'un point de vue global sur les deux KB_i ;
- Utilisation d'un point de vue local dans chaque KB_i.

III.2.1.1 Point de vue global

Un point de vue global est un point de vue *externe et unique* sur les deux Kbases à la fois. Dans ce type de point de vue, on "gomme" la HS en la ramenant à de *l'homogénéité* : on *crée* une Kbase *abstraite*, plus *large*, *englobant* de manière cohérente les deux Kbases des agents. Cette Kbase, notée $KB_{(1,2)}$, est qualifiée de *globale*, par opposition aux Kbases KB_1 et KB_2 qui sont alors qualifiées de *locales*. La Kbase globale peut être :

Construite en *intégrant* les données des deux Kbases locales et en s'assurant de leur *cohérence* mutuelle : c'est la proposition d'une *ontologie unique* [20]. La Kbase globale est alors *explicite*, elle est le modèle de l'ontologie unique. Cette voie est beaucoup explorée en bases de données hétérogènes où l'on cherche à *intégrer* des données ayant des schémas différents [54].

Simulée en *traduisant* les données de chaque Kbase à l'aide d'un *dictionnaire*. Ce dictionnaire est une *table* de *correspondance un-à-un* entre les *termes* de chaque Kbase locale ; elle ne peut donc être conçue que par un observateur *extérieur*. La Kbase globale est dans ce cas *implicite* car elle a servi à produire le dictionnaire et la connaissance de la Kbase globale est *distribuée* sur le dictionnaire. Cette proposition correspond à l'emploi *d'agents médiateurs* par exemple [19].

III.2.1.2 Point de vue local

Gérer la HS en utilisant un point de vue *local*, correspond à *s'interdire* de recourir à un point de vue global. Cette solution peut être adoptée pour plusieurs raisons :

1. La Kbase globale est difficile, coûteuse ou impossible à construire [54] : pour des raisons d'incohérence entre les données, de temps, de taille.
2. Le dictionnaire est impossible à écrire : pas de correspondance un-à-un possible par exemple.
3. L'application *requiert* un traitement au niveau local.

Dans InterViews, on se trouve toujours au moins dans le troisième cas (voir description scientifique du projet, chapitre I, page 24) pour les raisons suivantes :

1. On ne peut pas créer des dictionnaires en-ligne en fonction des applets enregistrées dans le système en temps réel ;
2. Les agents computationnels ne seraient plus libres *d'évoluer* ou d'être modifiées (par d'autres agents ou par l'utilisateur) *dynamiquement* ;
3. La *taille* de la Kbase globale nécessaire est imprédictible, puisque le nombre d'applets interagissant à un instant donnée n'est pas contrôlé (dans le cas idéal du moins) ;
4. La *cohérence* d'une telle Kbase globale serait toujours remise en cause ;
5. La HS peut être résolue entre les agents pris *deux* à *deux*, mais insoluble si on les considère *tous* à la fois, il faudrait donc une Kbase globale par paire d'agents interagissant ;
6. Dans le cas où un des agents dialogique est un *utilisateur humain*, on ne peut jamais prédire la Kbase qui "modélise" cet humain : les termes qu'il connaît, les sens différents et ambigus qu'il leur accorde.

III.2.2 "Résoudre" *versus* "gérer" l'hétérogénéité sémantique

III.2.2.1 Point de vue global : résolution

Dans le cas d'un point de vue global, la HS est *résolue.*

En effet, la mise en oeuvre du point de vue global implique un traitement avant l'interaction proprement dite. Ce traitement consiste en la création soit de l'ontologie globale, soit du médiateur. Mais dans les deux cas, les Kbases hétérogènes sont ramenées à une Kbase globale *homogène.*

Ainsi, avec l'utilisation d'un point de vue global, lorsque l'interaction démarre, la Kbase qui reçoit la requête est confrontée à une requête qui lui est sémantiquement *homogène,* elle n'a plus d'hétérogénéité à gérer.

En résumé, lorsqu'on résoud la HS avec un point de vue global, on la gère complètement *avant* l'interaction. Au niveau des agents qui interagissent ensuite, elle est *résolue.*

III.2.2.2 Point de vue local : gestion

Dans le cas d'un point de vue local, il n'y a pas de traitement préalable avant l'interaction proprement dite. La Kbase qui reçoit la requête, reçoit une expression qui est issue d'une autre Kbase *hétérogène.*

La requête envoyée par KB_e au pas d'interaction i est hétérogène, et il s'agit alors pour KB_r, de produire une réponse *malgré* l'hétérogénéité de la requête, on dit que Kb_r *gère* la HS.

III.2.2.3 Qu'est-ce que gérer l'hétérogénéité sémantique ?

Pour répondre à une requête, toute Kbase écrite en LD doit pouvoir en trouver une *interprétation* (voir chapitre II, table II.2.2.2.c). Notons ϵ_i la fonction d'interprétation de la Kbase KB_i.

Définition | *On dit que la Kbase KB_r gère une requête hétérogène $Q^i_{e \to r}$, si elle associe une interprétation : $\epsilon_r(Q_{e \to r})$ à cette requête, où ϵ_r est la fonction d'interprétation de KB_r sans avoir accès ni à la fonction d'interprétation, ni la Kbase de l'envoyeur.*

Notons que la Kbase KB_r peut trouver :

1. Aucune interprétation : dans ce cas, il lui sera impossible de construire une réponse (autre que "échec") ;
2. Une seule interprétation : dans ce cas, il lui sera possible de construire une réponse. Cette réponse peut être celle attendue par KB_e ou non, mais KB_r ne peut pas en avoir connaissance ;
3. Plusieurs inteprétations : dans ce cas, il lui faudra en choisir une. Soit elle est en mesure de le faire, et elle construira sa réponse avec des critères de compétition par exemple. Soit elle ne peut le faire, dans ce cas la réponse sera encore un "échec".

Avant d'aller plus loin, nous devons définir ce que nous entendons par un "échec".

III.2.3 Echec d'une interaction sémantiquement hétérogène

Selon Hirst [50], les problèmes sémantiques peuvent être classés en deux catégories, selon qu'ils sont *détectés* ou non par l'agent qui en est à l'origine :

- Not-Understanding Problem (NUP) ;
- Misunderstanding Problem (MUP).

Ces problèmes vont se traduire ou non par un échec de l'interaction. Nous définissons maintenant ces deux types de problèmes.

III.2.3.1 Not-understanding problem

Le Not-Understanding Problem (NUP) est l'échec à un pas d'interaction i du receveur KB_r à trouver *une* interprétation *valide* à la requête $Q^i_{e \to r}$. Comme nous l'avons dit, il se peut qu'il n'en trouve *aucune* ou bien qu'il en trouve *plusieurs* et qu'il soit incapable de décider laquelle est valide. Dans tous les cas le participant qui échoue, en est "conscient" : le problème est *détecté* par lui.

Il se formalise par :

$$NUP(i) ::= Card\{\epsilon_r(Q^i_{e \to r})\} \neq 1$$

où ϵ_r est la fonction d'interprétation de KB_r.

Concrètement, voici un exemple d'interaction avec NUP. Soient deux Kbases qui ont les descriptions suivantes :

	KB_r
TBox :	Stagiaire $\subseteq$Humain
	Thésard $\subseteq$Stagiaire
	DEA $\subseteq$Stagiaire
	$Humain \doteq \exists numeroTel \wedge \forall numeroTel.Integer$
ABox :	(Thésard #paul)
	(DEA #william)

	KB_e
TBox :	CDD$\subseteq$Humain
	Thésard $\subseteq$CDD
	Stagiaire $\subseteq$CDD
	DEA $\subseteq$Stagiaire
	$Humain \doteq \exists numeroTel \wedge \forall numeroTel.Integer$
	$numeroTel \subseteq numero$
ABox :	(Thésard #paul)
	(Stagiaire #bill)

KB_e envoie une requête qui demande le numéro de téléphone de *bill* :

$Q^1_{e \to r}$=?x | (*numeroTel #bill x*)

puisque *#bill* n'est pas connu de la ABox de KB_r, cette Kbase ne sera pas capable de trouver une interprétation pour cette requête.

cela aurait pu arriver aussi, si KB_e demandait le numéro de téléphone de Paul en employant seulement la désignation “numéro” :

$Q^2_{e \to r}$=?x | (*numero #paul x*)

cette fois, KB_r ne pourrait pas trouver d'interprétation à cause du rôle *numero* qu'il ne connait pas.

Nous verrons dans la section III.2.3.3 quelles solutions nous proposons, pour que KB_r puisse répondre tout de même quelque chose de pertinent dans ces cas de NUP.

III.2.3.2 Misunderstanding problem

Le Misunderstanding Problem (MUP) arrive quand un participant KB_e ou KB_r trouve *une* interprétation qu'il considère comme valide par rapport à sa Kbase locale, pour la requête $Q^i_{e \to r}$. Si la validité de cette interprétation doit être remise en cause par l'envoyeur de la requête KB_e, cela ne pourra se faire que si KB_e détecte une anomalie. Mais cela ne sera détectable que si cela se traduit par un échec à un pas d'interaction *ultérieur*, j ; c'est-à-dire si le MUP a engendré un NUP, plus tard dans la conversation. Cela se formalise par :

$$MUP(i) ::= \exists j > i \mid NUP(j)$$

Dans ce cas, nous sommes ramenés à traiter le NUP à un pas j.

Concrètement, voici un exemple d'interaction avec MUP. Reprenons les deux Kbases de l'exemple précédent. Soit la requête :

$Q^3_{e \to r}$=?x | ($numeroTel$ $\#y$ x) $\wedge$($Stagiaire$ y)

qui se traduit par "quel est le numéro de téléphone du stagiaire ?". Cette fois, on ne pose pas la question en référant directement au nom du stagiaire (dans la Abox de KB_e), qui est $\#bill$, mais par rapport à son type. L'agent KBr peut alors trouver une interprétation pour cette requête puisqu'il possède le concept de $Stagiaire$ dans sa TBox. Cependant, puisqu'il n'a pas la même TBox que KB_e, l'interprétation de KB_r peut associer par exemple $\#paul$ à $\#y$ dans cette requête, puisque pour KB_r on a bien ($Stagiaire$ $\#paul$).

La réponse associée n'est pas alors "fausse", mais elle ne correspond pas à la réponse attendue par KB_e.

Se pose alors le problème de la détection de ce problème. Dans cet exemple, KB_r va effectivement retourner un numéro de téléphone (celui de $\#paul$), et KB_e ne pourra pas détecter qu'il y a une erreur de compréhension : c'est un cas de MUP. Ce MUP ne sera détecté que s'il produit un NUP à un pas ultérieur. Par exemple, si KB_e utilise ce numéro de téléphone dans la suite de l'interaction pour faire référence à d'autres informations, qui finiront par trahir le MUP (sans dévoiler le pas d'interaction auquel il est né, ici $i = 3$).

III.2.3.3 Gérer l'hétérogénéité à l'échelle d'une requête

Que le NUP à résoudre soit la conséquence d'un MUP ou pas, il doit se résoudre au moment où il se révèle, au niveau *d'une seule* requête. Dans ce cas, KB_r doit gèrer la HS contenue dans la requête $Q^i_{e \to r}$, lors d'un seul pas d'interaction i, pour pouvoir produire la réponse $R^i_{r \to e}$ associée.

La HS se traduit alors par des *termes* hétérogènes dans l'expression de $Q^i_{e \to r}$. *Gérer* un *terme* hétérogène correspond à trouver une *interprétation* locale pour ce terme inconnu dans la Kbase du receveur.

Notre proposition est que cette interprétation peut être trouvée en appliquant des outils du raisonnement par analogie. Nous développons cette proposition dans la deuxième partie de la thèse.

III.2.4 Une interaction sémantiquement hétérogène

Après avoir listé les modes de gestion de la HS (globale ou locale), ainsi que les types de problèmes rencontrés dans une interaction avec HS (MUP et NUP), nous explorons les causes de la HS. Pour cela on fait les hypothèses (voir page 77) :

1. La gestion de la HS est locale ;
2. Elle se fait au niveau d'un seul pas d'interaction ;
3. Il y a un NUP au pas d'interaction étudié.

Dans ces conditions, nous définissons ce qu'est l'hétérogénéité, avant de détailler les implications que cela entraîne au niveau des Kbases.

III.2.4.1 Définitions

Une interaction I entre deux Kbases KB_1 et KB_2 est qualifié *d'hétérogène* si la requête produite par KB_1 est *sémantiquement hétérogène* par rapport à KB_2.

Considérons la définition que Weinstein [101] propose pour l'hétérogénéité sémantique pour une unique terminologie :

Définition La terminologie comprend des symboles qui ont des dénotations *inconsistantes* le même symbole référence des objets différents ;
la terminologie comprend des dénotations *redondantes* : des symboles différents référencent le même objet ; la terminologie comprend les *deux*.

En appliquant ce point de vue au problème d'une requête par rapport à une Kbase, on pose la définition suivante :

Définition *La requête $Q^i_{e\to r}$est sémantiquement hétérogène par rapport à KB_r si l'union : $Q^i_{e\to r} \cup KB_r$ est inconsistante.*

Puisque la requête $Q^i_{e\to r}$ a été produite par KB_e, on a par définition : $KB_e \models Q^i_{e\to r}$.

La définition précédente implique alors la définition d'une interaction hétérogène (interaction globale et non un *pas* d'interaction) :

Définition *Un pas d'interaction i entre deux Kbases KB_1 et KB_2 est qualifié d'hétérogène si l'union :$KB_{1\cup 2} = KB_1 \cup KB_2$*
des deux Kbases qui sont impliquées dans ce pas d'interaction résulte en une Kbase inconsistante.

L'interaction sera donc qualifiée de HS dès lors qu'elle fait intervenir deux Kbases hétérogènes sémantiquement au sens de Weinstein. Qu'une requête $Q^i_{1\to 2}$, considérée à un pas d'interaction i particulier, soit hétérogène ou pas, n'est pas pertinent. Une interaction hétérogène peut être constituée de requêtes qui ne sont pas toutes strictement hétérogènes au sens de Weinstein. Mais nous verrons que le fait que les deux Kbases soient hétérogènes, aura des conséquences sur la gestion des requêtes par la Kbase receveur, même si la requête elle-même n'est pas strictement hétérogène au pas i.

III.2.4.2 Union inconsistante des Kbases en interaction dialogique

Cette inconsistance de $KB_{1\cup 2}$ aura lieu ssi les Kbases :

1. sont *différentes : $KB_1 \neq KB_2$*
2. et qu'elles *partagent* un domaine.

III.2.4.2.a Les Kbases doivent être différentes En effet, si les Kbases sont égales, on a :

$$KB_1 = KB_2$$

Mais comme les Kbases modélisent des agents dialogiques, on a par définition KB_1 consistante. Alors :

$$KB_{1\cup 2} = KB_1 \cup KB_2 = KB_1 \cup KB_1 = KB_1$$

qui est consistante aussi. L'interaction n'est donc pas hétérogène (définition ci-dessus).

Il est donc nécessaire que $KB_1 \neq KB_2$

III.2.4.2.b Les Kbases doivent partager la connaissance d'au moins un terme Si les Kbases ne partagent pas de description à propos d'au moins un terme, il n'y a pas d'incohérence possible non plus.

En effet une base incohérente est une base dans laquelle on peut démontrer l'absurde $\perp$. Pour cela, il est nécessaire d'arriver à démontrer un fait, notons le A par exemple, et son contraire $\neg A$.

Supposons que le fait A soit démontrable dans KB_1 :

$$KB_1 \models A$$

alors $\neg A$ n'est pas démontrable dans KB_1 puisque c'est une Kbase qui modélise un agent dialogique et elle est donc consistante. L'assertion $\neg A$ doit donc être démontrable dans KB_2 :

$$KB_2 \models \neg A$$

et pour cela il faut que le terme A soit connu dans la Kbase KB_2, même si c'est pour asserter ou démontrer son contraire.

Le raisonnement est le même si l'on suppose au début que $KB_1 \models \neg A$

Ainsi, les deux Kbases partagent au moins la connaissance de faits à propos du terme A dans leurs ABoxes (défini comme étant le terme qui sert à démontrer l'inconsistance de leur union) ; et donc la connaissance définitoire à propos du terme A dans leurs TBoxes.

III.2.4.2.c Cas d'interaction sans communication Concrètement, une situation où les Kbases ne partagent pas la connaissance d'au moins un terme, est une situation où l'interaction ne se traduira pas par une *communication*.

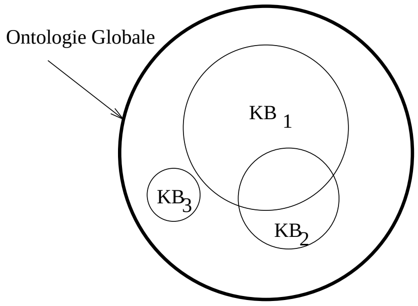
III.2.4.3 Conséquence sur l'intersection des Kbases

Ces deux conditions impliquent une *intersection* non nulle entre les domaines d'interprétation des Kbases KB_1 et KB_2 :

$$\epsilon_1(KB_1) \cap \epsilon_2(KB_2) \neq \emptyset$$

où $\epsilon_i(KB_i)$ désigne le domaine d'interprétation de la Kbase. Il s'agit de la fonction d'interprétation sur toute la base (voir chapitre II).

FIG. III.1 – **Cas a** : Cas d'intersection de domaines correspondant à la proposition d'un ontologie globale.



III.2.5 Implications sur les domaines d'interprétation des Kbases hétérogènes

III.2.5.1 Intersection de domaines

En considérant l'intersection $\epsilon_1(KB_1)\cap\epsilon_2(KB_2)$, nous donnons une classification des situations interactionnelles que l'on peut rencontrer en quatre cas.

III.2.5.1.a a - Ontologie globale Le cas illustré par la figure III.1 correspond à la proposition d'une ontologie globale. Dans ce cas, nous avons :

$$\exists KB_{globale} \mid \forall i,\ \epsilon_i(KB_i)\subseteq\epsilon_{globale}(KB_{globale})$$

Et il s'en suit que l'intersection de domaines utile pour résoudre la HS, est toujours l'intersection d'une Kbase avec cette $KB_{globale}$; et l'on a :

$$\forall i,\ \epsilon_i(KB)\cap\epsilon_{globale}(KB_{globale})= \epsilon_{globale}(KB_{globale}).$$

Nous avons déjà rejeté ce cas, qui correspond à construire $KB_{globale}$ en intégrant toutes les KB_i, et en gommant l'hétérogénéité (voir paragraphe sur le traitement local, page 77).

III.2.5.1.b b - Pas de communication Le cas illustré par la figure III.2 correspond à toutes les situations où :

$$\forall i,j,\ \text{avec } i\neq j,\ KB_i\cap KB_j=\emptyset$$

III.2.5.1.c c - Une Kbase commune : la Base Le cas illustré par la figure III.3 correspond à une Kbase commune à toutes les Kbases des agents. On appelle cette Kbase, la *Base* (ou *Ground* en anglais) , elle est notée KB_{base}. C'est elle qui sert de *médiateur* dans le cas de la construction d'un dictionnaire. On a :

$$\forall i,\ KB_i\cap KB_{base}\neq\emptyset$$

Dans ce cas, on pose l'existence d'une terminologie de base qui représente le *sous-domaine* d'expertise *commun* dans le cas où les agents sont orientés vers le *même* domaine d'application. Mais cette condition est en général *trop forte* lorsque les agents ne sont pas strictement orientés vers la même application. Enfin, constituer une telle base est difficile à mettre en oeuvre en pratique.

Fig. III.2 – **Cas b** : Cas d'intersection de domaines correspondant à une interaction ne pouvant mener à aucune communication.

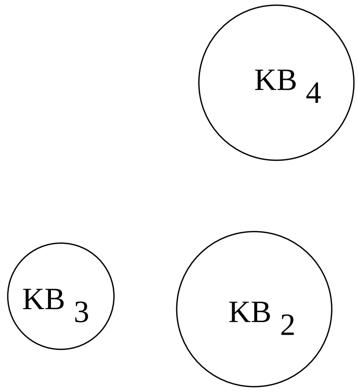


Fig. III.3 – **Cas c** : Cas d'intersection de domaines correspondant à la proposition d'une Kbase commune : la Base.

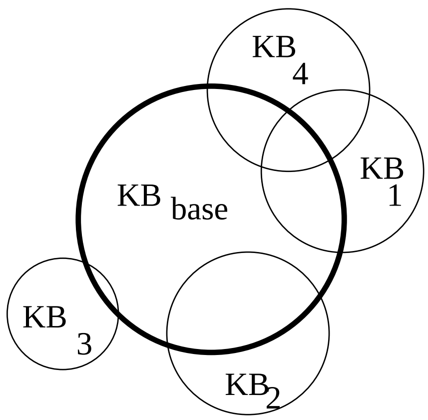
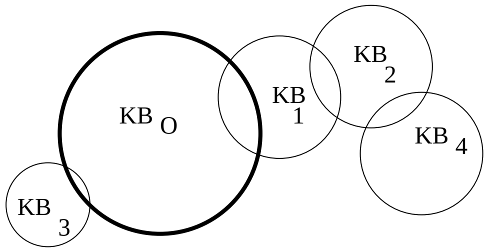


FIG. III.4 – **Cas d** : Cas d'intersection de domaines correspondant à la proposition d'une Kbase initiatrice : le *germe,* noté KB_0.



III.2.5.1.d d - Une Kbase initiatrice : le Germe Le cas illustré par la figure III.4, correspond à la proposition d'un agent *Germe* (*Seed* en anglais), noté KB_0, qui implique une interaction qui se déroule comme suit :

1. Au premier pas d'interaction, il existe un agent-germe ;
2. A chaque pas d'interaction suivant, tout agent entrant dans le système pour interagir partage au moins un sous-domaine avec :
   - soit le germe,
   - soit avec n'importe quel autre agent déjà enregistré dans le système , ce qui s'écrit :
   $\forall i$, $i > 1$ soit $(KB_i \cap KB_0 \neq \emptyset)$, soit $(\exists j,\ j \neq i,\ KB_i \cap KB_j \neq \emptyset)$

III.2.5.2 Différences de domaines

Le fait que $\epsilon_1(KB_1) \cap \epsilon_2(KB_2) \neq \emptyset$, implique qu'il existe des différences, et des ressemblances entre les domaines des Kbases.

Les sortes de *différences* entre ontologies ont déjà été analysées par Campbell [19] qui distingue les différences de :

Nommage hétérogénéité provenant de l'utilisation d'un même nom pour des objets différents ou de noms différents pour un même objet. Ce problème est recouvert par la question plus générale de la *référence.* Nous parlerons maintenant d'hétérogénéité sémantique de type référentielle.

Structure hétérogénéité provenant de la syntaxe. Cette hétérogénéité est écartée par le fait que nous utilisons le même formalisme pour exprimer toutes les Kbases ainsi que les requêtes et les réponses. Nous avons déjà mentionné au chapitre I que l'hétérogénéité sémantique pouvait s'envisager au niveau du vocabulaire et/ou de la grammaire. Nous avons déjà restreint le problème des agents dialogiques hétérogènes à une hétérogénéité de vocabulaire purement.

Taxonomie hétérogénéité provenant de la structuration en hiérarchie du domaine. Nous parlerons désormais d'hétérogénéité sémantique de type taxonomique.

Expressivité hétérogénéité provenant du langage de représentation. Comme pour la structure, nous écartons ce type d'hétérogénéité grâce à l'utilisation d'un même langage de formalisation pour les Kbases et les requêtes.

Les différences de structure syntaxique et d'expressivité ne sont pas pertinentes ici puisque nous avons supposé que les deux ontologies étaient exprimées sou formes de terminologies LD, ce qui nous ramène à n'envisager que les deux cas d'hétérogénéité sémantique suivants :

1. HS Taxonomique ;
2. HS Référentielle.

Nous avons défini les notions d'hétérogénéité sémantique, d'interaction, d'agent dialogique ; nous avons listé tous les cas de figure pour des interactions hétérogènes entre agents dialogiques. Nous proposons de nous restreindre à une gestion locale au cours d'un seul pas d'interaction par la base de connaissances réceptrice de la requête hétérogène. Nous examinons maintenant les deux types de HS que nos avons dégagés.

Ainsi, l'intersection entre les domaines des Kbases, peut se traduire par une intersection entre :

TBoxes : correspond à une HS terminologique, c'est-à-dire une hétérogénéité concernant des connaissances *terminologiques*

ABoxes, correspond à une HS référentielle, c'est-à-dire une hétérogénéité concernant la *référence* à des *individus* effectifs, implique donc des connaissances *assertionnelles* ou bien de définition de domaines dans les terminologies.

III.3 Etude d'un exemple d'interaction hétérogène entre deux Kbases

III.3.1 Préliminaires : gestion de l'interaction

Avant d'étudier des exemples, nous esquissons les algorithmes de traitement de l'interaction. Puis nous présentons les deux Kbases qui vont entrer en interaction dialogique, elle sont hétérogènes.

III.3.1.1 Algorithmes

Pour le traitement d'une interaction, nous distinguons les fonctions :

A1 : de traitement du cycle d'interaction ;

A2 : de traitement d'un pas d'interaction ;

A3 : de gestion de la requête ;

A4 : de gestion des hétérogénéités dans la requête.

III.3.1.1.a A1 : Gestion de l'interaction.

L'interaction est gérée de manière basique selon le schéma suivant :

	`GererInteraction(KB1,KB2)`
pas d'interaction :	`incr i`
initialisations :	`initi(KBe,KB1)`
	`initi(KBr,KB2)`
appel de fonction :	`GestionPasInteraction(i,KBe,KBr)`
fin :	`Validation`

III.3.1.1.b A2 : Gestion d'un pas d'interaction

Un pas d'interaction est géré selon l'algorithme suivant :

	`GererPasInteraction(i,KBe,KBr)`
émission de requête :	`Q=EmissionRequete(i,Requete(KBe))`
gestion de requête :	`R=GererRequete(Q,KBr)`

III.3.1.1.c A3 : Gestion de la requête

Une requête est géré par la Kbase réceptrice selon l'algorithme suivant :

	`GererRequete(Q,KBr)`
fct sur le nb de patterns de Q :	`for a=0 to Nb_patterns(Q)`
test de type sur patterns Q :	`T=TypePattern(a,Q)`
cas terminologique :	`if T in (concept,role) P=GererPatternTbox`
cas référentiel :	`else if in (individu) P=GererPatternAbox`
production d'une réponse	`R= ProduireReponse(P)`

Les patterns sont gérés par des fonctions qui appliquent soit des algorithmes de matching, soit des algorithmes d'adaptation dans le cas de termes hétérogènes.

III.3.1.1.d A4 : Gestion de l'hétérogénéité sémantique

Les algorithmes de matching et d'adaptation pour gérer la HS seront présentés dans la Partie 3 de la thèse, à l'aide des outils pour l'analogie que nous proposerons dans la Partie 2.

Quel que soit le fonctionnement interne de ces fonctions, nous supposons que nous disposons d'un algorithme d'adaptation complet, et nous traitons maintenant un exemple pour chaque type de HS. Les gestions de requêtes utilisent les fonctions et algorithmes définis ci-dessus.

III.3.1.2 Requêtes possibles

Nous rappelons que les requêtes sont des expressions de LD (voir chapitre II) contenant des *patterns*. Les patterns peuvent porter sur les types suivants :

- concept, noté $?X$;
- rôle, noté $?r$;

TAB. III.1 – Syntaxe générale des requêtes possibles pour le langage AL (voir chapitre II, page 67 pour une définition).

$$
\begin{array}{rl}
Q \rightarrow T_i & ::= ?X \mid \\
 & [constructor\ role\ ?X] \mid \\
 & [constructor\ ?r\ Concept] \mid \\
 & [constructor\ ?r\ ?X] \mid \\
 & [and\ ?X\ Concept] \mid \\
 & [not\ ?X] \\
?X & ::= \text{variable libre de concept} \\
constructor & ::= and \mid atleast \mid atmost \\
?r & ::= \text{variavle libre de rôle}
\end{array}
$$

– individu, noté $?x$.

La syntaxe complète de ces requêtes dépend donc du langage de LD choisi. Pour le langage AL, elle est donnée par la table III.1.

III.3.2 Exemple abstrait : deux agents dialogiques

Les Kbases et les requêtes présentées dans cet exemple respectent la syntaxe du langage de description AL (voir page 67) et du langage de requête associé à AL et défini ci-dessus (voir table III.1). Cet exemple sera repris dans le chapitre VIII, et sera traité avec les outils que nous développons dans cette thèse. En ce qui concerne ce chapitre-ci, nous illustrons simplement le sous-problème que nous devons résoudre afin de gérer l'hétérogénéité sémantique entre agents dialogiques modélisés par des bases de connaissances écrites en logique de descriptions.

Soient Tb_1 et Tb_2 les deux terminologies des deux Kbases KB_1 et KB_2 :

$$
\begin{array}{rl|rl}
Tb_1 = & \{A \doteq E \wedge C & Tb_2 = & \{F \doteq \forall s.C \\
 & D \doteq \forall r.C & & G \leq C \\
 & B \leq E & & E \leq G \wedge F \\
 & A \leq C\} & & A \doteq B \wedge C \\
 & & & A \leq C\}
\end{array}
$$

III.3.2.1 Hétérogénéité sémantique terminologique

La HS terminologique concerne les TBoxes des Kbases en LD. Les TBoxes des deux Kbases présentées ci-dessus sont hétérogènes. Mais la requête envoyée par une Kbase à un pas d'interaction particulier peut être :

1. Non hétérogène dans les termes qu'elle contient (pouvant conduire à un MUP) ;
2. Hétérogèné explicitement dans les termes qu'elle contient (au sens de Weinstein).

Ces deux cas font cependant partie de l'étude de l'interaction terminologiquement hétérogène. Nous allons voir que même dans le cas de la gestion d'une requête non hétérogène, dès lors que les TBoxes sont hétérogènes, il y a des conséquences pour la TBox réceptrice.

III.3.2.2 Requête non-hétérogène : réduction des mondes possibles

Plaçons-nous à un pas d'interaction i quelconque. La fonction d'initialisation de `GererInteraction`, fixe le rôle des Kbases envoyeur et récepteur :

– pour KB_e l'agent A_1 (qui est donc l'envoyeur pendant ce pas),
– pour KB_r l'agent A_2 (qui est donc le receveur pour i)

De même, la fonction de gestion du pas d'interaction `GererPasInteraction` débute avec les arguments $(i,\, KB_e,\, KB_r)$.

Soit alors l'émission de requête $Q^i_{1\to2}$:

$$KB_1 \models Q^i_{1\to2} = (A \doteq E \wedge ?X)$$

La fonction de gestion de requête `GererRequete` doit retourner une interprétation de $Q^i_{1\to2}$ dans la Kbase KB_2. Détaillons le raisonnement qui est nécessaire pour trouver cette interprétation :

1. On constate que A est un concept connu de KB_2, il n'est donc pas sémantiquement hétérogène.
2. La définition locale de A dans Tb_2 est : $A \doteq B \wedge C$. Cette définition est pourtant différente de celle que KB_1 possède localement dans Tb_1 pour le même symbole de concept : A.
3. On apparie l'expression $A \doteq E \wedge ?X$ avec $A \doteq B \wedge C$; où E est aussi un concept connu de Tb_2 avec la définition $E \leq G \wedge F$ où $G \leq C$ et donc $E \leq C$.
4. Pour répondre à $Q^1_{1\to2}$, il faut trouver un concept X tel que son intersection avec E, $X \cap E$, dénote *exactement* le même ensemble que celui dénoté par $B \cap C$, en sachant $E \leq C$.
5. Le concept X est B ssi $B \leq E$.
6. L'agent KB_2 donne pour réponse à $Q_{1\to2}$, la substitution valide suivante : $R^i_{2\to1} = [X \to B]$

En même temps que la production de cette réponse, KB_2 peut *informer* KB_1 de l'inférence établie : $B \leq E$. Si cette communication a lieu, l'information $B \leq E$ peut être utilisée à un pas d'interaction ultérieur pour détecter des MUP éventuels. La réponse fournie par KB_r est alors : $R^i_{2\to1} = [X \to B, with(B \leq E)]$ par exemple.

Dans tous les cas, l'effet de l'interaction de KB_2 avec KB_1 est *d'ajouter* une *information* (sous forme de contrainte) dans Tb_2 et donc de *réduire* le nombre de modèles *possibles* pour Tb_2.

III.3.2.3 Requête hétérogène

Considérons à présent le pas d'interaction j avec $j > i$, on a toujours $KB_e = KB_1$ pour ce pas.

Gérons la nouvelle requête posée par KB_1 à KB_2 :

$$Q^j_{1\to2} = ?X(\forall r.C)$$

La fonction de gestion de la requête, `GererRequete`, essaiera d'abord d'apparier de pattern de la requête avec la définition locale :

$$F \doteq \forall s.C$$

Ce matching ne pourra réussir avec la substitution $X \to F$, que si nous supposons *aussi* que $r \to s$. Mais nous savons que r est sensé être un rôle *constant* dans la requête telle qu'elle est envoyée, et non un pattern.

Dans ce cas, nous proposons *d'adapter* ce terme dans la requête pour lui donner un sens connu pour KB_2 dans Tb_2. On note une telle adaptation avec l'opérateur $\rightharpoondown$. Cette adaptation sera effectuée par une autre fonction que nous présenterons dans la troisième partie, comme mentionné plus haut.

Ainsi, la substitution valide donnée par KB_2 comme réponse sera :

$$R^j_{2\to 1} = (X \to F, r \rightharpoondown s)$$

III.3.2.4 Proposition pour gérer l'hétérogénéité de termes

Pour synthétiser, lors d'une interaction sémantiquement hétérogène pour les terminologies, on peut trouver deux cas de figure pour un pas d'interaction quelconque :

1. La requête n'est pas hétérogène elle-même ;
2. La requête est hétérogène.

Dans le cas d'une requête non-hétérogène, nous proposons de procéder à un pattern matching comme pour le traitement classique des requêtes dans les bases de données en LD [11], à cela près que des déductions supplémentaires peuvent être possibles, ayant des conséquences au niveau de la mise à jour de la TBox réceptrice.

La proposition que nous faisons pour résoudre l'hétérogénéité sémantique des *termes* contenus dans les requêtes, à un pas d'interaction, est donc la suivante :

- *Matcher* les termes *inconnus* avec les termes de la Kbase *locale*, comme on le fait pour des *variables* libres dans une requête normale à une base de données simple [13]. Cela peut entraîner un raisonnement, et même nécessiter des hypothèses supplémentaires. Ces hypothèses supplémentaires sont considérées comme des informations déduites par l'agent receveur et réduisant ainsi le nombre de modèles possibles de sa Kbase KB_r.
- *Adapt*er les termes matchés à l'aide de transformations élémentaires qui respectent le type des termes appariés. Cette adaptation se fait en utilisant des *analogies* avec la *structure* des connaissances contenues dans la Kbase locale (voir chapitre suivant, partie 2).

Ces deux opérations sont des opérations de base classiquement considérées dans le modèle de raisonnement par analogies. Les outils que nous souhaitons développer pour résoudre la HS entre agents dialogiques doivent donc s'inspirer des algorithmes de traitement automatique des problèmes d'analogies que nous pouvons étudier en Intelligence Artificielle. C'est ce que nous faisons dans le chapitre IV.

Le fait *d'adapter* un terme constant d'une requête est une notion *cruciale* pour résoudre l'hétérogénéité sémantique. Nous pensons que cette adaptation doit s'appuyer sur les relations de subsomption de la même manière que les substitutions valides lors des matchings, par exemple dans [7], [13].

III.3.3 Hétérogénéité de type référentielle

Les identifieurs dans les ABoxes des Kbases en interactions peuvent être *locaux* ou *globaux*.

III.3.3.1 Identifieurs globaux

Dans ce cas, les identifieurs dénotent un même objet qu'ils soient dans l'un ou l'autre des ABoxes.

Ceci est vrai pour tout agent enregistré dans le système. Ainsi chaque Kbase est réduite à être simplement une vue différente sur la même base [22]. La question est alors de savoir s'il est possible d'avoir différentes théories du même domaine. Mais ce cas est à mettre en correspondance avec les traitements globaux de la HS entre domaines, déjà écartés dans ce chapitre.

III.3.3.2 Identifieurs locaux

Dans ce cas, chaque agent utilise des identifieurs locaux pour référer à des objets de l'environnement *commun* en fonction de leur terminologie personnelle. Dans ce cas, il ne peut pas y avoir de *conflit* entre ABoxes.

Cependant, il y a alors un problème de référence aux objets. Comment savoir quel élément une Kbase désigne ?

Nous proposons d'avoir recours à une référence basée sur un *accès* par le *contenu*. C'est-à-dire que les identifieurs utilisés pour désigner des éléments du domaine commun sont remplacés par leurs descriptions, c'est-à-dire des assertions au sujet de leurs propriétés. Traiter ensuite cette expression de LD contenant une telle description revient à traiter un problème d'hétérogénéité terminologique.

III.4 Conclusions du chapitre

Nous avons donné la formalisation à l'aide d'un langage simple de logique de descriptions pour les notions de : agent dialogique, interaction, pas d'interaction, production d'une requete, interprétation pour la production d'une réponse, traitement de la requête, interaction hétérogénène sémantiquement [94].

Dans le cas d'une interaction hétérogène, nous nous sommes intéressés aux cas d'échec de l'interaction, aux types d'hétérogénéités ; et avons proposé de gérer l'hétérogénéité à un niveau local en préservant les différences plutôt que de gommer l'homogénéité en la résolvant à un niveau global. Nous avons ensuite illustré cette formalisation avec un exemple abstrait qui nous a permis de mettre en evidence la réduction des mondes possibles lors d'une interaction.

Nous nous intéressons dans la partie suivante au raisonnement à partir de cas, et en particulier à l'étape d'adaptation. Nous proposerons un modèle pour l'adaptation à l'aide d'un outil de représentation des connaissances basé sur des notions simples de topologie algébrique. C'est dans la troisième partie de la thèse que nous appliquerons le modèle d'adaptation au problème de l'hétérogénéité sémantique entre agents dans InterViews.

Deuxième partie

Un modèle topologique pour l'adaptation dans l'analogie

Chapitre IV

Le raisonnement à base de cas et l'adaptation en Intelligence Artificielle

Introduction

Dans la partie 1, nous avons introduit le problème de l'hétérogénéité sémantique entre agents, dans le cadre du projet InterViews. Nous avons ensuite formalisé ce problème à l'aide de l'outil des logiques de descriptions. Au terme de cette formalisation, nous avons isolé un sous-problème qui est au *coeur* du traitement de l'hétérogénéité dans une requête : c'est *l'interprétation,* d'*un* terme hétérogène à la fois, par l'agent *receveur* de l'interaction, lors d'*un* seul pas d'interaction, et ce, *sans* avoir de connaissances explicites sur le contenu de la base de l'envoyeur.

Afin de résoudre ce sous-problème, nous proposons d'utiliser le paradigme du raisonnement à partir de cas, et en particulier d'utiliser la notion d'adaptation pour interpréter les termes inconnus. En effet, nous assimilons le processus cognitif d'interprétation d'un terme inconnu en fonction des termes de vocabulaire connus d'un agent pour inférer le sens d'une requête et y répondre, à un raisonnement à partir de cas. Les cas connus utilisés sont alors les termes de vocabulaires déjà connus, et le cas nouveau à "résoudre" est le terme hétérogène, dont on ne connaît pas le sens.

Plan du chapitre

Nous allons donc présenter le domaine du raisonnement à partir de cas, domaine qui touche à l'Intelligence Artificielle et aux Bases de Données. Nous verrons que l'étape qui nous intéresse plus spécifiquement est celle de *l'adaptation* que nous désirons isoler et modéliser. Pour cela nous restreignons notre étude au raisonnement par analogie simple pour lequel nous fournirons ensuite un modèle (chapitre VI) à l'aide d'outils spatiaux, que nous pensons naturels pour l'analogie (chapitre V).

IV.1 Le raisonnement à partir de cas

Nous avons proposé, à la fin du chapitre III, de baser la gestion d'une requête hétérogène par un agent modélisé comme une Kbase écrite en LD, sur des algorithmes d'adaptation. Cette idée nous est venue en considérant que l'interprétation d'un terme inconnu par une Kbase qui reçoit une requête hétérogène, ne peut utiliser que des connaissances à propos des termes déjà connus par la Kbase. En effet, nous avons rejeté la résolution de l'hétérogénéité sémantique par un point de vue global, en nous imposant de la gérer à niveau local, c'est-à-dire uniquement avec les connaissances de la Kbase réceptrice.

Interpréter un terme hétérogène est alors considéré comme un raisonnement à partir de cas, en prenant comme cas de base, les termes de la Kbase ou de la requête. Nous allons, dans cette section, présenter le raisonnement à partir de cas, étudier ses caractéristiques, les étapes cruciales d'un tel raisonnement, les domaines d'application de ce type de raisonnement, avant de revenir sur son utilisation pour le problème précis de gestion de terme hétérogène qui nous préoccupe.

IV.1.1 Principe et définitions

Le principe sous-jacent au Raisonnement à Partir de Cas (RàPC) [59] (Case-Based Reasoning CBR, en anglais) est que des *problèmes similaires ont des solutions similaires*. Ainsi, on modélise le fait d'agir d'après *l'expérience*.

Il est donc crucial de définir les notions de *problème* et *solution*. Elles sont en général définies par deux ensembles qui dépendent du domaine d'application, et représentées par des structures de données. Dans certains domaines, on possède une notion d'exactitude (*correctness*) de la solution, mais ce n'est pas toujours le cas. Si plusieurs solutions sont possibles, elles peuvent être *proposées*, dans un ordre de priorité.

Par ailleurs, il est aussi nécessaire d'établir une distinction entre une *expérience* et un *cas*. Une *expérience* décrit ce qui a été vrai ou faux, correct ou pas, bon ou mauvais, plus ou moins utile, etc. Elle peut être représentée par une règle, une contrainte, une loi, ou bien un *enregistrement* d'un événement passé. Un cas, quand à lui, est défini selon :

Définition | *Un cas, est un enregistrement d'épisodes où un problème (ou une classe de problèmes) a été résolu même partiellement. Il est représenté par une paire ordonnée :* $(probleme, solution)$.

Une base de cas est alors un *ensemble* de cas. Pour des besoins d'accès à cette base, elle est souvent équipée d'une structure supplémentaire. Enfin, si la solution du cas consiste en une *action*, on doit aussi stocker ses *effets*, le cas est alors défini par un triplet :

$$(probleme, solution, effets)$$

Le raisonnement à partir de cas permet d'apporter une solution $solution_{cible}$ à un problème nouveau, appelé la cible, noté $probleme_{cible}$. Ainsi, on peut définir le cas nouveau :

Définition | Un cas cible, noté $(probleme_{cible}, solution_{cible})$ est composé du nouveau problème, noté $probleme_{cible}$, qu'il s'agit de résoudre en utilisant des cas déjà résolus, afin de produire une solution : $solution_{cible}$.

Ce cas nouveau sera résolu par un raisonnement à base de cas, qui utilise les cas déjà résolus par le passé, et qui s'appellent alors des cas sources :

Définition | Un cas source, noté $(probleme_{source}, solution_{source})$ est composé d'un problème, noté $probleme_{source}$, déjà résolu par le passé et dont la solution $solution_{source}$ avait été validée.

IV.1.2 Domaines d'application en l'Intelligence Artificielle

Avant d'entrer dans la description scientifique du raisonnement à base de cas, nous plaçons ce domaine de recherches par rapport aux principaux domaines de l'Intelligence Artificielle. En particulier, nous nous intéressons à deux domaines d'application qui utilisent des systèmes intégrant le raisonnement à partir de cas. Nous verrons que ces domaines sont très proches du problème de l'interaction hétérogène entre agents dialogiques dans InterViews.

IV.1.2.1 Recherche d'information

La recherche d'informations notée RI [59] et le RàPC sont souvent utilisés pour les mêmes tâches, mais rarement intégrés dans un même système : Ces deux techniques essayent de retrouver de l'information *pertinente* (relevant), à l'aide d'un système *flexible* et facile de *requêtes*. De plus, ils font intervenir la notion de pertinence *approximative* par le biais de la *similarité* et délivrent une collection de matchs *possibles*.

Par contre, ils opèrent sur des *types* de données différents. En effet la RI se préoccupe de données essentiellement *textuelles*, et le RàPC s'effectue sur des *vecteurs* (numériques ou symboliques), même si actuellement, les chercheurs utilisant le RàPC s'attaquent aussi à des informations textuelles. De plus, les systèmes effectuant de la recherche d'information retrouvent en effet de l'information, mais ne sont pas concernés par leur *adaptation* vers des cas nouveaux.

Enfin, les systèmes de RI peuvent manipuler des *quantités* de données bien supérieures (jusqu'à un million de documents) à ce que peuvent manipuler les systèmes de RàPC (jusqu'à quelques centaines de cas). La RI opère sans utiliser de la *connaissance* sur le problème à résoudre, mais utilise seulement un système *générique d'indexation*. Enfin, des systèmes de RI sont *évalués* selon des tests *standard* et répandus (précision, recall) tandis que l'évaluation d'un système RàPC se fait au cas par cas, et leur comparaison est difficile.

Ces différences sont bien entendu dues à la différence d'objectifs de ces deux domaines. Mais il est néanmoins intéressant de souligner leur ressemblances. Ainsi, on peut considérer que le domaine de la RI utilise seulement les deux premières étapes du cycle du RàPC, que nous présentons ci-dessous (voir figure IV.1). Nous avons déjà mentionné l'importance de l'accès aux informations dans l'interaction entre agents dans InterViews, puisque l'envoi d'une requête est vu comme un accès à une information dans la base de connaissances interrogée.

A cause du développement d'applications telles que le Web, il existe des besoins croissants en systèmes *intelligents* de recherche de données. Pour satisfaire à ces besoins, deux voies sont principalement explorées par les chercheurs :

- Le RàPC est utilisé pour *guider* un système de RI : l'utilisateur interagit avec la partie RàPC qui formule la meilleure requête pour le système de RI, et récupère le résultat de ce dernier pour éventuellement l'adapter ;
- Une intégration plus fine se fait par le biais de la *description textuelle* des *attributs* représentant les cas : on a alors besoin des techniques de RI (comme les n-grammes) pour les retrouver à partir d'un RàPC.

De plus, les systèmes de RI et de RàPC se rapprochent puisque :

- Les systèmes de RI traitent maintenant des données qui ne sont plus uniquement textuelles (émergence de documents multi-médias comme les sons et les images, recherche noms propres, données structurées) et utilisent des *connaissances* disponibles.
- Le RàPC manipule de plus en plus des bases plus *vastes*, et évolue vers des méthodes *d'évaluation*.

IV.1.2.2 Bases de données

Le domaine des Bases de Données (noté BD) s'intéresse à la recherche d'informations sur la base d'un matching exact. Les systèmes de BD manipulent des bases *gigantesques* (plusieurs ordres de grandeur au-dessus du RàPC), et leurs méthodes d'accès sont entièrement *dynamiques*. En effet, on peut mélanger des insertions, modifications ou suppressions de données avec des requêtes sans perte de performance.

Les DBMS (Data Base Managment Systems) peuvent être utilisées dans un système de RàPC de deux manières :

- En tant que *stockage persistant* de cas : le système de RàPC ne contient que des pointeurs vers les cas qui sont stockés et indexés dans la DBMS ;
- Pour réaliser un système de RàPC par des moyens propres aux DBMS : la recherche est effectuée en créant une *table* qui encode les mesures de similarité et les requêtes (de type SQL) appropriées. La difficulté est alors de générer les bonnes requêtes. Si l'on utilise des DBMS orientées objet (OODBMS), elles ne disposent pas de langage de requêtes, mais paradoxalement cela rend la tâche encore plus facile. La recherche est alors effectuée *sans* l'intermédiaire du langage de requêtes.

On utilise de plus en plus des données *incomplètes* ou incertaines en BD ce qui nécessite l'incorporation de notions de *similarité* par exemple, ou l'utilisation *d'algorithmes* de recherche communs dans le domaine du RàPC.

Les Systèmes d'Informations Géographiques (SIG) font intervenir des données *multi-dimensionnelles*, tout comme les documents multi-médias. Le problème qui se pose alors est qu'il n'existe plus *d'ordre partiel*, par exemple sur les objets spatiaux qui préserve leur proximité spatiale.

Enfin, à cause du développement des moyens tels que l'internet et de la propagation des intranets, les commerciaux développent de plus en plus des serveurs de BD *extensibles*, ORDBMS (*Object-Relational Database Management Systems*). Le but est alors de pouvoir incorporer justement des données multi-dimensionnelles.

IV.1.3 Description scientifique du RàPC

Nous avons défini dans les grandes lignes ce qu'était un raisonnement à partir de cas, et nous avons vu que les domaines d'application qui en font usage en Intelligence Artificielle sont assez proches des problématiques rencontrées dans InterViews. Afin d'étudier plus en détail la façon dont nous pourrions effectivement utiliser les notions et paradigmes du RàPC pour résoudre notre problème de gestion de requêtes hétérogènes par une Kbase écrite en LD, nous nous intéressons maintenant à la description scientifique du raisonnement à partir de cas.

IV.1.3.1 Le cycle du RàPC

Les phases du raisonnement à base de cas, qui ont été décrites par Aamodt et Plaza en 1994 sont illustrées par la figure IV.1. Il y a quatre étapes qui précèdent la production d'une solution pour la cas cible :

1. *Formulation* du problème en un nouveau cas cible (élaborer) : cette étape inclut le choix du formalisme de représentation des connaissances en fonction du domaine et des besoins de l'application.
2. *Sélection* d'un cas source : cette étape inclut une comparaison, avec une mesure de similarité entre le cas cible et tous les cas sélectionnables comme cas sources.
3. *Appariement* du cas cible avec la source sélectionnée, en utilisant des contraintes de structure. Cette étape peut être imbriquée avec la précédente dans l'action de retrouver.
4. *Adaptation* de la solution du cas source pour l'appliquer au cas cible.

Suivies de trois étapes qui s'effectuent après la production de la solution pour le cas cible :

1. *Révision* de la solution adaptée pour une validation ;
2. Mise en évidence des traits que ce cas apprend au système (*retain*) ;
3. *Stockage* (étape de learning), ou écarter le cas (discard), avec apprentissage.

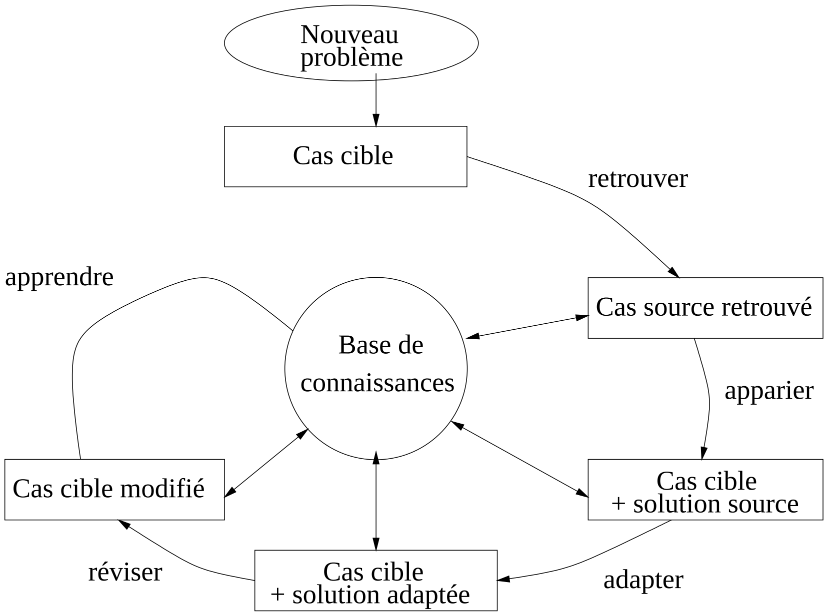
Les étapes 4, 5 et 6 sont postérieures à *l'adaptation* et donc à la production à proprement parler d'une solution pour le problème courant. Les étapes 1, 2 et 3, préparent cette adaptation. On retrouvera ces étapes préliminaires ainsi que l'adaptation elle-même, dans le *raisonnement* par *analogies* (qui inclut donc 4 étapes) qui peut donc être considéré comme un sous-processus inclus dans le cycle du RàPC [35], et que nous étudions dans la section IV.2.

IV.1.3.2 Importance de la représentation pour la formulation

Les cas qui seront stockés et sélectionnés doivent être représentés dans un certain formalisme. Le choix de ce formalisme est crucial pour le raisonnement à partir de cas, puisqu'il va falloir comparer des cas passés à la situation courante et cette comparaison se fera selon des procédures fortement dépendantes du formalisme choisi. En particulier, les traits mis en valeur par une telle représentation ne seront pas forcément les plus pertinents.

Rappelons qu'en représentation des connaissances, la façon dont est *exprimée* une connaissance consiste en une *structure* de données et des *opérations* d'inférence possibles dessus. Les structures de données les plus courantes utilisées en RàPC sont :

FIG. IV.1 – Le Cycle du RàPC, dont les étapes sont décrites par Aamodt et Plaza en 1994 [35].



– Attributs-valeurs ;
– Arbres de décisions
– Prédicats de contraintes ;
– Règles ;

Examinons-les rapidement.

IV.1.3.2.a Attributs-valeurs : Un attribut est la donnée d'un *nom* A, un *domaine* de valeurs $dom(A)$, et une *variable* x_A. Le domaine $dom(A)$ est *numérique* ou *symbolique*, et peut contenir une information supplémentaire comme un *ordre* (partiel ou total) par exemple. Les attributs sont souvent appelés des *features* et un vecteur d'attributs-valeurs est un *n-uplet*. Dans certaines applications les attributs sont *fixés*. Dans d'autres on travaille avec l'hypothèse de *monde ouvert* [49]. Si on ne donne que des listes d'attributs-valeurs, on a alors une *flat representation*, si on *structure* (hiérarchie objet par exemple), on a une *structural representation*.

IV.1.3.2.b Arbres de décision : Ils sont utilisés surtout pour les tâches de *classification*. Les noeuds sont des attributs, les arcs qui en partent, portent une de leurs valeurs, et les feuilles sont des *classes*. Ainsi, un *chemin* dans cet arbre représente le fait qu'un attribut avec une valeur particulière appartient à une classe donnée.

IV.1.3.2.c Prédicats de contraintes : Les *contraintes* sont des prédicats n-aires qui représentent des relations requises entre par exemple des attributs. Les contraintes dites hard, *doivent* être satisfaites, tandis que les contraintes dites soft, sont seulement des *voeux*, elles peuvent être ordonnées avec des priorités.

IV.1.3.2.d Règles : Une *règle de logique* est de la forme : $A_{1,}A_{2,...,}A_n \rightarrow B$. Quand on utilise des *variables*, elles sont traitées comme étant universellement *quantifiées.* Une règle contenant des variables est un exemple de connaissance abstraite (*Abstract Knowledge*), car elle représente potentiellement des connaissances différentes (instanciation des variables). L'application en avant (*forward application*) d'une règle autorise l'ajout de la connaissance B à la base (B peut être aussi une action à effectuer sur cette base), son application en arrière (*backwards chaining*) réduit la satisfaction de B à ses prémisses. Une grande différence avec l'approche RàPC, est que seulement les matching *exacts* des hypothèses peuvent mener à l'application d'une règle. En RàPC, on trouve des règles qui sont utilisées pour les inférences, mais aussi pour *transformer* la solution d'un cas sélectionné.

IV.1.3.2.e Représentation pour les domaines d'application du RàPC Contrairement aux formalismes que nous venons de voir, le formalisme de représentation des connaissances le plus répandu pour les domaines d'application des systèmes de RàPC (c'est-à-dire en RI, CBR, DBMS) est la *logique des descriptions.* En effet, elle fourni une représentation puissante, basée *objet,* assortie de règles *d'inférence.*

Ceci constitue une similitude supplémentaire avec le formalisme de représentation choisi pour notre problème d'hétérogénéité sémantique dans InterViews.

IV.1.3.3 Similarité et sélection des cas

Lorsque l'on désire utiliser des cas qui correspondent à des matching *non exacts*, on introduit la notion de *similarité.* Cette notion est définie comme la duale d'une *distance* (concept mathématique). Une similarité est une notion relative au domaine traité.

Une similarité peut être formalisée comme une *relation* à quatre membres $R(x, y, u, v)$, qui veut dire "x et y sont au moins similaires comme u et v. On définit alors les voisins les plus proches comme $VPP(x, z) \Longleftrightarrow \forall y, R(x, z, x, y)$. On peut considérer que le plus proche voisin est unique (VPP est alors une fonction), ou bien qu'ils sont multiples.

La similarité est traitée comme un *ordre partiel*, qui peut être réalisé par une *fonction* numérique $sim(a, b)$ appelée *mesure* de similarité, ou sa duale, la *distance.* Ces fonctions induisent une relation à 4 membres. En général on réduit les valeurs numériques au segment des réels $[0, 1]$ (*normalisation*), et la similarité à pour valeur 1 quand on compare un élément à lui-même. On peut parfois également ajouter une *symétrie* de la relation et *l'inégalité triangulaire.* La notion de distance, se base en général la distance de *Hamming*, ou bien sur des distances *euclidiennes* ou des mesures de *manhattan* pour des vecteurs à n dimensions. Voici quelques points de vocabulaire :

- **Mesure de similarité** *:* qui peut stocker des connaissances plus ou moins sophistiquées, comme la similarité de *surface* qui considère seulement des propriétés *syntaxiques*, ou la similarité *locale* qui traite seulement *d'un* attribut.
- **Similarité globale** *:* est celle qui est effectivement utilisée, elle représente une vision *holiste* des cas. Elle peut être dérivée à partir de similarités locales, en utilisant des *poids*, ou bien la position de chacun des attributs dans une *hiérarchie*, ou même en utilisant de la connaissances dite de *background.*

– **Similarité structurelle** : ce terme ambigu, peut désigner une similarité basée sur la *structure* des cas.

La sélection d'un cas est une opération commune avec les bases de données (BD). Une donnée est retrouvée dans une BD à l'aide d'une requête, et on doit opérer à un matching *exact*. Ce matching fait alors intervenir les problèmes *d'indexation*. Une requête vers une base RàPC a recours à des matchings *inexacts*.

Dans les deux types de bases, les *arbres* jouent un rôle primordial pour organiser les cas de manière efficace pour les retrouver. Une structure utilisée fréquemment est le *kd-tree*. Ces arbres n'excluent pas certaines branches qui auraient été exclues par un matching exact. Certaines zones peuvent être considérées à cause de la similarité, cela utilise les techniques de *ball-within-bounds* ou *ball-overlaps-bound test*, qui sont des raffinements des *techniques* sur les arbres. D'autres structures pour la sélection sont les *case retrieval nets* et les *discrimination nets*.

IV.1.3.4 Adaptation d'un cas

L'adaptation revient à *ajuster* un cas sélectionné pour le faire coïncider au problème cible à traiter. La plus simple *adaptation* est de prendre la solution du cas telle quelle.

C'est dans cette tâche que l'on utilise les principes du *raisonnement* par *analogie* (*analogical reasoning*). L'existence d'une analogie entre le problème source et le problème cible est alors utilisée pour le *transfert* du "principe" de la solution source, vers le problème cible, afin de produire une solution cible. Ce transfert s'opèrant entre deux problèmes différents, pouvant toucher à des domaines différents, il y a alors transfert entre deux domaines différents. Cependant, en RàPC, on ne l'utilise que pour transférer des solutions à l'intérieur d'un *même* domaine. C'est pourquoi nous étudions, dans la section suivante, les modèles et systèmes pour l'analogie, et que nous ne proposerons un modèle que pour un problème simple d'analogies intra-domaine (dans le chapitre VI).

IV.2 Adaptation et analogie

Pour étudier le raisonnement par analogie, nous nous intéressons tout d'abord brièvement aux contributions provenant de la philosophie et la psychologie. Nous étudions ensuite les principaux modèles proposées en Intelligence Artificielle afin de modéliser le raisonnement par analogies. Pour cela, nous présentons les systèmes qui ont été programmés en illustration de ces modèles.

IV.2.1 Raisonner par analogies

IV.2.1.1 Les contributions de la philosophie

Les contributions provenant de la philosophie, envisagent surtout l'analogie comme un *outil* de raisonnement puissant en *science* et l'étudient donc en rapport avec l'histoire des sciences

[62]. En effet, les philosophes ont essayé de fournir un cadre *conceptuel* pour proposer une *définition* de l'analogie.

Ainsi, Aristote [2] définit informellement le raisonnement par analogies comme étant un processus divisé en deux étapes : une *induction* produisant une *règle*, puis l'application de cette règle à la *cible*. Par la suite, les principales propositions d'autres philosophes datent du siècle passé, ou de ce siècle [62]. Par exemple, Mill, en 1873 [63], propose une mesure de similarité comme justification à l'application d'une règle d'analogie. En 1967, Ujomow [84] caractérise le raisonnement par analogie comme un transfert d'information d'une source vers une cible. Il imagine pour cela des conditions de plausibilité pour les inférences d'analogie, comme la pertinence d'un tel transfert. Il propose également de classer les raisonnements par analogies en :

- Structurel-fonctionnel : on infère des fonctions similaires pour des objets présentant des structures similaires ;
- Fonctionnel-structurel : on infère des structures similaires pour des objets remplissant des fonctions similaires ;
- Cause-effet : on infère des effets similaires pour des faits ayant des causes similaires.

En 1984, Weitzenfeld [102] considère le raisonnement par analogie comme un processus extra-logique, dans le sens où il utilise des prémisses qui ne peuvent pas être déduites d'une théorie. Ces prémisses sont appelées des structures déterminantes, elles sont des similarités entre la source et la cible qui vont *déterminer* les similarités *inférées*.

En 1989, Indurkhya [55], s'intéresse aux métaphores et au raisonnement par analogie, et introduit un ensemble de contraintes sémantiques pour qu'une analogie ait lieu. De plus, il définit les analogies comme des homomorphismes entre deux algèbres. Nous verrons que cette proposition nous intéressera beaucoup dans le contexte d'un modèle topologique pour l'analogie (voir chapitre VI). Enfin, citons Hesse, qui en 1966 [48], pose la différence entre les propriétés communes et les propriétés non communes qui mènent à la définition d'une analogie négative.

IV.2.1.2 Les contributions de la psychologie

Plus récemment, des recherches ont été menées en psychologie cognitive. En effet, l'analogie est considérée comme un processus de base intervenant dans l'apprentissage, la résolution de problèmes, et surtout dans le *développement* cognitif. Dans ce domaine, les modèles proposés sont toujours basés sur des résultats *expérimentaux*.

Les recherches empiriques ont établi l'existence de sous-processus dans le raisonnement par analogie et ont montré la différence entre l'utilisation d'analogies par des experts et par des novices [62]. Plusieurs théories [45] établissent une décomposition du raisonnement par analogies en étapes :

- **Représentation** des sources et de la cible.
- **Recherche** des **sources** candidates : la résolution de problèmes par analogie commence réellement lors de la recherche de cas sources qui pourraient servir de base au nouveau problème [41]. Des études montrent que les sujets accèdent aux sources par un index qui représente des traits de surface. Mais les composants de structure peuvent également intervenir. Dans tous les cas, les composantes de structures ont ensuite une plus grande importance que les traits de surface dans l'utilisation de la source.

- **Appariement** entre la source sélectionnée et la cible : les théories pour l'étape d'appariement vont du simple remplacement des rôles à des appariements très élaborés [62]. Gentner [37] montre que les analogies appariant des relations d'ordre supérieur sont préférées aux autres. D'autres contraintes sont également prouvées, comme les appariements se restreignant aux informations de même type [31]. Enfin, l'information appariée dans une analogie dépend également du but [53].
- Evaluation et **adaptation** de la solution vers la cible ;
- Apprentissage avec **généralisation** de ce transfert analogique.

Ces théories ont souvent donné naissance à des systèmes informatiques. Les principales contributions sont au nombre de trois :

- La SMT (structure mapping theory) proposée par D. Gentner, qui a donné le système SME (structure mapping engine) [38] ;
- ACME qui est le système incarnant la théorie de Holyoak et Thagard [52] ;
- IAM qui incarne le modèle de Keane [62].

Ces systèmes ont servi à tester les modèles en les comparant aux performances humaines dans des tests de *validation*.

IV.2.1.3 Définition

Au terme de cette présentation, et pour unifier la terminologie utilisée lors de la présentation des modèles qui vont suivre, nous proposons une définition des termes employés pour parler d'analogie :

Une analogie est une relation à quatre termes A, B, C, D notée (on parle souvent du carré d'analogie) :

$$A : B :: C : D$$

dans laquelle le couple $A : B$ vérifie une relation R_{AB} "analogue" à la relation R_{CD} vérifiée par le couple $C : D$. Le fait que ces relations soient analogues, est noté par l'opérateur ::. On dit que $A : B$ est le cas *source*, alors que $C : D$ est le cas *cible*.

Enfin, avant de passer aux modèles plus complets proposés pour l'analogie, voici schématiquement les étapes traditionnellement établies pour la résolution d'une analogie :

1. Recherche des relations liant A à B (R_{AB}) ;
2. Recherche du domaine de C à modifier en fonction de ce qu'il partage avec A ;
3. Application de la relation R_{AB} à C ;
4. Vérification éventuelle avec $R_{AC}(B)$.

L'étape 3 correspond alors à la fameuse étape *d'adaptation* puisqu'il s'agit d'adapter la relation R_{AB} au domaine C qui a priori n'a aucune raison d'appartenir au domaine de départ de cette relation uniquement établie sur la base des éléments A et B. De plus on dit qu'on adapte la structure C en D en fonction de R_{AB}.

Nous allons maintenant voir qu'une résolution d'analogie est un processus bien plus compliqué faisant intervenir d'autres fonctions cognitives. Nous ne présentons ici que les modèles qui ont donné naissance à des systèmes informatiques étant donné que nous sommes à la recherche de modèles implémentables. De plus, nous ne présenterons que les principaux modèles ayant ensuite entraîné la proposition de systèmes plus sophistiqués mais se réclamant toujours de ces mêmes courants.

IV.2.2 Le système ANALOGY proposé par Evans

Le domaine de la résolution de problèmes de type tests de Q.I. a joué le rôle de problème paradigmatique [46] pour l'analogie aux débuts de l'Intelligence Artificielle. Proposé par Evans en 1968 [30], le système ANALOGY constitue un précurseur dans le domaine. Il est destiné à résoudre des tests d'intelligence fondés sur des analogies géométriques et graphiques. Il fonctionne à partir de la connaissance de :

- Relations topologiques ;
- Relations et transformations géométriques ;
- Modifications (suppression ou addition de tracé).

La tâche consiste à sélectionner la bonne figure parmi un choix de 5 figures de sorte qu'elle complète correctement l'analogie. La recherche des solutions s'effectue suivant la démarche suivante :

- Génération de règles de transformation possibles entre A et B ;
- Génération de règles de transformation permettant de passer de C à chacune des figures proposées ;
- Comparaison des deux jeux de règles et sélection de la meilleure solution.

Plusieurs règles de transformation peuvent être associées à un couple de figures. De plus, la dernière étape de comparaison implique une certaine mesure de similarité. Enfin, remarquons qu'il se peut que le système ne trouve pas de solution "adéquate".

Les auteurs insistent sur le fait qu'il y a un choix à effectuer parmi un certain nombre de solutions proposés. En fait, il serait impossible de laisser le sujet construire complètement une solution dans ce genre de tests car il serait bien difficile de juger de l'exactitude de la solution.

Nous avons un avis différent sur ce point puisque nous pensons qu'il est plus intéressant de construire une ou plusieurs solutions à condition que nous puissions les justifier par rapport à une transformation.

IV.2.3 SME : système basé sur le modèle proposé par Gentner

IV.2.3.1 Présentation du modèle théorique SMT

Le modèle le plus répandu et utilisé en IA est certainement celui proposé par Gentner, pour lequel l'analogie est fondamentalement un appariement entre deux domaines, qui préserve les structures [96].

Selon Gentner, une théorie de l'analogie doit décrire comment le sens d'une analogie est dérivé à partir du sens de ses composants [38]. La théorie qu'elle propose s'appelle la "Structure-Mapping Theory", notée dans la suite SMT. Cette théorie est basée sur un certain nombre de règles qui se résument en deux points forts :

1. les règles dépendent uniquement de propriétés *syntaxiques* de la *représentation* (et pas du contenu),
2. les analogies sont clairement distinguées des autres phénomènes tels que les constats de similarité, les applications d'abstractions et tous les autres types de comparaisons.

Ainsi, dans SMT, l'analogie est vue comme le résultat d'un appariement structurel entre domaines [96]. Cette modélisation repose sur l'hypothèse implicite que nous nous représentons les concepts et situations avec une structure[70]. Cette approche est à rapprocher de la théorie Chomskienne pour la linguistique qui se focalise également sur un aspect purement syntaxique de la problématique, ce qui a le mérite d'en faciliter le formalisation. Deux principes sont introduits pour la phase *d'appariement structurel* :

IV.2.3.1.a Préférence pour les relations : Les *relations* entre les objets sont utilisées plutôt que les attributs des objets pour être appariées entre elles. En effet, Gentner part de la constatation que la force ou l'efficacité d'une analogie ne dépend pas forcément de chaque attribut de la même manière : c'est-à-dire que tous les traits d'une situation ne sont pas forcément pertinents de la même façon. Par ailleurs, dans une analogie du style 3 : 6 :: 2 : 4, ce ne sont pas les attributs que le chiffre 3 partage avec le chiffre 2 qui vont être utiles pour trouver le chiffre 4, mais ses relations avec le chiffre 6 avec lequel il est mis en correspondance [38]. Ainsi, une théorie basée sur les relations entre composants semble plus adaptée à la modélisation d'une analogie de ce type. Les situations sur lesquelles vont être effectuées des analogies sont vues comme des systèmes d'objets, d'attributs d'objets et de relations entre objets[1].

IV.2.3.1.b Principe de systématicité : Les relations qui sont appariées sont sélectionnées par le *principe* de *systématicité* [26]. Ce principe stipule que lorsqu'une structure est sélectionnée dans un domaine source, le sous-ensemble de cette structure qui est effectivement apparié avec la cible est choisi en fonction de la cohérence de l'appariement qui en résulte. En effet, les appariements doivent produire une correspondance un à un entre des entités de la source et de la cible, et l'arité des prédicats appariés doit être la même également. Enfin, les relations ayant une arité supérieure sont préférées car elles contribuent plus à la connectivité structurelle du tout. L'intuition qui est illustrée dans ce principe [96] est que les attributs (prédicats d'arité un) dépendent énormément du domaine et ne peuvent donc pas être appariés à l'identique. Tandis que les relations connectant de manière forte plusieurs sous-structures, sont plus centrales dans la sémantique de la situation.

IV.2.3.2 Le système SME correspondant à SMT

Ce système [31] constitue l'implémentation pour l'Intelligence Artificielle de la théorie de Gentner appelée SMT. Nous avons vu que dans SMT, l'analogie est vue comme le produit d'un appariement structurel entre domaines [96]. Une conséquence intéressante de ceci est que l'algorithme qui en découle est *général* : en effet, à partir une représentation structurée de n'importe quel domaine, il est possible de rechercher et effectuer des appariements. Ainsi, il

[1] Ce qui reste compatible avec la vision propre aux logiques de descriptions.

n'y a pas besoin de disposer d'algorithmes différents selon les domaines et les analogies considérées. Cependant, une conséquence moins intéressante est aussi qu'il faut disposer de cette représentation structurée sur les domaines pour appliquer les appariements en question. Cette représentation structurée des cas doit se faire à l'aide d'un jeu de primitives strict. Les représentations sont alors canoniques dans lesquelles des sémantiques similaires sont représentées à l'aide de structures isomorphes.

Soit un cas source noté S et représenté par les noeuds $s_1, ..., s_n$, et un cas cible, noté C et représenté par les noeuds $c_1, ..., c_n$. L'appariement nécessaire pour effectuer une analogie concerne les noeuds d'objets de la source et la cible selon :

$$App : s_i \to c_i$$

Cet appariement génère des candidats et implique un ensemble d'inférences à opérer sur la cible pour importer les prédicats sur les noeuds de la source S vers la cible C. Les règles d'appariement sont :

- Abandonner les attributs ;
- Préserver les relations entre les objets ;
- Décider quelles relations sont préservées (systématicité).

Ainsi, ces règles sont purement structurelles. Les relations privilégiées sont les relations d'ordre supérieur pour les appariements. De plus, ces appariements sont des correspondances un à un d'objets.

IV.2.4 Le modèle de Holyoak et Thagard : ACME

ACME est le système correspondant au modèle proposé par Holyoak et Thagard [52]. Ce système utilise un ensemble de contraintes *sémantiques*, *structurelles* et *pragmatiques*, par opposition à celui de Gentner qui se limitait aux contraintes de nature structurelle. Un raisonnement par analogie dans ACME est décomposé en quatre étapes :

1. **Trouver** une **source** plausible pour l'analogie ;
2. La mettre en **correspondance** avec la cible ;
3. **Transférer** l'information de la source vers la cible ;
4. **Apprendre** l'analogie.

Le modèle de ACME, comme celui que nous serons amenés à proposer dans le chapitre VI, n'a pas la prétention ni l'ambition d'être plus performant qu'un humain mais seulement de modéliser un processus cognitif *plausible*. De plus, ils s'intéressent à ces problèmes par le biais de *l'apprentissage* et de la recherche d'information en *mémoire* en général.

Leur idée directrice est d'appliquer des contraintes en *parallèle*, ce qui nous séduit beaucoup. En effet, ils cherchent à modéliser le processus *d'insight*. C'est un point qui nous semble tout à fait intéressant à étudier pour une approche cognitive de l'analogie. La technique d'implémentation choisie par les auteurs est celle des réseaux de *neurones* ce qui permet bien un traitement distribué et en parallèle. Cette approche connexioniste permet de coder les contraintes sémantiques, structurelles et pragmatiques comme des préférences ou des pressions qui guident le système vers des appariements plus cohérents.

Comme pour SME, il y a également une contrainte sur l'isomorphisme de structures entre les domaines appariés. Cependant, tandis que le principe de systématicité de Gentner aboutit au rejet de plusieurs analogies qui pourraient avoir lieu [96], dans ACME, la contrainte pragmatique permet des appariements non parfaits, c'est-à-dire où les isomorphismes ne sont pas exacts.

De plus, le réseau connexionniste sur lequel s'appuie l'appariement, est construit dynamiquement (inhibitions) en fonction des analogies. Ainsi, ce réseau ne reflète pas de la connaissance à propos du monde, mais plutôt une méta-connaissance construite spécialement pour l'analogie étudiée.

Enfin, ce modèle utilise des stratégies de compétition entre les solutions ce qui ne nous concerne pas pour le moment car nous ne sommes pas intéressés par une classification des solutions. Mais nous verrons que le modèle que nous proposerons se base sur des outils de représentation symbolique qui exploitent une représentation spatiale des connaissance, et non sur une représentation distribuée par réseaux de neurones.

IV.2.5 Chalmers, French, Mitchell, Hofstadter : Le système Copycat

Douglas Hofstadter et ses étudiants proposent une modélisation des processus d'analogie comme le résultat d'un processus cognitif plus général appelé la perception de haut-niveau (HLP pour High-Level Perception en anglais) [23]. Dans cette théorie la représentation d'un concept de haut niveau est faite par interaction avec des processus de bas-niveau, à savoir la perception.

Le système Copycat, proposé par Mitchell [67] [66], incarne les propositions de Hofstadter et Chalmers. Selon eux, lorsqu'une personne fait une analogie, elle se sert de sa perception de certains aspects structurels de la situation. Établir une analogie revient alors à mettre en évidence certains aspects, pas toujours les plus triviaux, de la situation : cela correspond à une grande capacité de flexibilité des processus de perception. Ainsi, une représentation rigide de la situation ne peut refléter ce processus, et ce sont forcément certaines facettes d'une représentation qui sont *sélectionnée* pour un contexte donné. D'ailleurs, il en découle que le raisonnement par analogies est un mécanisme puissant d'enrichissement d'une représentation.

Dans Copycat, on admet que les deux processus : perception de la situation et appariement structurel, sont importants pour effectuer une analogie, mais on met l'accent sur le premier (alors que Gentner par exemple met l'accent sur l'appariement structurel).

Le système Copycat, travaille sur des chaînes alphabétiques. Ainsi, les problème de perception d'un bas niveau (reconnaissances de formes, segmentation) sont évités, alors que le problèmes est assez complexe pour illustrer des analogies assez riches.

L'architecture proposée dans Copycat est encore une fois parallèle et non déterministe. Elle fonctionne par interactions constantes entre des agents de perception et un réseau associatif de concepts (à la manière d'une terminologie). Ils se situent donc à mi-chemin entre les approches connexionistes et les approches symboliques.

IV.3 Conclusions du chapitre

Nous avons fait un rapide état de l'art concernant le raisonnement à partir de cas puis le raisonnement par analogies. En effet, il apparaît que la raisonnement à partir de cas, qui est largement utilisé pour des applications de bases de données et de bases de connaissances, est composé de plusieurs étapes. Parmi ces étapes, celle qui semble le moins développée, et qui ne trouve pas de correspondance dans le domaine des bases de connaissances en logique de descriptions, est celle de l'adaptation. C'est précisément cette étape qui nous intéresse et pour laquelle nous souhaiterions proposer un modèle simple.

Nous nous sommes donc intéressés dans un deuxième temps au raisonnement par analogie, qui utilise également cette étape d'adaptation (appelée *transfert*) mais qui présente l'avantage de pouvoir isoler cette étape dans un sous-problème plus simple. Nous avons également brossé un rapide état de l'art dans le domaine de l'analogie.

Il en ressort que les modèles les plus simples modélisent l'analogie comme un processus incluant toujours un appariement structurel. L'importance accordée à cette étape est plus ou moins grande, puisque certains ajoutent des contraintes de sélection pragmatique des solutions (après l'appariement) ou d'autres accordent une grande importance aux processus de perception (avant l'appariement). En ce qui nous concerne, pour résoudre le problème des termes hétérogènes dans une requête, étant donné l'aspect déjà très syntaxique de la formulation de notre problème, il nous parait judicieux et suffisant de nous restreindre purement au phénomène d'appariement de structures. Ainsi, sans aller plus avant dans l'étude détaillée de l'analogie, nous voudrions utiliser cette notion d'appariement structurel afin de proposer des solutions algorithmiques pour notre problème d'hétérogénéité sémantique.

A ce stade de notre étude, nous proposons d'utiliser des outils spatiaux pour fournir un bon modèle d'appariement structurel et d'adaptation dans l'analogie. Ainsi, nous allons proposer un modèle basé sur des notions simples de topologie algébrique, que nous exposons maintenant, dans le chapitre V.

Chapitre V

Outils de topologie algébrique pour la représentation des connaissances

Motivations

Les *concepts spatiaux* et les *structures spatiales* peuvent être utiles pour représenter des connaissances qui ne sont pas nécessairement spatiales. En effet, nous adhérons à l'hypothèse selon laquelle il existe une *analogie* de structure entre l'espace concret et d'autres domaines de connaissance, puisque notre interaction avec l'espace concret au travers de notre corps est la base de notre expérience et donc de la construction de nos bases de connaissances [44]. Par exemple, Habel [44] met l'accent sur la structure d'*ordre* commune à l'espace et au temps et qui se traduit entre autres par l'utilisation du même mot “entre” [1] pour parler d'un objet “entre” deux autres *spatialement*, mais également d'un instant compris “entre” deux autres *temporellement*.

En Intelligence Artificielle, les concepts spatiaux servent déjà de base pour construire des *formalismes* de *représentaion des connaissances*. Nous pouvons ainsi raisonner dans plusieurs domaines en modelant des structures spatiales qui restent à définir. Nous nous pencherons ici sur l'étude particulière d'un outil pour la représentation spatiale de connaissances, basé sur la *topologie algébrique*.

En effet, les relations spatiales exploitables pour représenter des connaissances, se divisent généralement en trois grandes catégories :

1. Les concepts topologiques : régions, bords, transformations ;
2. Les concepts métriques : distances, tailles ;
3. Les concepts d'ordre : comparaisons, positions relatives.

Nous nous intéressons ici à une utilisation des concepts topologiques uniquement et nous aborderons ensuite les aspects liés au concept de relation d'ordre. Nous ne sommes pas intéressés

[1] en anglais “between”.

par les concepts métriques puisque nous nous pencherons sur des tâches cognitives symboliques qui ne requièrent pas l'intervention d'une métrique.

L'évaluation d'un système de représentation des connaissances se fait selon 4 critères essentiellement [60] : Abstraction : la représentation ne doit contienir que des informations pertinentes pour que le pouvoir d'abstraction est satisfaisant.

Expressivité toute la connaissance nécessaire doit être exprimable dans la représentation.

Naturalité l'expression de la connaissance nécessaire doit se faire de manière naturelle[2], les propriétés nécessaires doivent être accesibles (visibles, calculables) facilement.

Efficacité l'exécution du programme résultant se fait de manière efficace (surtout en termes de temps pour une plausibilité cognitive)

Certaines de ces évaluations doivent se faire sur le système de représentation des connaissances lorsqu'il sera déjà élaboré, tandis que d'autres permettent de guider notre choix du formalisme de représentation. En particulier, nous attachons une grande importance aux critères de naturalité et d'expressivité, qui sont liés. Pour cette raison, et puisque nous nous intéressons aux concepts topologiques nous allons étudier la possibilité d'élaborer un formalisme pour la représentation spatiale (et topologique) des connaissances basé lui-même sur la topologie. Pour satisfaire au critère d'efficacité nous devons utiliser une branche constructiviste de la topologie, du point de vue d'un informaticien, nous choisissons la topologie algébrique. De plus, des formalismes ayant déjà prouvé leur fort pouvoir d'abstraction, tels que les graphes conceptuels sont basés sur les graphes. Or, les graphes, sont un cas particulier de complexes simpliciaux, notion élémentaire de la topologie algébrique.

Résumé du chapitre

Dans ce chapitre, nous introduisons les concepts élémentaires de topologie algébrique utilisés par la suite : les complexes simpliciaux. Nous présentons la Q-Analyse qui permet de représenter les relations binaires à l'aide de complexes. Nous proposons ensuite un moyen d'utiliser cette Q-analyse pour représenter les prédicats et les concepts. Nous donnons des exemples pour argumenter notre proposition et la comparer à d'autres systèmes.

V.1 Éléments de topologie algébrique

V.1.1 Objet d'étude

La topologie a pour objectif d'étudier l'*espace* pris comme un ensemble de points muni d'une relation de voisinage. Dans la notion d'espace, on s'intéresse plus à cette structure de voisinage qu'aux points eux-mêmes. Ainsi, on essaie de formaliser les notions de *continuité*, de *limite*, de *bord*, de *chemin*, etc. Dans ce document, nous noterons ce formalisme de représenation des connaissances CAT pour Combinatorial Algebraic Topology [40].

[2]Par exemple par une correspondance naturelle, dite *analogique* entre le matériel représenté et la représentation, voir Johnson-Laird et les modèles mentaux [56].

V.1.2 Simplexes et complexes

Nous présentons maintenant la structure de base que nous utilisons dans le formalisme CAT : le complexe simplicial. Voici une définition préliminaire :

Définition : On appelle k-cellule, un morceau d'espace « abstrait » de dimension k et qui ne contient aucun "trou".

Ces cellules sont des briques de base pour la construction d'un espace C moyennant les conditions :

- Le bord d'une cellule est constitué d'une union finie de cellules de C ;
- L'intersection de deux cellules de C est soit vide, soit une autre cellule de C.

On dit alors que C est un *complexe cellulaire*. La structure abstraite correspondante est celle de *complexe simplicial* :

Définition Un complexe simplicial abstrait est un couple (V, C) où V est un ensemble de sommets et C un ensemble de parties finies de V telles que si $s \in C$ alors toutes les parties $s' \in s$ appartiennent aussi à C.

Les éléments de C sont des simplexes abstraits, la dimension d'un simplexe est son nombre de sommets moins un, la dimension du complexe C est celle du plus grand simplexe qu'il contient.

Un simplexe peut être vu comme un polyèdre prenant appui sur des sommets. La figure V.1 montre un simplexe :

- De dimension 0 : le simplexe est réduit à un point, il se note $\langle a_0 \rangle$ par exemple si le sommet est nommé a_0 ;
- De dimensions 1 : le simplexe est un arc appuyé sur deux sommets de dimension 0, il se note par exemple $\langle a_1, a_2 \rangle$ où a_1 et a_2 sont les noms donnés aux deux sommets ;
- Et de dimension 2 : le simplexe est une surface triangulaire appuyé sur trois arcs de dimension 1, eux-mêmes appuyés sur 3 points correspondant aux simplexes de dimension 0, il se note par exemple $\langle a_1, a_2, a_3 \rangle$.

V.1.3 Complexes et graphes

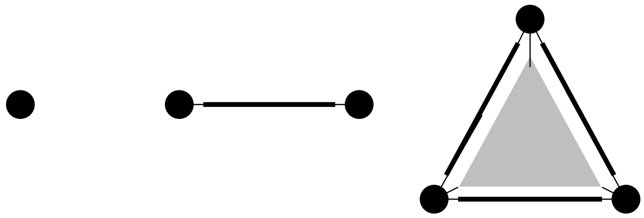
Nous notons que la notion de complexe abstrait généralise la notion de graphe. En effet, tous les complexes de dimensions $n < 2$, sont des *graphes*, selon la définition :

Définition Le couple $G = (S, A)$, est un *graphe*, dont l'ensemble des sommets est S, et l'ensemble des arêtes est $A \subseteq S \times S$.

qui est similaire à celle donnée pour les complexes.

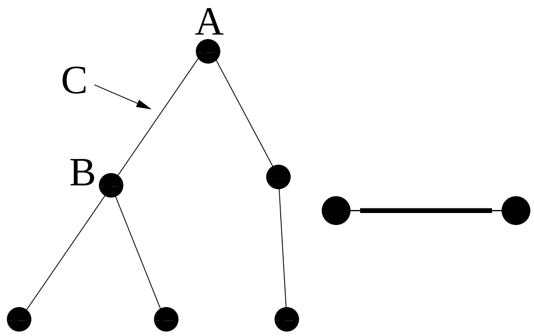
Ainsi, nous pouvons voir sur la figure V.2 que chaque partie d'un graphe est un simplexe. Les sommets sont des simplexes de dimension 0 et les arêtes des simplexes de dimension 1. De plus, les intersections entre arêtes sont bien des simplexes, le graphe en entier est donc

FIG. V.1 – Réalisations géométriques des simplexes de dimension 0 (c'est un point), 1 (c'est un arc) et 2 (c'est une surface triangulaire). Notons que chaque simplexe est appuyé sur des simplexes de dimension inférieure de 1 et contient tous ses sous-simplexes.



(a) 0-simplexe (b) 1-simplexe (c) 2-simplexe

FIG. V.2 – Un graphe est composé de 1-simplexes et de 0-simplexes. Un 1-simplexe est une arête, un 0-simplexe est un sommet.



(a) bout de graphe (b) arc ou 1-simplexe un complexe simplicial. La dimension de ce complexe est celle du plus grand simplexe qu'il contient c'est-à-dire 1 aussi.

Parmi les représentations de connaissances à base de réseaux sémantiques[3], les formalismes proposés reposent souvent sur le concept de graphe (comme les graphes conceptuels par exemple). Cette structure de représentation est en effet très commode pour représenter des hiérarchies et des ontologies. Puisque les graphes sont un cas particulier de complexes, nous verrons comment il est également naturel de représenter des hiérarchies dans le formalisme CAT, et nous verrons également que cela présente un avantage du point de vue du traitement des exceptions.

Enfin, précisons que les complexes simpliciaux ne sont pas inconnus en informatique : ils sont utilisés par exemple pour la construction de volumes 3D et ont été proposés pour la formalisation des algèbres de processus et la représentation de contraintes spatio-temporelles.

V.1.4 Chemin polygonal

Dans un complexe, deux simplexes peuvent se toucher, on dit alors qu'ils partagent une *face*. Par ailleurs, un complexe simplicial est un ensemble de simplexes dont les intersections sont des simplexes. Mais tous les simplexes ne sont pas obligatoirement connectés entre eux : c"est à dire qu'ils ne partagent pas tous une face. Ceux qui sont connectés partagent des simplexes de dimensions variables, ils seront donc plus ou moins "fortement" connectés.

De plus, deux simplexes donnés ne sont pas forcément immédiatement en contact entre eux, mais il se peut qu'ils soient en contact par l'intermédiaire d'autres simplexes. Ces intermédiaires peuvent être plus ou moins nombreux. Ces descriptions intuitives sont formalisées à l'aide de la notion de chemin polygonal :

Définition Une suite finie $\alpha = (s_0, ..., s_n)$ de simplexes d'un complexe K s'appelle chemin polygonal d'origine s_0 et d'extrémité s_n si pour tout couple de simplexes consécutifs (s_i, s_{i+1}), $s_i \cap s_{i+1} \neq \emptyset$.

La dimension du chemin α est la dimension du plus petit simplexe interscetion le long du chemin, plus un. Si $P_0 = P_n$, on dit que α est une chaîne polygonale fermée, elle définit un lacet sur $\mid K \mid$, de point-base P_0. Remarquons que les sommets d'une chaîne polygonale peuvent se retrouver plusieurs fois dans α.

V.2 Q-ANALYSE

V.2.1 Présentation

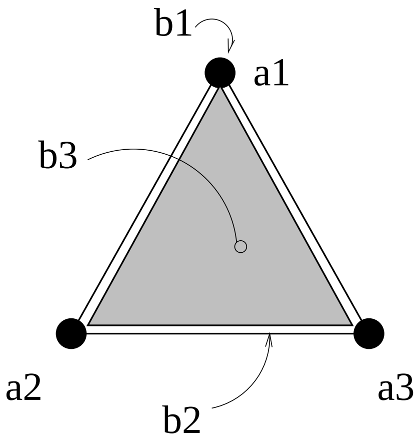
Atkin a proposé en 1977, une technique permettant de représenter des relations binaires à l'aide de complexes simpliciaux [4]. Ce type de représentation, ainsi que l'analyse des structures

[3]On distingue les formalisme à base de logique, les formalisme procéduraux, les réseaux sémantiques, et les formalismes structurés comme les scripts, frames, les langages orientés objet [60].

λ	a_1	a_2	a_3
b_1	1	0	0
b_2	0	1	1
b_3	1	1	1

TAB. V.1 – Matrice d'incidence associée à λ.

FIG. V.3 – Représentation simpliciale de la relation binaire λ définie par la matrice d'incidence de la table V.1.



connectées qu'elle exhibe, est appelé la *Q-Analyse*. Elle a déjà été utilisée pour modéliser les interactions entre agents, les relations sociales [4, 21] et les positions aux échecs [4], par exemple.

Avant d'entrer dans la partie analyse de la Q-analyse, voici comment Atkin propose de représenter les relations binaires.

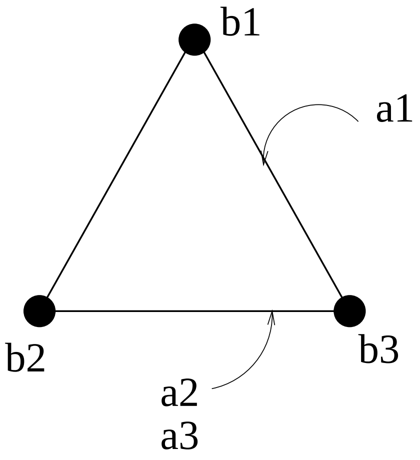
V.2.2 Représenter des relations binaires

Considérons la relation binaire λ de l'ensemble $A = \{a_1, a_2, a_3\}$ dans l'ensemble $B = \{b_1, b_2, b_3\}$, décrite par sa matrice d'incidence (table V.1). Lorsqu'un élément a_i de A est λ-relié à un élément b_j de B, la case correspondante contient un 1, sinon elle contient un 0. Dans cette relation, nous n'admettons que des valeurs binaires, ici 0/1, mais aussi *Vrai/Faux*, etc. Il est possible d'adapter la technique qui va suivre pour la représentation de matrices à plusieurs valeurs [21], par le biais de seuils par exemple, mais nous n'utiliserons pas cette extension dans nos travaux pour l'instant.

Représentons maintenant les éléments a_i de A comme des sommets dans un espace. Ils sont tous distincts. Chaque élément b_j de B peut alors être représenté à partir des sommets auxquels il est λ-relié (voir figure V.3). Par exemple, l'éleḿent b_1 est seulement relié à a_1, nous dessinerons alors b_1 comme un simplexe reposant uniquement sur le sommet a_1 (lui-même un simplexe de dimension 0). Ainsi, b_1 et a_1 sont des 0-simplexes[4], confondus. Ensuite, b_2 est λ-relié à a_2 et

[4]La dimension d'un simplexe, rappelons-le est le nombre de ses sommets moins 1. Ici il y a un seul sommet.

FIG. V.4 – Représentation simpliciale duale de la relation binaire λ par rapport à la représentation construite en figure V.3.



a_3, il sera alors représenté comme un 1-simplexe reposant sur les deux 0-simplexes a_2 et a_3, et les contenant[5]. Enfin, b_3 est λ-relié à tous les éléments de A. Il sera représenté comme un 2-simplexe (trois sommets moins 1, donne une dimension 2) reposant sur les trois sommets a_1, a_2, a_3. Un 2-simplexe est ici une face triangulaire.

V.2.3 Représentation duale

Nous pouvons également considérer la représentation duale (figure V.4) dans laquelle ce sont les éléments de B qui sont les sommets. Cette représentation duale permet de mettre en valeur d'autres structures dans l'organisation des connaissances représentées. Toujours d'après la matrice d'incidence de la table V.1, nous pouvons représenter a_1 comme un 1-simplexe reposant et contenant les 0-simplexes b_1 et b_3, a_2 comme un 1-simplexe reposant et contenant b_2 et b_3, et a_3 possède la même représentation que a_2.

Nous remarquons ici que a_2 et a_3 sont λ-reliés aux mêmes éléments de B, ils ne sont pas distinguables, ils jouent le même rôle. Cette information était *contenue* dans la représentation de la figure V.3, mais elle n'y était pas mise en valeur. En effet, si nous regardons plus en détails cette première représentation, on peut déjà voir que a_2 et a_3 jouent un rôle symmétrique. Ils sont tous les deux un sommet de b_2, et un sommet de b_3, et n'ont aucune autre relation qui les distingue. Mais il faut utiliser la reprśentation duale pour le mettre en évidence visuellement.

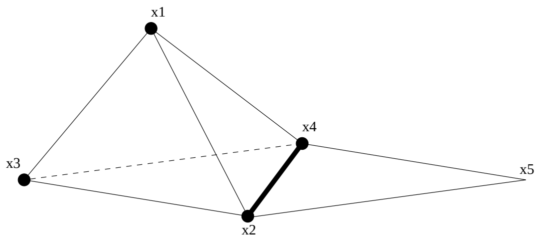
V.2.4 Q-connectivité

L'étude de toutes les parties connectées d'un CS représentant une relation binaire, constitue le centre de ce qu'Atkin appelle la *Q-Analyse* [4].

Considérons deux simplexes σ_1 et σ_2 d'un complexe simplicial K. Nous venons de voir que σ_1

[5]Attention, un simplexe contient ses bords.

FIG. V.5 – Deux simplexes connectés par un simplexe de dimension 1. Ils partagent une 1-face, qui constitue donc une chaîne polygonale de longueur 1.



et σ_2 sont connectés par une chaîne, s'il existe une séquence finie de simplexes :

$$\{\sigma_{a_1}, \sigma_{a_2}, ..., \sigma_{a_3}\}$$

tels que :

$$\begin{array}{ll} (1) & \sigma_{a_1} \leq \sigma_1 \\ (2) & \sigma_{a_n} \leq \sigma_2 \\ (3) & \sigma_{a_i} \quad \text{et} \quad \sigma_{a_{i+1}} \text{ partagent une face.} \end{array}$$

Où $\leq$ est la relation d'inclusion de simplexes. Nous parlons en Q-analyse d'une *chaîne de q-connectivité*, avec q, le plus petit des entiers a_i. Ainsi, un n-simplexe est n-connecté à lui-même par une chaîne de longueur 0, et ne peut être $(n+1)$-connecté à aucun simplexe. Prenons un exemple concret et considérons les deux simplexes :

$$Y_1 = \langle x_1, x_2, x_3, x_4 \rangle$$

et

$$Y_2 = \langle x_2, x_4, x_5 \rangle$$

Ces deux simplexes sont 1-connectés par le 1-simplexe $\langle x_2, x_4 \rangle$ (voir figure V.5).

Pour chacun des simplexes d'un complexe K donné, nous pouvons distinguer deux valeurs particulières de q-connectivité :

- La valeur q_{min} la plus petite de connectivité du simplexe à un autre simplexe dans K ;
- La valeur q_{max} la plus grande, qui est bornée par la dimension du simplexe lui-même.

Le fait de *lire* ces zones connectées entre elles par plus ou moins de longueur de chaîne, plus ou moins dimensionnée, a une correspondance en termes de représentation des connaissances, et va nous permettre de représenter des concepts de manière suffisamment naturelle, expressive et souple.

TAB. V.2 – Matrice d'incidence de la relation μ reliant les 10 premiers nombres entiers aux 4 prédicats de parité, noté p_1 , imparité, noté p_2, primalité, noté p_3, et multiplicité de 3, noté p_4.

	p_1	p_2	p_3	p_4
1	0	1	0	0
2	1	0	1	0
3	0	1	1	1
4	1	0	0	0
5	0	1	1	0
6	1	0	0	1
7	0	1	1	0
8	1	0	0	0
9	0	1	0	1
10	1	0	0	0

V.3 Représentation de prédicats

Nous avons étendu la technique de représentation introduite par Atkin, afin de permettre la représentation *d'ensembles* de *prédicats*. Considérons l'ensemble $P = \{p_1, ..., p_n\}$ de n prédicats et l'ensemble A d'objets. Il est possible de représenter la valeur de ces prédicats sur l'ensemble des objets par un complexe en considérant la relation binaire $\mu \in A \times P$ telle que l'élément de matrice $\mu(a_i, p_j) = 1$ si le prédicat p_j est vrai pour a_i.

Prenons par exemple :

$$A = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$P = \{pair, impair, premier, multiple3\}$$

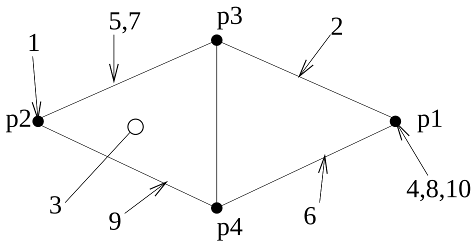
La matrice d'incidence de μ est alors donnée par la table V.2 et la représentation simpliciale en prenant les prédicats comme sommets est donnée par la figure V.6. Les nombres $4, 8, 10$ sont représentés comme équivalents car aucun prédicat ne permet de les discerner.

En effet, avec une représentation simpliciale, deux éléments ayant les mêmes propriétés (selon les prédicats de l'ensemble P) sont représentés comme équivalents. De même, deux prédicats ne sont représentés comme différents que s'il existe un objet du monde (de l'ensemble A) sur lequel leurs valeurs diffèrent. Cette propriété est *l'égalité des indiscernables.*

Cet exemple [85] était involontairement proche d'un exemple cité par P. Burmeister dans [18], à propos de l'analyse des *Formal Concepts* [103]. Cependant, il est très intéressant de pouvoir comparer les deux. En effet la technique proposée par Atkin est elle-aussi assez proche de celle utilisée en Formal Concepts Analysis (en abrégé FCA). De plus, le but de ce deuxième formalisme est également de pouvoir *visualiser* des données sur des *réseaux* de concepts [18].

Dans la théorie FCA, on commence également par considérer des relations binaires entre objets et ensembles d'attributs que l'on peut comparer à nos *prédicats.* Dans l'exemple qu'ils donnent, ils considèrent les objets :

FIG. V.6 – Représentation simpliciale de la relation μ reliant les 10 premiers nombres entiers aux 4 prédicats de parité, imparité, primalité et multiplicité de 3, dont la matrice d'incidence est celle de la table V.2.



$$O = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

et les attributs-propriétés :

$$A = \{pair, impair, premier, carre, cube\}$$

Leur travail est d'abord effectué directement sur la matrice d'incidence avant de passer à une représentation sous forme de diagramme de Hasse.

Notons que les complexes simplicaux peuvent s'exprimer eux-aussi sous forme de diagrammes de Hasse mais avec perte d'évidence spatiale. En effet, il est possible de représenter une relation d'ordre partiel sur les éléments d'un ensemble, par un graphe prenant pour sommets, les éléments de l'ensemble. Cependant, ce graphe peut être vite illisible, du fait du grand nombre d'arêtes.

Une solution pour améliorer la lisibilité du graphe, consiste alors à ne pas mentionner les arêtes redondantes. En effet, une relation d'ordre partiel vérifie les propriétés de transitivité, antisymmétrie et réflexivité, ainsi pour $a \in E, b \in E, c \in E$, alors $a < b, b < c \Rightarrow a < c$. Il est donc inutile de faire figurer les trois arcs (a, c), (a, b) et (b, c), sur le graphe puisque l'arc (a, c) peut être déduit des autres. Un graphe sur lequel les arcs (a, c) sont économisés, s'appelle un *diagramme de Hasse*, ou diagramme transitif.

Les complexes possèdent cette propriété de transitivité, puisqu'un complexe qui est contenu dans un deuxième, lui-même contenu dans un troisième, est forcément contenu dans ce troisième. En fait, on peut associer de manière canonique un treillis à un complexe : les éléments du treillis sont les simplexes ordonnés par inclusion.

Revenons maintenant à l'exemple de Burmeister. Les deux étapes principales effectuées sur les matrices avant de passer à un diagramme de Hasse, sont celles de :

- Enlever les doublons : cette opération s'appelle *purifier* la matrice ; on enlève tous les objets qui sont reliés exactement aux mêmes propriétés pour n'en laisser qu'un seul représentant. Cette propriété est *l'égalité des indiscernables*, naturelle et automatique lorqu'on passe à une reprŝentation en CS.
- Enlever les objets-propriétés : c'est l'étape de *réduction* qui consiste à enlever les objets qui ne sont reliés qu'à une seule propriété. Cette opération est immédiate lorsqu'on passe en représentation avec des CS, puisque dans ce cas l'objet considéré est identique au 0-simplexe auquel il est relié.

TAB. V.3 – Matrice d'incidence de la relation λ entre l'ensemble des fleurs F et l'ensemble des couleurs C.

λ	tulipe	coquelicot	pensée	myosotis
rose	1	0	1	1
jaune	1	1	1	0
bleu	0	0	1	1
blanc	1	1	1	1

TAB. V.4 – Matrice Λ d'incidence de la relation $\lambda \subset F \times C$.

Dans CAT, ces deux opérations ne sont pas des opérations mais des conséquences de la spacialité de la représentation. Les indiscernables se "retrouvent" au même endroit, ainsi que les objets-propriétés.

V.3.1 Exemples et analyse

Prenons un exemple simple de relation λ pour illustrer notre propos. Considérons l'ensemble F, de types de fleurs [21] :

$$F = \{\text{tulipe}, \text{coquelicot}, \text{pensée}, \text{myosotis}\}$$

et l'ensemble C, de couleurs :

$$C = \{\text{rose}, \text{jaune}, \text{bleu}, \text{blanc}\}$$

Une relation intéressante entre ces deux ensembles serait λ telle que :

la fleur f_i est λ-reliée à la couleur c_j ssi
il existe des fleurs du type de f_i ayant la couleur c_j sur terre

dont la matrice d'incidence Λ est donnée par la table V.3. Nous pouvons alors représenter les couleurs comme des complexes simpliciaux (voir figure V.7), chaque couleur étant définie par une ligne de la matrice Λ et constituant un simplexe. Le 2-simplexe représentant la couleur rose sera alors :

$$rose = \langle \text{tulipe}, \text{pensée}, \text{myosotis} \rangle$$

tandis que le 2-simplexe associé à jaune sera :

$$jaune = \langle \text{tulipe}, \text{pensée}, \text{coquelicot} \rangle$$

Nous pouvons aussi nous intéresser au complexe dual où les couleurs sont des sommets, et les fleurs des simplexes (voir figure V.8), la matrice de λ^{-1} est obtenue simplement en transposant Λ.

Nous pouvons maintenant exhiber, le 1-simplexe associé à la *frontière* entre la couleur rose et la couleur jaune. On ne peut attribuer aucun nom de couleur à ce 1-simplexe, pourtant,

FIG. V.7 – Représentation simpliciale de la relation binaire entre l'ensemble des fleurs F et l'ensemble des couleurs C.

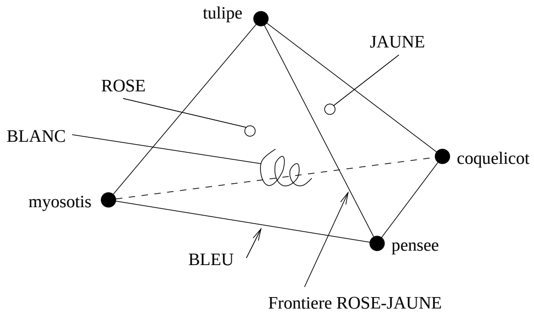


FIG. V.8 – Représentation duale pour l'exemple des fleurs : les sommets sont en minuscules et sont des éléments de l'ensemble C des couleurs. Le 3-simplexe représentant l'élément *pensee* de F, contient tous les autres, il est désigné par une flèche en « ressort » .

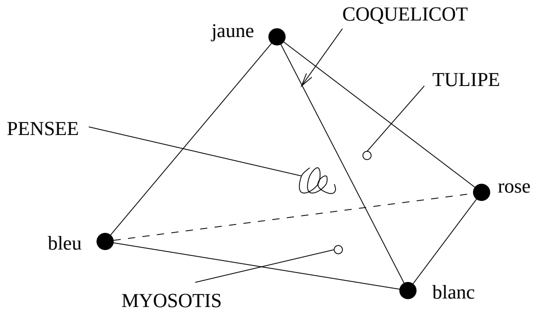
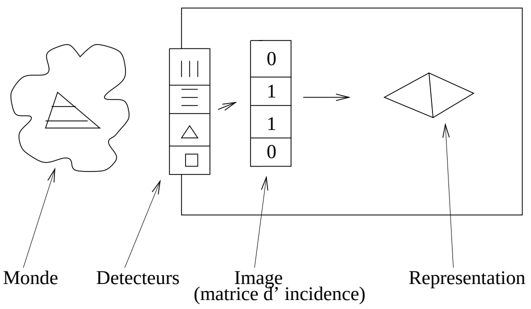


FIG. V.9 – Modèle de construction d'une représentation simpliciale par catégorisation, inspiré de la proposition de Holland [51].



il ressort clairement de cette figure et il est manipulable. Par contre, si l'on observe la table de la matrice Λ ci-dessus, les "1" définissant cette frontière existent, certes, mais ne sautent absolument pas aux yeux.

V.4 UN FORMALISME SIMPLICIAL DE REPRÉSENTATION DES CONNAISSANCES

V.4.1 Construction par catégorisation

Plusieurs auteurs ont étudié le passage entre l'« environnement » et la construction d'une représentation. Le morphisme reliant un modèle avec le monde représenté dépend du « processus même d'ordonancement du monde : la catégorisation » [42]. Il existe plusieurs approches de l'activité de catégorisation, Holland [51] propose un modèle basé sur des fonctions de catégorisation réalisant des classes d'équivalence sur les stimuli, cette proposition est illustrée par la figure V.9.

Dans ce modèle [91], les détecteurs sont des prédicats pouvant prendre les valeurs 0 ou 1 selon que les objets à détecter allument le détecteur ou non. Ces détecteurs $d_1, ..., d_n$ codent les états du monde à l'aide d'une fonction de catégorisation C qui partitionne les états du système et divise le monde en classes d'équivalence[6]. La fonction C traite donc les objets qui ne sont pas discernables grâce à ces détecteurs comme équivalents : il y a *égalité des indiscernables*.

À l'issue de l'encodage de Holland, il est aisé d'extraire une représentation simpliciale. Considérons le vecteur d'états des détecteurs associé à la perception d'un objet, comme une ligne de la matrice décrivant la relation binaire qui existe entre les objets du monde et les détecteurs. Chaque objet est alors un simplexe, et un état du système, un complexe.

[6]Un modèle basé sur une telle catégorisation met en correspondance plusieurs objets avec une représentation, c'est un homomorphisme [51].

V.4.2 Propriétés

V.4.2.1 Représenter un concept

Reprenons l'exemple des fleurs donné plus haut, et regardons encore la frontière mise en valeur sur la figure V.8. On pourrait l'interpréter, par exemple, comme le domaine commun d'extension du rose et du jaune parmi les fleurs de la nature. Dans cet exemple, cette frontière existe d'elle-même, et correspond à un concept que nous n'avions jamais eu besoin de nommer par ailleurs. Cependant, ce type de frontière peut tout aussi bien correspondre à une sous-catégorie qui a déjà reçu un nom, si nous en avions eu l'utilité.

Prenons un deuxième exemple pour illustrer cette remarque. Soit A, l'ensemble des animaux suivants :

$$A = \{\text{chameau}, \text{lézard}, \text{tigre}, \text{araignée}, \text{faisan}, \text{mésange}\}$$

et soit N, l'ensemble possible de leurs nutriments :

$$N = \{\text{eau}, \text{plante-verte}, \text{fruit}, \text{insecte}, \text{viande}\}$$

Considérons la relation : *un animal est relié à un nutriment, si ce nutriment fait partie de son régime alimentaire habituel.*

Nous pouvons alors procéder à la même analyse qu'avec les fleurs, ce qui nous mène à construire un complexe composé de simplexes. Chaque simplexe représente un animal, s'appuyant sur les sommets représentant ce dont il se nourrit. Ainsi, tous les animaux « buveurs d'eau » auront un sommet $\langle \text{eau} \rangle$ commun. Tous les carnivores auront l'arête $\langle \text{eau}, \text{viande} \rangle$ en commun, les « herbivores » auront peut être l'arête $\langle \text{eau}, \text{plante-verte} \rangle$ en commun, et ainsi de suite. Nous notons que la plupart de ces frontières portent déjà un nom précis (herbivores, carnivores, ...) car elles désignent des familles d'animaux classiques.

Ce type de représentation nous permet une *auto-organisation* des objets en concepts à partir de leur propriétés. Les concepts ne possèdent alors plus de bords déterminés et *fixes*. Cette auto-organisation mène aussi à la *découverte* de concepts sous-jacents.

Les parties q-connectées d'un complexe sont les parties qui ont des bords en commun, ces bords peuvent être interprétés comme des frontières entre concepts dans une représentation simpliciale. Les concepts sont alors représentés par ce qu'ils ont de *commun* et de *différent* avec les autres.

Selon que nous considérons une représentation en fonction des propriétés ou bien la représentations duale en fonction des objets, nous allons pouvoir définir les concepts en *extension* ou en *intension*.

En effet, un concept est défini en *extension* si l'on énumère l'ensemble des *objets* qui sont des exemples connus représentant (ou inclus dans) ce concept. C'est ce qui est visible dans l'exemple fleurs lorsque l'on représente les fleurs comme des sommets et les propiétés de couleur comme des simplexes construits dessus. La propriété "jaune" est un simplexe contenant des *exemples* de "jaunesse".

Un concept est défini en *intension* si l'on donne les propriétés communes à tous ses membres. C'est ce qui est visible dans l'exemple des nombres lorsque les propriétés sont représentées comme des sommets et les nombres par dessus. Dans cet exemple, le simplexe $\langle 6, 7 \rangle$ contient les simplexes $\langle p_1 \rangle$ et $\langle p_4 \rangle$, il existe un concept défini comme vérifiant ces deux propriétés (et contenant aussi les nombres $4, 8, 10$ qui sont *identifiés* avec les simplexe $\langle p_1 \rangle$).

Dans le formalisme FCA [18], les concepts sont également définis en extension par l'ensemble des objets qui appartiennent au concept et en intension par l'ensemble des attributs valides pour ces objets. Un concept est un trio $c = (Obj, Att, Relation)$, ainsi un concept *est* la matrice d'incidence. Cette expression des concepts est proche de la notre, mais leur représentation de ces concepts n'est pas naturelle et ne permet pas de mettre directement en avant les propriétés partagées et de comparer ces concepts entre eux, d'étudier leurs relations.

Maintenant que nous savons "regarder" les CS pour définir des concepts, comment peut-on les organiser ?

V.4.2.2 Hiérarchies de classes

Puisque les complexes sont une généralisation des graphes à plusieurs dimensions, nous pouvons tout naturellement représenter des hiérarchies de concepts à l'aide de CS.

Plaçons nous par exemple dans le cadre des langages orientés objets en informatique. Les objets sont alors définis avec des attributs qu'ils héritent de leurs parents, et des attributs qui leur sont propres. Une classe est alors un ensemble d'objets qui partagent un ensemble d'attributs. La représentation d'une classe se fera tout naturellement par le biais de la représentation des attributs qui la caractérisent.

La relation d'héritage des classes est alors la relation d'inclusion des complexes. En effet, un classe A définie par un seul trait (feature en anglais), sera identifiée à un 0-simplexe par exemple. Une classe B héritant de cette classe, mais ayant en plus un deuxième trait contiendra sa classe mère, mais sera plus grande qu'elle de une dimension (attriubée à ce deuxième trait). Ainsi les classes les plus hautes dans la hiérarchie seront celles qui sont contenues par toutes les autres, celles qui se trouvent à leur intersection. Les classes les plus particularisées sont celles qui ont besoin du plus grand nombre de traitspour être spécifiées, elles sont donc représentées par des simplexes plus vastes, car nécessitant plus d'information.

Les critères suivants sont suffisants pour contraindre l'obtention effective d'une hiérachie [90], [69] :

- Il doit y avoir un partage maximal des attributs entre les classes.
- Un nombre minimal de simplexes sont distingués.
- Une classe hérite du maximum de classes qu'elle contient.
- Les objets concrets sont les simplexes maximaux de la représentation.

Nous voyons sur la figure V.10, que l'attribut f_1 est partagé par toutes les classes. Il est en effet sommet du simplexe $\langle f_1, f_5 \rangle$, du simplexe $\langle f_1, f_2, f_3 \rangle$ et du simplexe $\langle f_1, f_2, f_4 \rangle$, de même, $\langle f_1, f_2 \rangle$ est commun ensuite à $\langle f_1, f_2, f_3 \rangle$ et $\langle f_1, f_2, f_4 \rangle$. Enfin, les classes, sont bien des simplexes maximaux qui se distinguent par au moins un attribut chacune.

FIG. V.10 – Représentation simpliciale d'une hiérarchie de classes à l'aide des traits (features) caractérisant chaque classe.

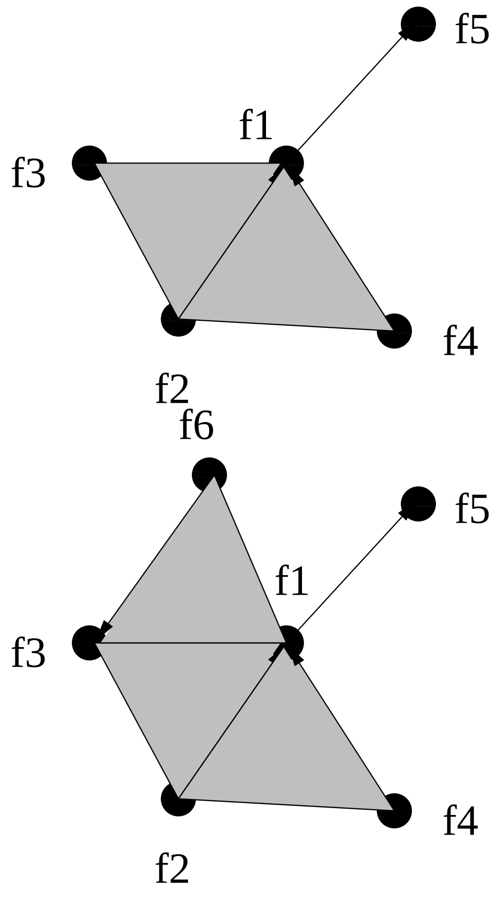
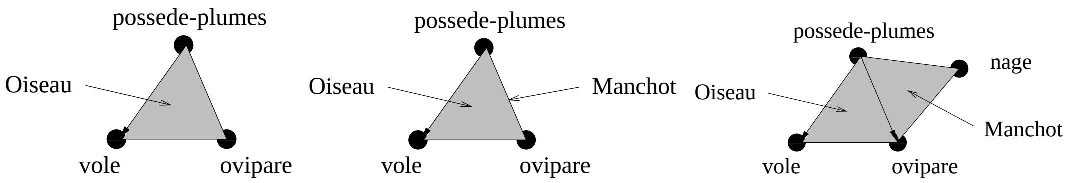


FIG. V.11 – Représentation simpliciale des exceptions. Ici, le manchot est différencié de la classe oiseau (définie en (a)) soit par un attribut en mois (figure (b)) comme le fait de voler, soit aussi par un attribut en plus (figure (c)) comme le fait de nager.



V.4.2.3 Traitement des exceptions

Le traitement des exception est un sujet qui pose souvent problème dans les systèmes de représentation des connaissances. En effet, soit on est obligé d'ignorer tout simplement cet aspect de la représentation des connaissances pour avoir une hiérarchie de classes rigoureuse, soit on est obligé de modifier la hiérarchie aux dépends de ses propriétés fondamentales d'héritage.

Ce problème vient souvent du fait qu'une classe est nécessairement nommée, étiquettée [16], et que ses héritiers sont des branches qui partent de cette étiquette. Nous avons vu ici, que les concepts sont définis au choix en intension (par les attributs partagés par les exemples connus) ou en extension (par les exemples connus), sans avoir explicitement recours à un étiquettage direct de chaque concept. En effet, un étiquetage pose toujours le problème de la granularité d'une représentation que nous aborderons dans la sous-section suivante.

Dans CAT, il est possible de décrire un élément en tant qu'*exception* par rapport à une classe, puisque nous le représentons par ce qu'il a de *différent* par rapport aux éléments définissant cette classe.

Par exemple, considérons la classe oiseau définie en intension par trois propriétés, "possède-plumes", "vole" et "ovipare". Cette classe est représentée sur la figure V.11 a). Essayons de représenter l'exception "manchot" par rapport à cette classe. Il ne vérifie que les propriétés "possède-plumes" et "ovipare", il est donc représenté comme un 1-simplexe appartenant au simplexe "oiseau". Cela représente une sous-classe qui ne vérifie que ces deux caractères et dont la classe oiseau hérite (figure V.11 b)). Cela aurait été impossible si "oiseau" était déjà une propriété atomique. Mais elle est heureusement elle-même, dans tous les cas, exprimable en fonction de propriétés plus fines.

La reprśentation de l'exception sera différente de celle de tous les autres objets entrant dans la classe "oiseau". Cette différence peut même se traduire par une autre propriété en plus comme "nage" par exemple comme sur la figure V.11 c). Ici l'objet "manchot" et les objets de "oiseau " partagent encore une sous-classe définie par les propriétés "oiseau" et "ovipare", mais cette sous-classe n'est pas celle à laquelle appartient "manchot". On peut alors dire qu'un manchot est une exception à la classe des oiseaux car il ne vole pas, ou bien qu'un manchot est un oiseau qui ne vole pas *et* qui nage.

V.4.2.4 Plasticité

Une bonne caractéristique d'un formalisme de représentation des connaissances doit être son *exhaustivité*. C'est-à-dire qu'à chaque objet du problème traité, doit correspondre une représentation dans le modèle. Si cette exhaustivité n'est pas atteinte, comme c'est toujours le cas même dans les système cognitifs réels, on doit pouvoir ajouter des connaissances nouvelles au modèle de manière facile et consistante.

Nous avons vu que dans le cas du formalisme CAT, une connaissance nouvelle peut être représentée par rapport aux attributs (ou propriétés) de la représentation en cours, s'insérer dans une classe, sans la contenir totalement. Ce nouvel exemple peut ne contenir qu'une portion de classe correspondant au nombre d'attributs réels qu'il partage avec cette classe.

Un ré-étiquetage n'est pas nécessaire et pourtant un "nouveau" concept est représenté qui est l'intersection de ce nouvel objet (exemple) avec la classe à laquelle il appartient presque. L'ancienne classe hérite de cette nouvelle classe ainsi que ce nouvel objet, la hiérarchie est réorganisée sans avoir besoin de repenser la granularité du système.

V.4.2.5 Abstraction

La granularité de départ d'une représentation est souvent difficile à doser. En effet, trop forte, elle posera des problèmes d'efficacité d'exécution. Notons que ces problèmes peuvent sembler sans importance pour certaines modélisations non appliquées, mais il faut tout de même ne pas dépasser une limite qui interdirait la plausibilité cognitive du modèle[7]. Tandis qu'une granularité trop faible posera à nouveau des problèmes d'expressivité du formalisme.

La granularité est pourtant aussi un problème initial dans CAT, puisqu'il faut choisir la granularité pour les propriétés qui vont nous servir de *base* pour exprimer tous les objets rencontrés.

Dans un CS, chacun des sommets peut être considéré comme étant lui-même représentable avec une granularité plus fine. En effet, cela correspond à la simple opération algébrique de changement de base. Si une granularité est jugée insuffisante à une étape de la construction de la représentation, on peut projeter toute la représentation courante selon une nouvelle base, plus fine en exprimant les anciennes propriétés en fonction des nouvelles.

Nous avons fait un tour des caractéristiques de la représentation des connaissances basée sur le formalisme CAT. Nous allons voir maintenant quelles sont les opérations qui sont possibles naturellement dans CAT sur ces représentations. nous verrons ensuite quels type de *raisonnements* nous pouvons avantageusement modéliser avec ce formalisme.

V.5 Conclusions du chapitre

Le formalisme CAT semble suffisament riche pour prétendre supporter une représentation des connaissances basée sur la métaphore spatiale. Nous l'avons choisi pour des raisons de naturalité et d'expressivité, et nous l'avons étudié pour tester ses autres propriétés. Il semble se comporter de façon prometteuse pour la représentation des hiérarchies, le traitement des exceptions, l'ajout de connaissances nouvelles, qui peuvent intervenir dans des applications

[7]De plus en plus de modèles accordent de l'importance à l'aspect efficacité d'un modèle car il est lié au paramètre de temps de réponse mesurable par les psychologues. Un temps de réponse trop long suppose que le mode de résolution n'est pas le plus naturel possible. Pour rendre compte de l'adaptabilité aux problèmes et à la rapidité de résolution des systèmes cognitifs modélisés, il faut se contraindre à des modèles performants [97]

utilisant la connaissance pour un raisonnement par analogies. En effet, d'après ce que nous avons dit au chapitre précédent sur l'analogie, nous cherchons à représenter des structures, afin des les apparier.

En ce qui concerne le modèle de construction de représentations simpliciales, nous soulignons les propriétés communes entre le modèle de Holland [51] et les complexes simpliciaux qui nous permettent de construire des représentations qui nous semblent naturelles et adaptées à l'expression et à l'utilisation de relations spatiales.

Chapitre VI

Le système ESQIMO

Introduction

Dans ce chapitre, nous présentons le modèle pour le raisonnement par analogies (défini au chapitre IV) basé sur les concepts de topologie algébrique présentés au chapitre V.

Le problème choisi pour modéliser le raisonnement par analogies est la résolution de tests de Q.I. dans sa version la plus simple. Ce genre de problème consiste à compléter une série de trois figures. Le raisonnement utilisé lors de cette tâche est analogique et non supervisé (la solution n'est pas choisie parmi une liste de propositions).

Notre objectif est de concevoir et de développer un système qui utilise des outils de la topologie algébrique pour résoudre ce type de problème. Notre système, nommé ESQIMO[1], a été prototypé en `Mathematica` et implémenté en `Caml`. Nous exposerons sa conception et donnerons de nombreux exemples de réponses produites par ESQIMO.

Notre approche pour résoudre diagrammatiquement ce type de problème consiste à représenter le rapport de deux figures par un chemin dans un certain espace. La création d'une quatrième figure par analogie consiste alors à déformer une figure le long de ce chemin.

Résumé du chapitre

Nous commençons par décrire précisément le problème, puis nous esquissons quelques uns des très nombreux travaux qui ont été menés sur l'analogie. Nous présentons ensuite les réponses aux tests de Q.I. fournies par des sujets humains, au cours d'une courte expérience que nous avons organisée. Ces réponses nous permettront d'apprécier la diversité des justifications produites.

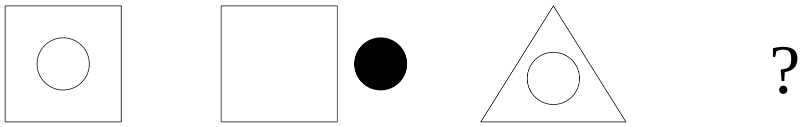
Dans la troisième partie du chapitre, nous décrivons précisément l'approche du système ESQIMO, ainsi que l'algorithme utilisé pour produire diagrammatiquement l'analogie. Nous donnons ensuite des exemples commentés de résolutions de tests par ESQIMO.

[1]L'acronyme ESQIMO est choisi pour Espace QI Modèle, sa signification est expliquée plus tard dans ce chapitre.

FIG. VI.1 – Présentation d'un énoncé typique des questions posées dans les tests de Q.I., avec des éléments de nature numérique.

6 12 10 ?

FIG. VI.2 – Présentation d'un énoncé typique des questions posées dans les tests de Q.I., avec des éléments de nature géométrique.



L'algorithme ESQIMO est ensuite largement discuté. Nous examinons comment il pourrait être étendu, ainsi que les hypothèses sur lesquelles il repose. Enfin, plusieurs généralisations sont proposées.

VI.1 Les problèmes du type test de Q.I.

VI.1.1 Définition et présentation

Le problème que nous nous proposons d'étudier revient plus généralement à résoudre une analogie à quatre membres, du type de l'énoncé E_0 suivant :

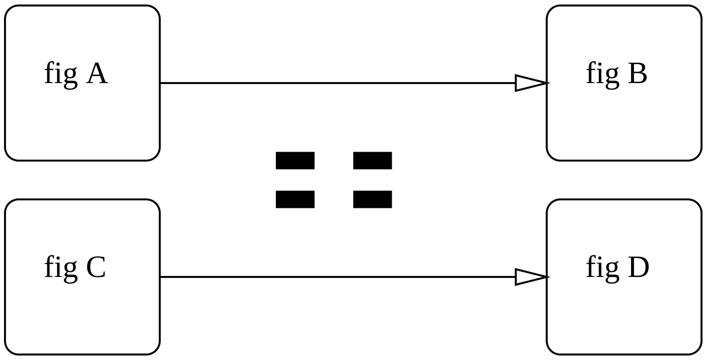
"A est à B, ce que C est à D."

Où les objets A, B et C sont donnés et peuvent être composés d'éléments d'un univers d'objets possibles Ω qu'il faut alors définir pour spécifier complètement le problème E_0. Les figures VI.1 et VI.2 montrent deux énoncés typiques de ce genre de test, E_1 basé sur des éléments de nature numérique, et E_2 basé sur des éléments de nature géométrique.

Il s'agit, lors de cette tâche, de compléter une suite de trois figures par une 4^e de sorte que cette figure soit à la troisième, ce que la 2^e est à la 1^e. Plus exactement, il est typiquement présenté au sujet un premier couple de figures A et B qui sont affirmées être en rapport non précisé. Ensuite, une troisième figure C est présentée pour être complétée par le sujet en un couple (C, D) qui vérifie le même rapport que le couple (A, B), à déterminer (voir figure VI.3).

Pour la modélisation, nous considérons que le sujet doit obligatoirement passer par au moins une étape intermédiaire où il s'intéresse au rapport entre les figures A et B. Ce rapport

FIG. VI.3 – Rapport d'analogie dans un test de Q.I géométrique.



est ensuite adapté pour être appliqué à C et ainsi obtenir une solution considérée comme acceptable. Ce qui va nous intéresser ici est justement le jeu de rapports existant entre A, B, C et D.

Nous ne cherchons pas à savoir ce que mesure ce genre de test. De plus, nous ne nous intéresserons pas à une solution unique mais à toutes les solutions engendrables dans la mesure où nous sommes capables de les justifier.

Le rapport analogique à résoudre dans ce genre de problèmes, est schématisé par le tableau VI.3, où l'on voit que la relation entre A et B est analogue à la relation entre C et D.

On décompose habituellement la résolution de cette analogie en quatre étapes :

1. Déterminer la(les) relation(s) possible(s) entre A et B : R_{AB}.
2. Déterminer la(les) relation(s) possible(s) entre A et C : R_{AC}.
3. Appliquer R_{AB} à C en se restreignant au domaine déterminé grâce à R_{AC}.
4. Vérifier la correspondance entre B et D par la relation R_{AC}.

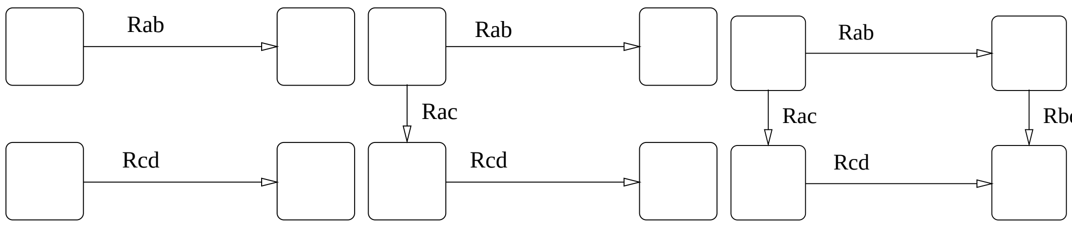
En fait, il existe plusieurs niveaux de résolution de cette analogie. À chaque niveau, on considère une symétrie supplémentaire dans le rapport, ce qui est schématisé par le tableau VI.3. Au premier niveau, c'est la relation R_{AB} qui est analogue à la relation R_{CD}. Au deuxième niveau, il faut considérer la relation R_{AC} afin de déterminer le domaine de C sur lequel s'applique l'analogie. Au troisième niveau, enfin, on considère que la relation R_{BD} doit être l'analogue de la relation R_{AC} et que ce rapport est parfaitement symétrique.

VI.1.2 Le système ANALOGY

Le système ANALOGY [46] proposé par Evans en 1968, et que nous avons déjà présenté au chapitre IV, page 104, est destiné à résoudre des tests d'intelligence fondés sur des analogies géométriques et graphiques. Il fonctionne à partir de la connaissance de :

- Relations topologiques ;
- Relations et transformations géométriques ;
- Modifications (suppression ou addition de tracé).

FIG. VI.4 – Trois niveaux de l'analogie $A : B :: C : D$: Le premier niveau montre une analogie où seules les relations R_{AB} et R_{CD}, sont analogues entre elles. Le deuxième niveau tient compte du rapport R_{AC}, dans la détermination de l'analogie entre les relations R_{AB} et R_{CD}. Le troisième niveau impose une analogie complètement symétrique, les relations R_{AB} et R_{CD} sont analogues, et les relations R_{AC} et R_{BD} aussi.



La tâche consiste à sélectionner la bonne figure parmi un choix de 5 figures de sorte qu'elle complète correctement l'analogie. La recherche des solutions s'effectue suivant la démarche suivante :

- Génération de règles de transformation possibles entre A et B ;
- Génération de règles de transformation permettant de passer de C à chacune des figures proposées ;
- Comparaison des deux jeux de règles et sélection de la meilleure solution.

Plusieurs règles de transformation peuvent être associées à un couple de figures. De plus, la dernière étape de comparaison implique une certaine mesure de *similitude*. Enfin, remarquons qu'il se peut que le système ne trouve pas de solution adéquate.

Les auteurs insistent sur le fait qu'il y a un choix à effectuer parmi un certain nombre de solutions proposés. En fait, il serait impossible de laisser le sujet construire complètement une solution dans ce genre de tests car il serait bien difficile de juger de l'exactitude de la solution.

Nous avons un avis différent sur ce point puisque nous pensons qu'il est plus intéressant de construire une ou plusieurs solutions à condition que nous puissions les *justifier*. Une justification est une *explication* de la procédure de raisonnement qui a permi *d'aboutir* à une solution donnée. Une solution ne peut pas être évaluée dans l'absolu pour des problèmes de type tests de QI, le plus important pour un sujet est d'être *capable* de fournir une justification de la solution, et que cette justification soit ensuite correctement *appliquée* au problème.

VI.2 Étude expérimentale

Avant de nous lancer dans l'implémentation de la simulation d'un processus cognitif, nous devons procéder à une expérimentation informelle.

L'objectif n'est pas d'analyser l'aptitude des humains à résoudre ce genre de tests, mais simplement d'illustrer la diversité des réponses qui peuvent être produites quand le test n'est pas supervisé (c'est-à-dire, lorsque nous ne proposons pas de choisir la solution dans une liste prédéfinie, mais qu'il faut la construire soi-même).

FIG. VI.5 – Expérience 1

FIG. VI.6 – Expérience 2

VI.2.1 Le protocole

Nous avons effectué un test sur un échantillon non représentatif de 10 personnes, tous des informaticiens de sexe masculin entre 20 et 34 ans. Cependant, le but n'est pas d'en tirer une analyse de la population testée ni même des conclusions quant à ce que ce test est susceptible de mesurer. Nous cherchons uniquement à constituer une base d'exemples des nombreuses solutions qu'un humain peut produire, afin de posséder un matériel varié pour analyser les types de démarches différentes dans le raisonnement *diagrammatique* [8].

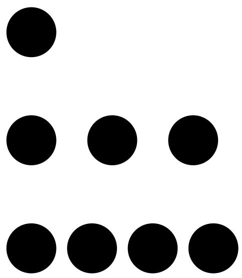
Le test se compose de quatre devinettes graphiques du type test de Q.I. Une illustration de cette série de tests est présentée sur la figure VI.5. Sur un si petit échantillon, les réponses sont déjà variées et nous nous sommes intéressés, dans un deuxième temps, aux justifications qui accompagnaient ces réponses.

La feuille de tests de Q.I. soumise à l'expérimentation sur 10 sujets inclut les séries de figures A, B, C des figures VI.5, VI.6, VI.7 et VI.8. Elles doivent à chaque fois être complétées par une figure D, de sorte que les 4 figures vérifient le rapport $A : B :: C : D$. La figure D doit être *construite* (aucun choix de figures n'est proposé). De plus, toute justification et toute réponse est à considérer car elle représente une transformation possible faisant passer de A à B et il s'agit de les envisager toutes.

FIG. VI.7 – Expérience 3

FIG. VI.8 – Expérience 4

FIG. VI.9 – Réponses données au test de QI de l'expérience 1 de la figure VI.5



VI.2.2 Résultats et commentaires

Aucun des tests de la série n'a donné lieu à une réponse unique parmi les 10 sujets. Ceci illustre bien le fait qu'il ne peut en aucun cas exister une seule réponse considérée comme correcte d'après l'énoncé.

Les résultats possibles pour chacune des séries sont donnés par les figures VI.9, VI.10, VI.11 et VI.12. Nous numérotons les solutions possibles pour chaque expérience avec une lettre. Par exemple la première ligne de la figure VI.9, sera désignée par $(1a)$, etc. Afin de préciser les fréquences correspondantes sur la table suivante :

FIG. VI.10 – Réponses données au test de QI de l'expérience 1 de la figure VI.6

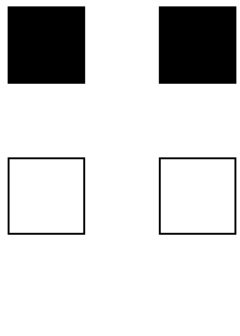


FIG. VI.11 – Réponses données au test de QI de l'expérience 1 de la figure VI.7

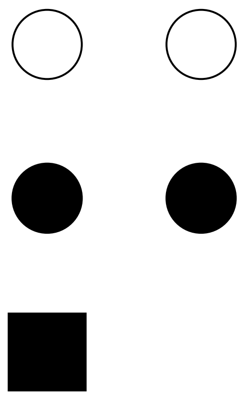
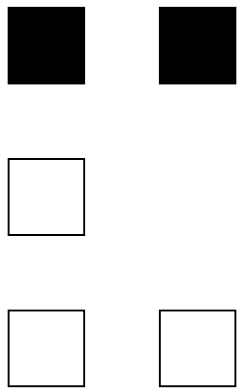


FIG. VI.12 – Réponses données au test de QI de l'expérience 1 de la figure VI.8



	(a)	(b)	(c)	Total
Expérience 1	4	2	4	10
Expérience 2	9	1		10
Expérience 3	8	1	1	10
Expérience 4	8	1	1	10

Les différentes solutions données pour un même énoncé ont ensuite été justifiées par les sujets. Voici les justifications fournies pour le premier énoncé, à partir d'une relation supposée exister entre A et B, appliquée ensuite à C :

- $(1a)$: Le raisonnement porte uniquement sur le nombre de figures et pas sur leur nature. Les sujets disent s'être placés dans un univers binaire, après 2, il y a 1. La relation proposée entre A et B est d'ajouter 1, lorsqu'on ajoute 1 à C, cela fait 3, donc 1, en binaire.
- $(1b)$: Même relation entre A et B, ajouter 1, mais dans un univers décimal (par exemple).
- $(1c)$: La relation entre A et B est maintenant de passer du nombre d'éléments à son double. Ainsi C, contenant 2 figures, en contient 4 de même nature.

Dans cet exercice, les sujets ont raisonné sur le nombre d'éléments puisque la nature des éléments entre les figures A et B ne change pas. Une même solution peut aussi être justifiée de manières différentes. La solution $(1b)$ peut être perçue comme une incrémentation ou comme une multiplication par deux du dernier élément. Ces deux justifications donnent le même résultat ici mais auraient donné un résultat différent si les deux éléments de la figure C avaient été de nature différente. Regardons maintenant les justifications proposées pour le deuxième énoncé :

- $(2a)$: L'univers est encore considéré comme binaire, mais cette fois dans le sens où il n'existe comme figures possibles que des carrés et des ronds. Dans la mesure où l'exemple propose un changement de forme de carré vers rond, il faut changer cette fois la forme de rond vers carré. Cette première relation est combinée à une relation sur le nombre d'éléments, qui passe de un à deux.
- $(2b)$: Étrangement, la couleur des figures est changée dans cette solution. L'argument est que la couleur de la figure d'arrivée est blanche, il faut donc garder cette caractéristique.

Nous voyons à partir de cet énoncé, que les attributs des figures peuvent être considérés comme déterminés ou indéterminés. Par exemple, ceux qui ont répondu $(2a)$, considèrent qu'il n'y a pas lieu de changer la couleur et conservent la couleur noire. Ceux qui ont répondu $(2a)$, considèrent que même si la couleur entre A et B n'a pas changé, il faut tout de même s'intéresser à la couleur de la figure d'arrivée et prendre la même pour D. Certains s'intéressent aux attributs de la figure d'arrivée, d'autres aux attributs qui ont changé seulement entre A et B.

Pour le troisième énoncé :

- $(3a)$: Ici la relation considérée, consiste en le changement de forme qui passe de rond à carré ou de carré à rond dans un univers à deux seules formes possibles, ainsi que la multiplication par deux du nombre des éléments.
- $(3b)$: La relation est la même que pour la réponse $(3a)$, à ceci près que le changement de couleur est également pris en compte ici.

(3c) : Seul le changement de couleur a été appliqué à C.

Enfin, voici les justifications fournies pour le quatrième énoncé :

(4a) : Changement de couleur, de forme et multiplication par deux.
(4b) : Changement de forme.
(4c) : Changement de forme et multiplication par deux.

Nous pourrions considérer qu'une solution n'est juste que si elle fait intervenir la plus grande relation entre A et B, c'est-à-dire que le fait d'omettre le changement de couleur, ou bien la multiplication par deux dans le cas de la réponse (4b), sont des erreurs. Ici, nous ne tirerons pas ce genre de conclusion, ce sont bien tous ces processus sans exclusion que nous cherchons à modéliser, ils sont tous à inclure dans ce que nous avons appelé le raisonnement *diagrammatique* [8], [24].

VI.3 Présentation du système ESQIMO

Nous avons implémenté un système de résolution de tests de Q.I. en modélisant les processus du raisonnement par analogie à l'aide des outils de la topologie algébrique présentés dans le chapitre V. Le système aini réalisé se nomme ESQIMO pour **ES**pace-**QI**-**MO**dèle.

VI.3.1 Modélisation des données du problème

VI.3.1.1 Les données du problème

Pour modéliser ces raisonnements, nous nous limitons à un univers géométrique simple. Ces éléments seront les seuls connus dans l'univers d'ESQIMO. Les éléments géométriques sont choisis pour être les mêmes que ceux présentés dans l'étude de Weber [100], car il a l'avantage de la simplicité tout en engendrant une variété nécessaire pour nos commentaires. Comme lui, nous allons choisir un petit nombre de propriétés géométriques de base sans tenir compte des considérations métriques, des orientations dans l'espace ou des distances.

En fait, nous ne devrions pas parler d'éléments en tant qu'individus. D'après ce que nous avons dit dans la section précédente, ce sont les propriétés que les éléments partagent entre eux qui vont nous intéresser.

Nous choisissons des propriétés minimales de *forme*, de *taille* et de *couleur*. Chaque propriété p_i sera représentée simplicialement dans le système ESQIMO à l'aide des *instances* e_j qu'il connaît et dont il sait qu'elles vérifient p_i. Cet ensemble d'instances pourrait être augmenté d'autres instances si elles étaient nouvellement perçues par ESQIMO. Soit P l'ensemble des propriétés connues dans ESQIMO :

- Propriétés de forme : p_1 : rond ; p_2 : carré ; p_3 : triangle ;
- Propriétés de couleur : p_4 : blanc (ou creux) ; p_5 : noir (ou plein) ;
- Propriétés de taille : p_6 : petit ; p_7 : grand.

Il y a donc 7 éléments dans P. Remarquons que dans cette famille de propriétés nous ne tenons pas compte de la position relative des éléments dans une figure (au-dessus de, à gauche de, ...).

Les sept propriétés sont traitées de manière indépendante les unes des autres de sorte que nous incluons le minimum de connaissance a priori dans notre système. En effet, ESQIMO ne regroupe pas les propriétés selon qu'elles ont trait à la forme, la couleur ou la taille d'un élément. De plus, nous avons choisi trois formes pour que toutes les relations ne soient pas complémentaires deux à deux. Nous nous donnons des éléments représentant l'intersection de ces propriétés. Nous pourrions en imaginer plusieurs pour une intersection mais nous en choisissons le minimum, c'est-à-dire un seul. Ces éléments doivent être perçus comme des classes dans la terminologie objet, les instances correspondant aux éléments effectivement présents dans les figures A, B, C et D.

VI.3.1.2 L'ensemble des éléments Ω

La totalité des éléments manipulés par le système est présentée sur la figure VI.13, c'est l'univers des éléments connus, appelé Ω :

$$\Omega = \{e_1, e_2, ..., e_{12}\}$$

Cela veut dire que les figures A, B, C ainsi que la solution D à construire par le système, sont toujours composées à partir d'éléments de Ω. La solution n'est donc pas supervisée, mais les éléments de la solution, eux, le sont.

VI.3.1.3 Représentation des propriétés

Nous construisons une représentation des connaissances qu'ESQIMO possède sur l'univers des éléments Ω et sur les propriétés de P sous forme de complexe simplicial, noté C_Ω.

Pour le construire, nous commençons par construire le simplexe de chacune des relations p_i. Par exemple, la propriété d'être *rond* sera représentée par un simplexe $S_1 = S_{rond}$ dont tous les sommets seront les représentants connus par notre système comme ayant la propriété *rond*. Ce simplexe possède quatre sommets e_1, e_4, e_7, et e_{10}, il est donc de dimension 3 :

$$S_1 = S_{rond} = \langle e_1, e_4, e_7, e_{10} \rangle$$

Un 3-simplexe est représenté graphiquement par un tétraèdre plein, ainsi, la propriété de rondeur est illustrée par la figure VI.14.

De même, les propriétés p_2 et p_3 d'être *carré* ou *triangulaire*, sont représentées par des 3-simplexes selon la même technique :

$$S_2 = S_{carre} = \langle e_2, e_5, e_8, e_{11} \rangle$$

$$S_3 = S_{triangulaire} = \langle e_3, e_6, e_9, e_{12} \rangle$$

FIG. VI.13 – Éléments de base manipulés par le système ESQIMO : c'est le produit cartésien des ensembles d'instances possibles compte tenu des 3 formes, 2 couleurs et 2 tailles possibles (7 propriétés de P), de cardinalité $2 * 2 * 3 = 12$.

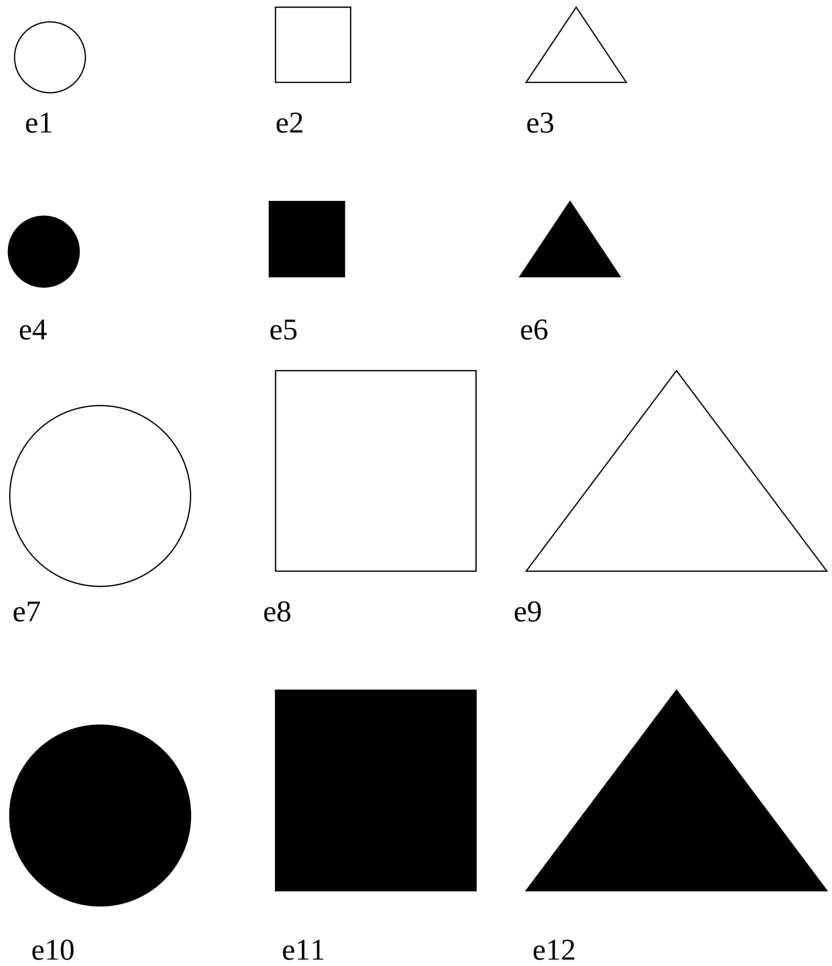


FIG. VI.14 – Représentation simpliciale $S_1 = S_{rond}$ de la relation de rondeur p_1 dans ESQIMO : nous avons dessiné la forme géométrique des éléments e_1, e_4, e_7, et e_{10}, pour faciliter la compréhension. Ce simplexe est une partie de C_Ω.

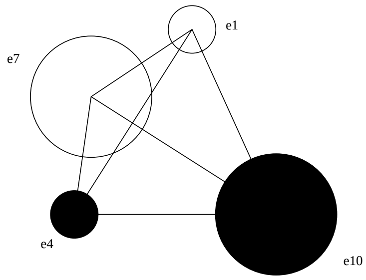
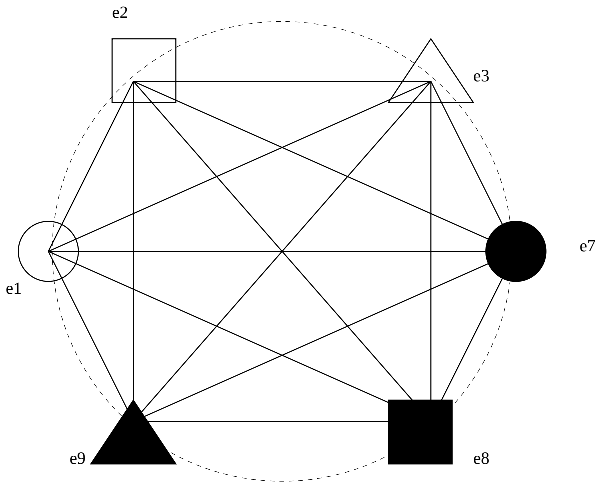


FIG. VI.15 – Représentation polygonale du simplexe $S_6 = S_{petit}$: nous avons dessiné la forme géométrique que chaque sommet est censé représenter, pour faciliter la compréhension. Attention : ceci n'est pas un graphe plan mais un polyèdre de dimension 5.



Par contre, les simplexes représentant les autres relations auront 6 sommets chacun et seront donc des 5-simplexes :

$$S_5 = S_{plein} = \langle e_7, e_8, e_9, e_{10}, e_{11}, e_{12} \rangle$$

$$S_4 = S_{creux\langle e_1, e_2, e_3, e_4, e_5, e_6 \rangle}$$

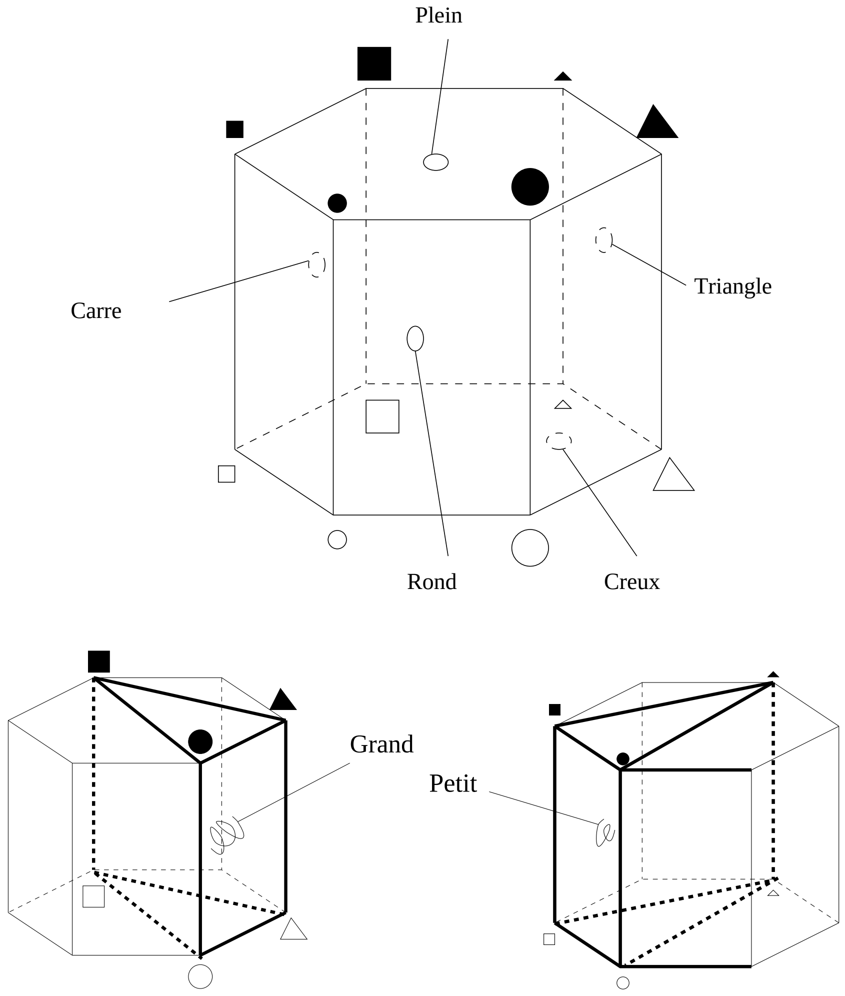
$$S_6 = S_{petit\langle e_1, e_2, e_3, e_7, e_8, e_9 \rangle}$$

$$S_7 = S_{grand\langle e_4, e_5, e_6, e_{10}, e_{11}, e_{12} \rangle}$$

Il est difficile de représenter ces simplexes autrement que grâce à la représentation en cercle, introduite au chapitre V. À titre d'exemple, nous avons représenté de cette façon le simplexe S_{petit} correspondant à la propriété *petit* sur la figure VI.15.

Comme nous l'avons remarqué précédemment, nous pouvons représenter toutes les propriétés de P sur un seul et même complexe C_Ω, en utilisant des dimensions supplémentaires pour chaque prédicat p_i différent. Ainsi, les éléments de l'univers sont représentés dans toutes les structures auxquelles ils participent. Nous représentons donc l'état des connaissances sur les éléments de l'univers Ω, par C_Ω, (voir figure VI.16 pour une idée intuitive) à l'aide de la matrice d'incidence $\Lambda_{ESQ} : P \times \Omega$ ci-dessous :

FIG. VI.16 – Une représentation 2D de la représentation simpliciale C_Ω de Λ_{ESQ} : les simplexes S_{plein} , S_{creux}, S_{petit} et S_{grand} ne sont pas des graphes plans mais des simplexes de dimension 5.



Λ_{ESQ}	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}
rond	1			1			1			1		
carré		1			1			1			1	
triangle			1			1			1			1
creux	1	1	1	1	1	1						
plein							1	1	1	1	1	1
grand				1	1	1				1	1	1
petit	1	1	1				1	1	1			

FIG. VI.17 – Représentation simpliciale S_4 d'une figure simple composée d'un seul élément e_4 (voir définition de e_4 par la figure VI.13).

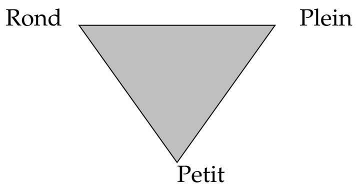
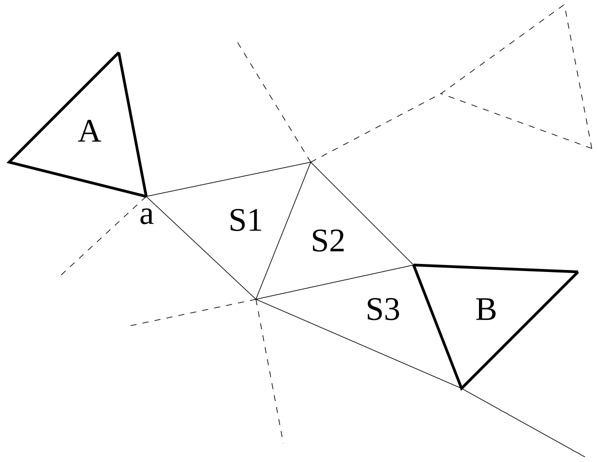


FIG. VI.18 – Exemple abstrait d'une adaptation d'un simplexe S_A en un simplexe S_B, le long du chemin polygonal trouvé entre les deux : le simplexe S_A se déforme en le simplexe S_B, le long d'un chemin $(S_A, S_1, S_2, S_3, S_B)$, de dimension 1 et de longueur 5. En effet, l'intersection entre S_A et S_1 est réduite au 0-simplexe $\langle a \rangle$. Quand S_A se transforme en S_1, seule la propriété a est préservée.



VI.3.2 Transformations de représentations simpliciales pour l'adaptation

VI.3.2.1 Figures simples à un seul élément

Considérons, dans un premier temps, la transformation simple qui permet de passer d'une figure A simple composée d'un seul élément e_i à une autre figure simple B composée seulement d'un élément e_j. En utilisant la matrice d'incidence. Une figure est représentable par un simplexe (voir figure VI.17 pour la représentation simpliciale de e_4 par exemple), et ce simplexe appartient forcément à C_Ω. Ainsi, trouver une transformation de e_i en e_j, correspond à trouver un chemin polygonal dans C_Ω du simplexe S_i associé à e_i au simplexe S_j associé à e_j. C'est cette déformation qui constitue la première étape de la résolution du test par analogie ; elle est schématisée par la figure VI.18.

Ce chemin peut ne pas exister, s'il n'existe pas de moyen de passer d'un sommet de S_A à un sommet de S_B en passant par des arêtes de C_Ω. Pour déterminer cela, il faut étudier les q-connexions des complexes de A et de B (voir chapitre V).

Intéressons-nous à la signification intuitive d'un chemin polygonal dans C_Ω. Pour cela, considérons de nouveau l'exemple de la figure VI.18. Soient deux simplexes successifs S_i et S_{i+1}, sur le chemin polygonal qui mène de S_A à S_B. S_i et S_{i+1} partagent un simplexe, noté σ. Le simplexe σ représente ce qui est resté *invariant* quand S_i se transforme en S_{i+1} le long de cette étape du chemin polygonal. Ainsi, plus la dimension de σ est grande, et plus la similitude (le nombre de propriétés communes) entre S_i et S_{i+1} est grande.

Si S_A et S_B ne sont pas directement connectés par un simplexe comme σ, mais qu'ils sont q-connectés par un chemin de longueur l, alors il y a l étapes de transformations élémentaires, chacune d'elles laissant au moins q propriétés invariantes. Évidemment, plus le chemin est long, plus le simplexe de départ et d'arrivée peuvent être différents, même si la dimension du chemin est grande.

Nous privilégions donc, dans le système ESQIMO, les transformations de figures obtenues en utilisant les chemins polygonaux les plus courts et de plus grande dimension possible. Nous verrons que cela conduit soit à choisir les plus courts parmi ceux de plus grande dimension soit à l'inverse.

VI.3.2.2 Appliquer une transformation à un simplexe quelconque

Appliquons maintenant la transformation ci-dessus qui fait passer d'un simplexe e_i à e_j, à un autre simplexe, disons e_k. Ainsi, le simplexe e_k va être déformé selon le chemin polygonal déterminé entre e_i et e_j. Nous verrons comment généraliser à la déformation d'un complexe K quelconque ensuite.

En effet, il s'agit de trouver les propriétés de la figure C que l'on souhaite transformer selon R_{AB}. Comme nous l'avons vu lors de l'étude expérimentale, certains sujets choisissent de changer tout ce qui est possible en C, d'autres gardent le maximum de caractéristiques de C. Ainsi le domaine de C à transformer varie, et produit des solutions différentes.

Le problème est donc d'étendre une fonction f qui transforme un simplexe donné en un autre simplexe donné, en une fonction $\overline{f}$, de domaine de définition K tout entier.

Nous voulons que $\overline{f}$ soit une *application simpliciale*, c'est-à-dire que si $\sigma_1 \subset \sigma_2$, alors $\overline{f(\sigma_1)} \subset \overline{f(\sigma_2)}$. Cette condition indique simplement que $\overline{f}$ respecte la *structure* du complexe.

Il y a plusieurs prolongements de f en $\overline{f}$ possibles, chacun correspondant à un raisonnement différent. ESQIMO envisage 4 prolongements différents par le biais de 4 heuristiques détaillées plus bas.

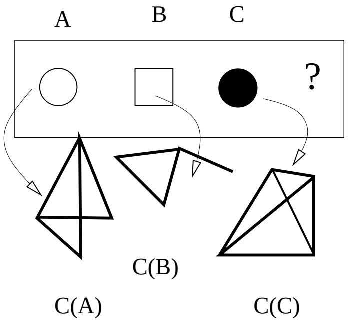
Nous avons posé les mécanismes de base permettant de résoudre notre problème d'analogie de manière *diagrammatique*, dans le cas où les figures A et B sont réduites à des simplexes. Ce mécanisme élémentaire de résolution est illustré par la figure dans le cas où C est aussi réduit à un simplexe.

VI.3.3 Procédure générale de résolution d'analogies

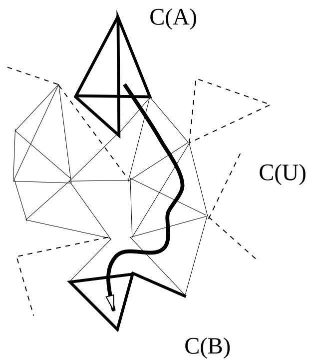
VI.3.3.1 Figures générales A et B non élémentaires

Les figures A, B, et C peuvent être composées d'un nombre $n \geq 1$ d'éléments e_i, elles sont alors représentées par des *complexes* simpliciaux, C_A, C_B, et C_C. Ces complexes sont construits à

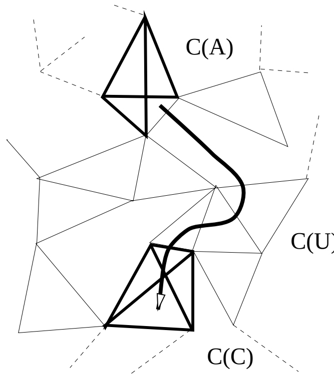
FIG. VI.19 – Les quatre étapes de la modélisation du raisonnement diagrammatique pour la résolution des tests de Q.I, dans le cas où A, B et C sont représentées par des simplexes.



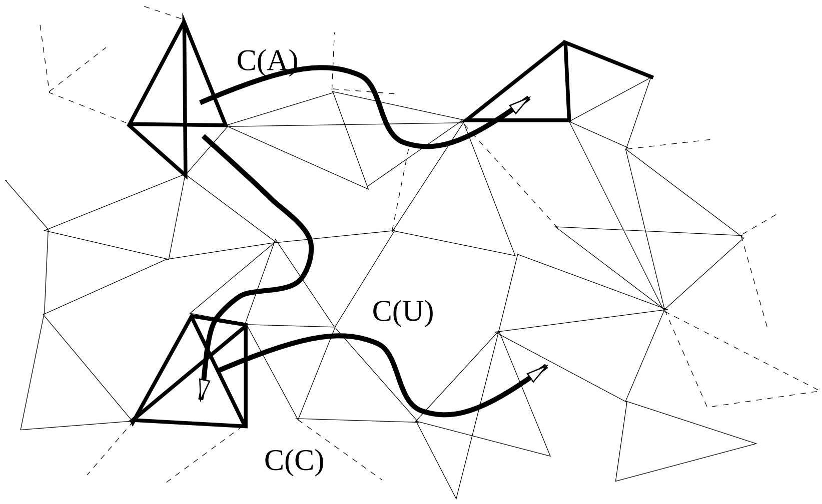
(a) Représentation simpliciale des figures du test de QI.



(b) Chemin polygonal entre S(A) et A(B) dans C(Omega).



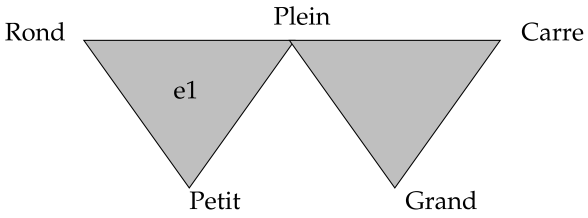
(c) Recherche du domaine de S(C) à transformer.



(d) Application de la transformation simpliciale au domaine de C.

FIG. VI.20 – Un exemple de figure composée et sa représentation sous forme de complexe avec la relation de voisinage de base.

(a) Figure composée à représenter.



(b) Représentation simpliciale de la figure composée.

partir des simplexes représentant leurs composants, comme le montre la figure VI.20. Rappelons que, comme nous l'avons déjà mentionné dans ce chapitre, nous ne nous intéressons pas à *l'orientation* de la figure, ni à la *distance* ou aux positions *relatives* de ses éléments.

Nous pouvons alors nous ramener au cas des figures simples (section précédente), en considérant des *appariements* des simplexes de C_A et de C_B, et l'application de la transformation correspondante, aux simplexes de C. Examinons à présent ces appariements de simplexes.

VI.3.3.2 Appariements des sommets de C_A et C_B

Cet appariement peut être effectué selon plusieurs méthodes. Soit $Som(A)$, l'ensemble des sommets de C_A, et $Som(B)$ l'ensemble des sommets de C_B.

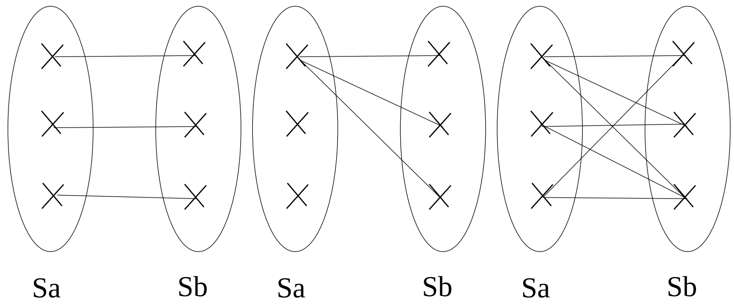
Si $Card(Som(A)) = Card(Som(B))$, les appariements possibles sont tous ceux qui permettent un recouvrement total de $Som(B)$. Par exemple, nous pouvons tout simplement apparier un sommet de C_A avec un et un seul sommet de C_B. Toutes les permutations de cet appariement seront aussi acceptables. Mais nous pouvons aussi projeter toute partie de C_A sur un sommet de C_B, du moment que tous les sommets de $Som(B)$ sont appariés. Des exemples d'appariements autorisés quand $Card(Som(A)) = Card(Som(B))$sont donnés par la figure VI.21.

Si $Card(Som(A)) > Card(Som(B))$, il faut que certains sommets de C_A disparaissent ou bien que plusieurs sommets de C_A se transforment en un seul sommet de C_B. Des exemples d'appariements autorisés lorsque $Card(Som(A)) > Card(Som(B))$ sont donnés en figure VI.22.

Enfin, si $Card(Som(A)) < Card(Som(B))$, il faut qu'au moins un sommet de C_A se transforme en plusieurs sommets de C_B, ou bien qu'il y ait *création* de sommets, ou que plusieurs sommets de C_A se transforment en encore plus de sommets dans C_B. Toutes ces heuristiques d'appariement ont été envisagées et programmées. Des exemples d'appariements autorisés lorsque $Card(Som(A)) < Card(Som(B))$ sont donnés en figure VI.23.

Le fait d'envisager des processus d'appariements où un sommet de C_A peut être apparié à plusieurs sommets de C_B, correspond à une déformation simultanée de ce sommet en deux sommets distincts de S_B. La déformation de C_A vers C_B peut se faire selon plusieurs chemins polygonaux plus ou moins longs de C_Ω.

FIG. VI.21 – Appariements autorisés lorsque $Card(Som(A)) = Card(Som(B))$.

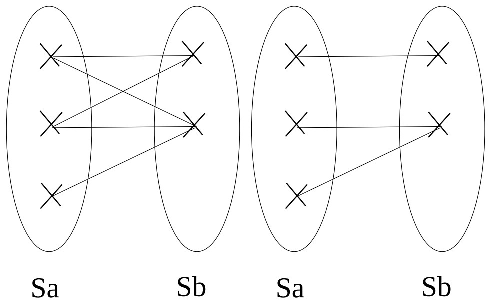


(a) Appariement 1-1.

(b) Certains sommets ne sont pas appariés, ils disparaissent au cours de la transformation.

(c) Certains sommets sont appariés au meme sommet à l'arrivée, ils se confondent au cours de la transformation.

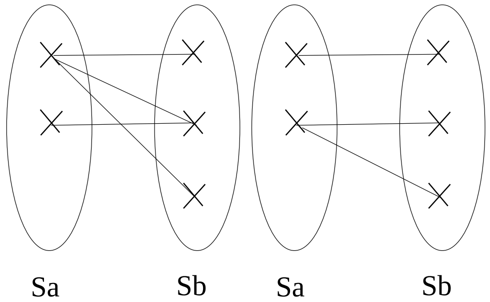
FIG. VI.22 – Appariements autorisés lorsque $Card(Som(A)) > Card(Som(B))$.



(a) Certains sommets sont appariés au meme sommet à l'arrivée.

(b) Tous les sommets en trop sont appariés au dernier sommet à l'arrivée.

FIG. VI.23 – Appariements autorisés lorsque $Card(Som(A)) < Card(Som(B))$.



(a) Certains sommets sont appariés au meme sommet.

(b) Le dernier sommet est apparié avec tous les sommets non encore appariés à l'arrivée.

Pour chaque appariement établi, nous déterminons une déformation élémentaire associée pour passer de chacun des éléments de la paire appariée à l'autre. Ce sont ces déformations élémentaires que nous appliquerons ensuite en parallèle sur un domaine de S_C qu'il reste encore à déterminer. Les déformations élémentaires trouvent une réalisation soit successive soit simultanée selon la décomposition de la transformation considérée.

VI.3.3.3 Application de la transformation à C_C, et construction de la solution C_D

C'est par le même processus d'appariements que nous nous proposons de trouver le domaine de S_C, sur lequel appliquer la transformation trouvée pour passer de C_A à C_B.

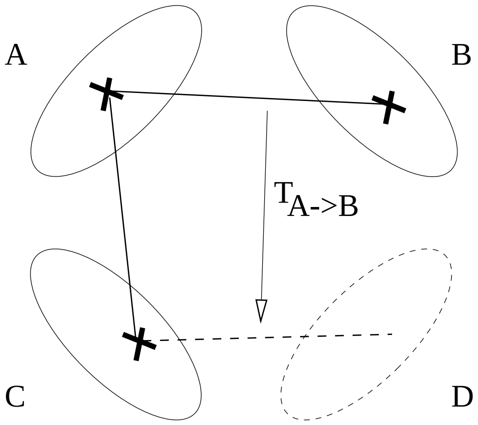
Pour appliquer la transformation R_{AB} sur le domaine de C_C que nous venons de déterminer, il suffit de *déformer* le domaine de C_C, le long de la déformation de C_A vers C_B, notée $C_A \mapsto C_B$. Les déformations peuvent être nombreuses, les solutions seront alors nombreuses. Plusieurs déformations peuvent aboutir à une même solution, cela correspond à des justifications différentes données par des sujets à une même réponse.

Un appariement des sommets des représentations de A et de B doit donc être mis en correspondance avec un appariement des sommets de C_A avec les sommets de C_B. Dans le cas où C_A, C_B et C_C contiennent un seul simplexe, disons S_A, S_B et S_C respectivement, il suffit d'appliquer à S_C, la transformation $S_A \mapsto S_C(S_C)$, comme le montre la figure VI.24.

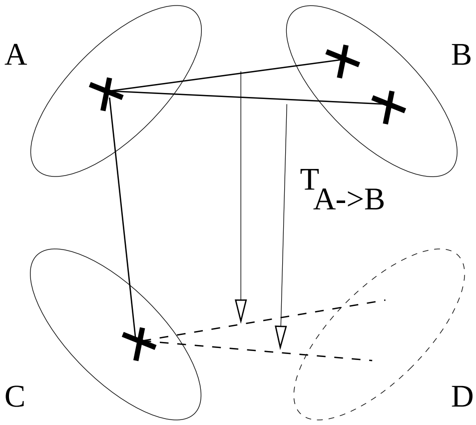
Dans le cas où C_B contient deux simplexes, S_B et S'_B, on dispose de deux transformations $S_A \mapsto S_B$ et $S_A \mapsto S'_B$ que l'on applique alors en *parallèle* sur les simplexes de C_C (voir figure VI.24 b).

Dans le cas où C_C contient deux simplexes S_C et S'_C, il y a plusieurs appariements envisageables : si S_A est apparié à S_C *et* S_C' , alors $S_A \mapsto S_B$ est appliqué en *parallèle* à S_C et S'_C (voir figure VI.24 c).

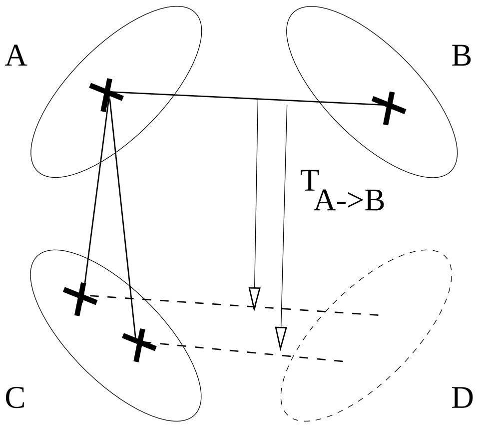
FIG. VI.24 – Application de R_{AB} sur les simplexes de C_C, et construction de C_D. Les croix symbolisent les simplexes contenus dans la représentation des figures, les ensembles symbolisent les complexes associés aux figures A, B, C et D. Les flèches en pointillés correspondent à l'application de R_{AB} sur les composantes de C_C. Dans les cas et l'application des différents appariements mène à de nombreuses solutions possibles pour C_D.



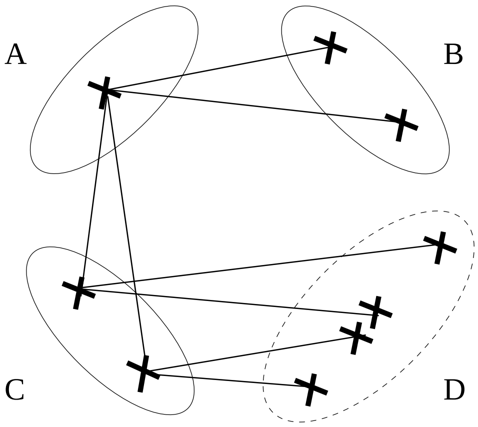
(a) Les figures sont composées d'un seul simplexe.



(b) Le figure B se représente en deux simplexes.



(c) La figure C se représente avec deux simplexes.



(d) Les figures B et C se représentent avec deux simplexes.

Enfin, dans le cas général, notons S_B^i, les simplexes de C_B, liés au simplexe S_A^i de C_A, et S_C^i, les simplexes de C_C liés à S_i. Alors, les transformations $S_i \mapsto S_B^i$ sont à appliquer en *parallèle* aux simplexes S_C^i (voir figure VI.24 d).

VI.3.4 Algorithme de ESQIMO et remarques sur l'implémentation

ESQIMO a été prototypé en Mathematica [104], puis implémenté en Caml. Le logiciel est lui-même disponible sur le Web (site du LIMSI[2]), et un manuel d'installation et d'utilisation est également disponible comme rapport interne [86][3].

Nous donnons, pour commencer, l'algorithme général d'ESQIMO, noté App_{ESQ}, qui résume toutes les procédures de résolution. Cet algorithme utilise des noms de fonctions pour chaque procédure de résolution qui sont expliqués ensuite, dans le coprs de cette section. Enfin, les étapes de l'application de cet algorithme à un exemple de figure A, B et C sont illustrées par la figure VI.19.

```
App_ESQ :
C_A, C_B et C_C = représentations des figures A, B, et C.
AssocSet (A-B)= appariements entre C_A et C_B
   AppApp(A-B)
      heuristique = App1 ou App2
   Pour les paires (A-B) appariées :
      ChemSet = Chemins entre les simplexes appariés de AssocSet
         C_courant = C_A
         Tant que C_B pas atteint
         Qconnexion C_courant
         PartageDeLignes
      Piste = ChemSet → étapes élémentaires
      PistMin = recherche d'une suite minimale dans Piste
      DéfElem = Pistes en déformations élémentaires
AssocSet (A-C)= appariements entre C_A et C_C
   AppApp(A-C)
      heuristique = App1 ou App2
   Pour les paires (A-B) appariées :
   Domaine = domaine de C_C d'après AssocSet (A-C)
Simultanément = application parallèle de
   Successivement = application successive des
      DéfElem au Domaine de C_C.
Choisir = ensemble des solutions obtenues.
```

[2] voir http ://www.limsi.fr/Individu/erika/

[3] En ligne à l'adresse http ://www.limsi.fr/Individu/erika/esqimo.html

VI.3.4.1 Représentations

Nous avons utilisé une représentation alternative des complexes abstraits sous forme d'ensembles de lignes de la matrice d'incidence de C_Ω. Un simplexe est donc un *vecteur*, et un complexe, un *tableau*. Cette représentation est possible, puisque nous travaillons toujours dans C_Ω, qui est donné a *priori*. Elle n'est pas utilisable quand les sommets de l'univers ne sont pas connus, ou s'ils peuvent évoluer dynamiquement. Cette implémentation des complexes a l'avantage d'être bien adaptée aux traitements dans ESQIMO. Notons que de nouveaux univers de figures élémentaires sont définissables sur des fichiers séparés, mais ils ne peuvent plus évoluer au cours de la résolution d'une analogie (pas d'apprentissage).

Pour représenter un élément par des simplexes (voir figure VI.19 a)), il suffit d'extraire la ligne de la matrice d'incidence pour l'élément, comme nous l'avons montré précédemment. Afin de déterminer si deux simplexes sont connectés, nous écrivons une fonction `Qconnexion` qui calcule la q-connexion. Comme les simplexes sont représentés par leur ligne dans la matrice d'incidence, c'est la fonction `PartageDeLignes`, qui vérifie effectivement si deux simplexes sont connectés.

VI.3.4.2 Appariements

La première étape de l'algorithme présenté est la recherche d'appariements, nous avons programmé deux heuristiques, nommées `App1` et `App2`. La fonction `Appariements` appelle l'une de ces deux heuristiques (avec la deuxième qui est appliquée par défaut).

La première fonction, `App1`, correspond à l'implémentation de la première heuristique. Tous les appariements de n'importe quelle partie de l'ensemble S_A avec chaque sommet de S_B sont calculés, et la fonction renvoie un ensemble d'appariements désigné par la tête `AssocSet`.

La deuxième fonction, `App2`, correspond à l'implémentation de la deuxième heuristique. Ici, on procède à un appariement $1-1$, et dans *l'ordre* de classement des sommets de S_A et de S_B. S'il reste des sommets de S_A non appariés, ils disparaissent, ce qui revient à les apparier à l'ensemble $\emptyset$.

VI.3.4.3 Les chemins

Les appariements se traduisent en termes de chemins (voir figure VI.19 b))à parcourir pour aller d'un sommet de l'appariement au sommet d'arrivée de l'appariement. La fonction `Chemins` détermine les chemins existant entre deux simplexes. Les simplexes sont toujours exprimés en termes de lignes de la matrice d'incidence des relations de l'univers.

Pour deux sommets appariés, il existe beaucoup de chemins possibles, l'ensemble des chemins renvoyé commence par la tête `ChemSet`. Chaque `ChemSet` est composé d'une `Piste` à suivre pour déformer un sommet en l'autre. Une `Piste` est un fragment élémentaire de chemin, elle correspond donc à un déplacement élémentaire le long d'un chemin.

VI.3.4.4 Les déformations élémentaires

Chaque `Piste`, peut se traduire en termes de déformations élémentaires. En effet, lorsque l'on chemine le long d'un `Piste`, cela revient à changer l'une des propriétés du simplexe qui se déforme. Chaque `Piste` correspond donc à une `DefElem`. Il y a plusieurs heuristiques pour déterminer `DefElem`, c'est pourquoi nous en avons implémenté cinq, nommées `DefElem1` à `DefElem5`. La fonction `CurrentDefElem`, appelle l'une des heuristiques, nous choisissons `DefElem1` par défaut. `DefElemm[a, b][s]`, correspond à l'application de la déformation de `a` en `b`, à `s`.

Les `DefElem` sont groupées pour constituer la déformation *globale* à appliquer sur le domaine de C_C. Certaines `DefElem` seront appliquées simultanément, lorsqu'elles concernent des sommets différents qui se transforment en *parallèle*. D'autres déformations seront appliquées successivement lorsqu'elles correspondent à des déformations le long d'une même piste pour un sommet donné. Ainsi, nous introduisons les têtes `Successivement` et `Simultanement`. De plus, on choisira en priorité d'appliquer la déformation qui correspond à la piste de longueur minimale, calculée à l'aide de la fonction `PisteMin`.

VI.3.4.5 Résolution générale

Nous définissons une fonction `AppApp` qui se charge de sélectionner un appariement des sommets de S_A et S_B et un appariement de S_A avec S_C (voir figure VI.19 c)), pour proposer une *résolution* (figure VI.19 d)). Nous avons deux heuristiques correspondant aux deux fonctions `AppApp1` et `AppApp2`.

À partir de cette mise en correspondance, la fonction `Resolve` effectue dans l'ordre les quatre étapes de la résolution, par les appels aux fonctions définies précédemment, comme montré par l'algorithme que nous avons donné en tête de cette section.

VI.4 Exemples commentés

Regardons maintenant les solutions que ESQIMO propose pour 8 tests de QI simples. Chaque exemple est numéroté, nous en avons fourni l'entrée et la sortie affichée par la système et une illustration géométrique à chaque fois. ESQIMO peut donner la trace de son raisonnement, ce qui constitue une justification de l'analogie. En particulier, il fournit des informations sur :

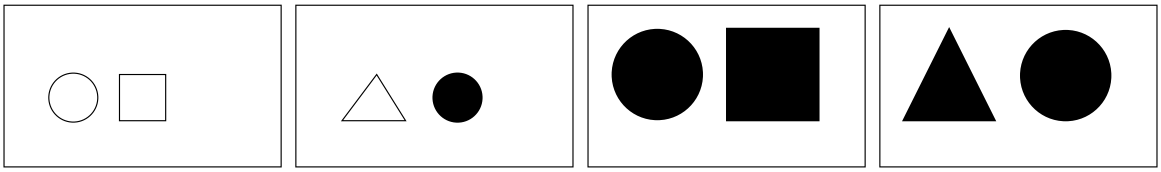
- Les appariements `AssocSet` choisis ;
- Les transformations élémentaires `D-elem` qu'il applique en parallèle (`Simultanement`) ou en série (`Successivement`) ;
- Le domaine `Domaine` de C_C sur lequel il l'applique ;
- L'ensemble `Choisir` des solutions.

VI.4.0.6 Exemple 1

Voici le test de QI posé à ESQIMO :

```
A1={LigneDe[e1],LigneDe[e2]};
B1={LigneDe[e3],LigneDe[e4]};
C1={LigneDe[e10],LigneDe[e11]};
Resolve[A1,B1,C1,App2,AppApp2]
```

FIG. VI.25 – Exemple 1 de résolution de test de Q.I. avec ESQIMO.



Voici la réponse fournie par ESQIMO au test 1 (figure VI.25) :

```
Choisir[AssocSet[FromTo[1,{1}],FromTo[2,{2}]],
        AssocSet[FromTo[1,{1}],FromTo[2,{2}]]]
Simultanement[
   Domaine[1,
   Successivement[
      "D-elem"[RondQ->0,TriangleQ->1]],{e10}],
      Domaine[2,
   Successivement["D-elem"[CreuxQ->0,PleinQ->1],
      "D-elem"[CarreQ->0,RondQ->1]],{e11}]]}
Choisir[{e12,e10}]
```

Justification : La transformation consiste à transformer le rond de gauche en triangle et le carré de droite en rond noir. Ici, la couleur du triangle obtenu à droite est noire car les éléments de départ étaient déjà noirs, contrairement à la figure A.

VI.4.0.7 Exemple 2

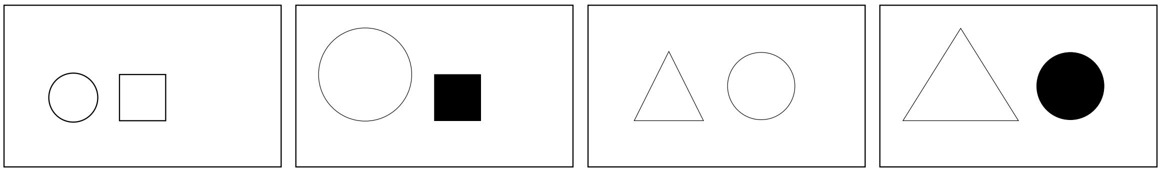
Voici le test de QI posé à ESQIMO :

```
A1={LigneDe[e1],LigneDe[e2]};
B1={LigneDe[e7],LigneDe[e5]};
C1={LigneDe[e3],LigneDe[e1]};
Resolve[A1,B1,C1,App2,AppApp2]
```

Voici la réponse fournie par ESQIMO au test 2 (figure VI.26) :

```
Choisir[AssocSet[FromTo[1,{1}],FromTo[2,{2}]],
        AssocSet[FromTo[1,{1}],FromTo[2,{2}]]]}
Simultanement[
   Domaine[1,Successivement
                "D-elem"[PetitQ->0,GrandQ->1]],{e3}],
   Domaine[2,Successivement[
                "D-elem"[CreuxQ->0,PleinQ->1]],{e1}]]}
Choisir[{e9,e4}]
```

FIG. VI.26 – Exemple 2 de résolution de test de Q.I. avec ESQIMO.



Justification : La transformation proposée correspond à agrandir la première figure, colorier la deuxième en noir. Ici, la figure agrandie est celle de gauche, sa triangularité n'est pas altérée.

VI.4.0.8 Exemple 3

Voici le test de QI posé à ESQIMO :

```
A1={LigneDe[e1],LigneDe[e5]};
B1={LigneDe[e4],LigneDe[e2],LigneDe[e8]};
C1={LigneDe[e3],LigneDe[e4]};
Resolve[A1,B1,C1,App2,AppApp2]
```

Voici la réponse fournie par ESQIMO au test 3 (figure VI.27) :

```
Choisir[AssocSet[FromTo[1,{1}],FromTo[2,{2,3}]],
        AssocSet[FromTo[1,{1}],FromTo[2,{2}]]]}
Simultanement[
   Domaine[1,Successivement[
      "D-elem"[CreuxQ->0,PleinQ->1]],{e3}],
   Domaine[2,
      Duplique[
         Successivement[
            "D-elem"[PleinQ->0,CreuxQ->1]],
                 Successivement[
                    "D-elem"[PetitQ->0,GrandQ->1],
                    "D-elem"[PleinQ->0,CreuxQ->1]]],{e4}]]}
Choisir[{e6,e1,e7}]
```

Justification : La transformation proposée correspond à noircir la première composante, éclaircir la deuxième, et la dupliquer en plus grand. Encore une fois, les propriétés de rondeur, carréïté ou triangularité ne sont pas altérées.

FIG. VI.27 – Exemple 3 de résolution de test de Q.I. avec ESQIMO.

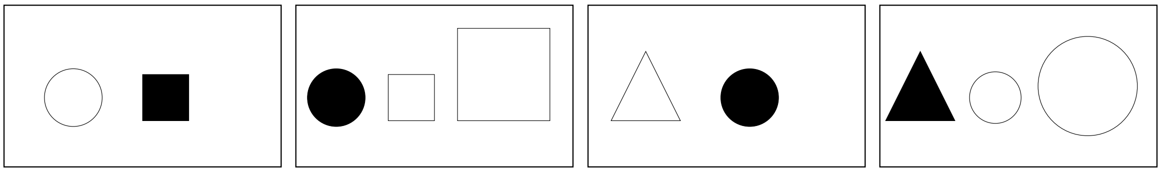
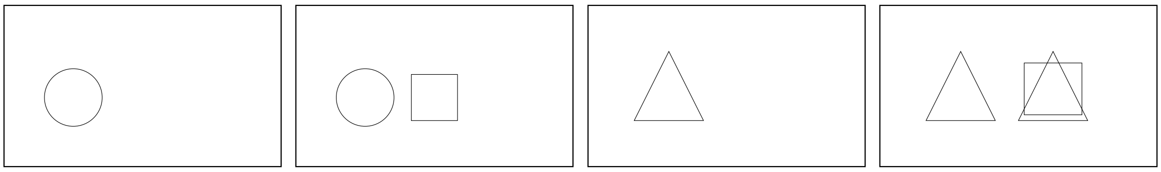


FIG. VI.28 – Exemple 4 de résolution de test de Q.I. avec ESQIMO.



VI.4.0.9 Exemple 4

Voici le test de QI posé à ESQIMO :

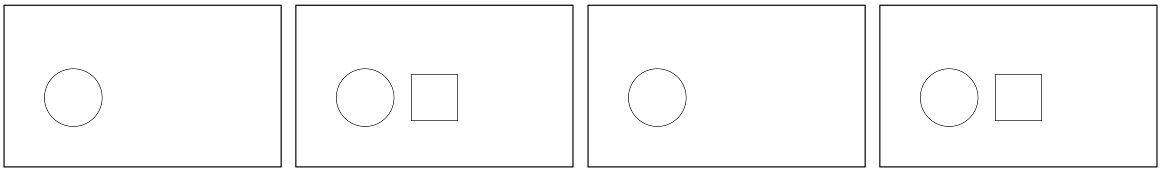
```
A1={LigneDe[e1]};
B1={LigneDe[e1],LigneDe[e2]};
C1={LigneDe[e3]};
Resolve[A1,B1,C1,App2,AppApp2]
```

Voici la réponse fournie par ESQIMO au test 4 (figure VI.28) :

```
Choisir[AssocSet[FromTo[1,{1,2}]],
        AssocSet[FromTo[1,{1}]]]}
Simultanement[
   Domaine[1,
      Duplique[
         Successivement[
            "D-elem"[]],
            Successivement[
               "D-elem"[RondQ->0,CarreQ->1]]],{e3}]]}
Choisir[{e3,"Unknown"[False,True,True,True,False,True,False]}]
```

Justification : La transformation proposée correspond à dupliquer la figure en la rendant carrée. La triangularité n'étant pas enlevée pour autant, il y a donc production d'une solution non-stable. En effet, ici la fonction d'application de la déformation ne fait pas passer la propriété triangulaire à 0.

FIG. VI.29 – Exemple 5 de résolution de test de Q.I. avec ESQIMO.



VI.4.0.10 Exemple 5

Voici le test de QI posé à ESQIMO :

```
A1={LigneDe[e1]};
B1={LigneDe[e1],LigneDe[e2]};
C1={LigneDe[e1]};
Resolve[A1,B1,C1,App2,AppApp2]
```

Voici la réponse fournie par ESQIMO au test 5 (figure VI.29) :

```
Choisir[AssocSet[FromTo[1,{1,2}]],AssocSet[FromTo[1,{1}]]]}
   Simultanement[
      Domaine[1,
         Duplique[
            Successivement[
               "D-elem"[]],
               Successivement[
                   "D-elem"[RondQ->0,CarreQ->1]]],{{e1}}]]}
Choisir[{e1,e2}]
```

Justification : La transformation proposée correspond à dupliquer la figure, et rendre la deuxième carrée en lui enlevant sa caractéristique ronde. Ici, la figure C proposée est exactement égale à A, D est alors exactement égale à B.

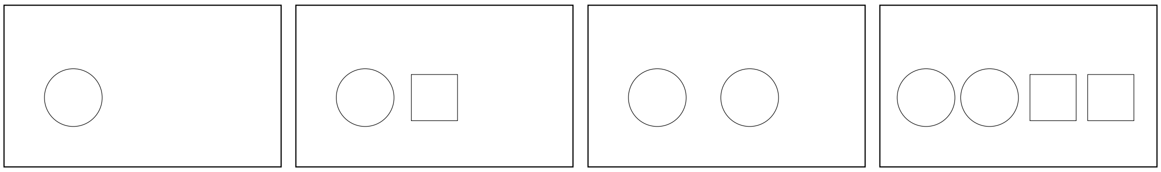
VI.4.0.11 Exemple 6

Voici le test de QI posé à ESQIMO :

```
A1={LigneDe[e1]};
B1={LigneDe[e1],LigneDe[e2]};
C1={LigneDe[e1],LigneDe[e1]};
Resolve[A1,B1,C1,App1,AppApp2]
```

Voici la réponse fournie par ESQIMO au test 6 (figure VI.30) :

FIG. VI.30 – Exemple 6 de résolution de test de Q.I. avec ESQIMO.



```
Choisir[AssocSet[FromTo[1,{1,2}]],
        AssocSet[FromTo[1,{1,2}]]]}
Simultanement[
   Domaine[1,
      Duplique[
         Successivement[
            "D-elem"[]],
            Successivement[
               "D-elem"[RondQ->0,CarreQ->1]]],{e1,e1}]]}
Choisir[{e1,e1,e2,e2}]
```

Justification : La transformation proposée correspond à dupliquer chaque figure en : elle-même et une deuxième carrée et moins ronde. Comme la figure de départ contient deux éléments, la solution contient quatre éléments correspondant à la duplication et transformation de chacun d'entre eux.

VI.4.0.12 Exemple 7

Voici le test de QI posé à ESQIMO :

```
A1={LigneDe[e1]};
B1={LigneDe[e1],LigneDe[e2]};
C1={LigneDe[e4]};
Resolve[A1,B1,C1,App1,AppApp2];
```

Voici la réponse fournie par ESQIMO au test 7 (figure VI.31) :

```
Choisir[AssocSet[FromTo[1,{1,2}]],
        AssocSet[FromTo[1,{1}]]]}
Simultanement[
   Domaine[1,
      Duplique[Successivement[
         "D-elem"[]],
         Successivement[
           "D-elem"[RondQ->0,CarreQ->1]]],{e4}]]}
Paire[1,{e4,e5}]
```

FIG. VI.31 – Exemple 7 de résolution de test de Q.I. avec ESQIMO.

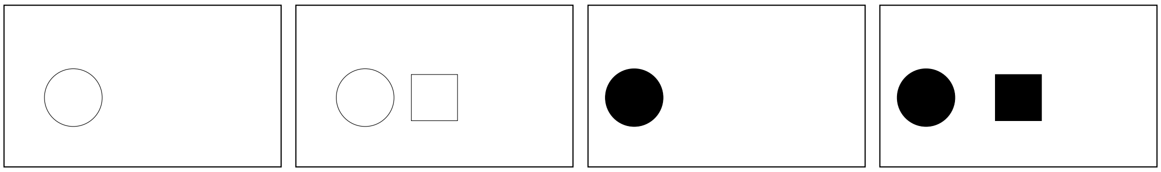
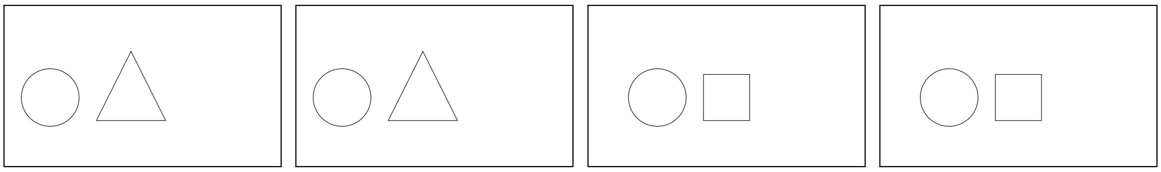


FIG. VI.32 – Exemple 8 de résolution de test de Q.I. avec ESQIMO.



Justification : La transformation proposée correspond à la même que précédemment, mais appliquée sur un élément effectivement rond, ce qui donne bien un carré, stable. La couleur noire n'est pas enlevée.

VI.4.0.13 Exemple 8

Voici le test de QI posé à ESQIMO :

```
A1={LigneDe[e1],LigneDe[e9]};
B1={LigneDe[e1],LigneDe[e9]};
C1={LigneDe[e1],LigneDe[e2]};
Resolve[A1,B1,C1,App2,AppApp2];
```

Voici la réponse fournie par ESQIMO au test 8 (figure VI.32) :

```
Choisir[AssocSet[FromTo[1,{1}],FromTo[2,{2}]],
        AssocSet[FromTo[1,{1}],FromTo[2,{2}]]]}
Simultanement[Domaine[1,
    Successivement["D-elem"[]],{e1}],
        Domaine[2,Successivement[
           "D-elem"[]],{e2}]]}
Paire[1,{e1,e2}]
```

Justification : La transformation proposée correspond à l'identité.

VI.5 Discussion de l'algorithme ESQIMO

Dans cette section, nous allons discuter de l'approche ESQIMO pour résoudre un problème d'analogie. Nous avons fait apparaître dans la présentation de l'algorithme un certain nombre de choix, qui peuvent sembler arbitraires, par exemple dans la politique d'appariement. En fait, ces choix constituent autant de variantes possibles d'une même famille d'algorithmes. Nous n'avons pas eu le temps d'explorer les diverses instances de cette famille, ni de comparer leurs résultats, mais nous allons discuter les différents paramètres de cette famille :

1. la description d'une configuration,
2. le choix de l'unité de transformation,
3. la détermination du domaine d'une transformation,
4. le choix d'un chemin,
5. la transformation associée à un chemin,
6. la formalisation de la similitude des chemins.

Chacun de ces points constitue autant de possibilités de paramétrer l'approche ESQIMO. Nous allons les détailler ci-dessous.

VI.5.1 Description d'une configuration

La description d'une configuration est faite à l'aide d'un ensemble de prédicats, suivant le cadre présenté au chapitre V. Cet ensemble de prédicats est arbitraire, et nous pouvons donc, sans rien toucher au reste de l'algorithme, compléter les descriptions ou bien en choisir d'entièrement différentes.

Cependant, décrire une configuration uniquement en termes de prédicats sur les éléments de la configuration peut être contraignant. Prenons l'exemple, dans le domaine des figures géométriques, de la *position* d'un élément de la figure. A *priori*, la position n'est pas un prédicat puisqu'elle associe des coordonnées (x, y) à chaque élément de la figure. Deux solutions sont alors envisageables.

– On peut toujours trouver un codage de ce type de propriété en termes de prédicats. Par exemple, pour la position, on peut envisager autant de prédicats que d'emplacements pour un élément du graphique. Si notre figure est décrite sur un damier 4×4, il y aura 16 prédicats.
– On peut considérer un autre codage qui utilise un complexe simplicial. Par exemple, pour la position, si ce que l'on veut prendre en compte c'est en fait le voisinage d'un élément (un carré est à coté du rond), on peut coder la position comme une relation binaire (cf. chapitre V) e_i `VoisinsDe` e_j . Dans ce cas, le complexe codant la nouvelle propriété ne peut pas être intégré dans le complexe C_Ω. Il faut donc considérer plusieurs complexes différents pour représenter une configuration : C_Ω C'_Ω C''_Ω. Cependant, sur chaque complexe, le travail à effectuer est similaire à celui à réaliser sur C_Ω.

La deuxième approche généralise simplement ESQIMO en envisageant une résolution en parallèle sur plusieurs complexes. Le cas idéal correspond à celui où ces complexes sont sans interaction (par exemple, n'importe quelle figure peut se trouver à n'importe quelle position). Si les propriétés envisagées interagissent, alors il faut qu'elles soient codées dans le même complexe.

VI.5.2 Le choix de l'unité de transformation

Dans nos exemples, nous avons considéré que l'analogie entre deux configurations dépendait uniquement des rapports d'analogie entre les éléments de la configuration. Un élément de la configuration étant représenté par un simplexe dans un complexe, cela se traduit par la recherche d'un chemin entre deux simplexes, ce chemin représentant la transformation des éléments de départ. L'analogie se représente donc par un ensemble de chemins entre simplexes.

La situation peut se généraliser sans problème en considérant des chemins entre complexes. Cette généralisation s'interprète naturellement de la manière suivante : on ne représente plus cette fois la transformation d'un élément en un autre élément, mais plus généralement, la transformation d'un groupe d'éléments en un autre groupe d'éléments. On peut ainsi exprimer la transformation de la paire (carré, rond) en triangle alors que carré ne se transforme pas en triangle, pas plus que le rond. Le complexe considéré ne doit pas correspondre nécessairement à une union d'éléments. Par exemple ,le complexe (carré, rondeur) représente un carré en présence d'un élément rond quelconque.

Le problème à résoudre concerne alors le choix des complexes considérés. Ce problème ne fait que généraliser le problème du choix des simplexes à apparier. Nous nous restreignons dans la suite aux problèmes des simplexes à apparier, nos commentaires étant valables aussi si nous généralisons la situation aux complexes.

VI.5.3 Détermination du domaine d'une transformation

Ce problème est celui du choix des simplexes qui vont se transformer l'un en l'autre. Ce choix se pose en fait deux fois : la première fois pour apparier des éléments de A avec des élément de B et une seconde fois pour décider quels éléments de B sont appariés avec quels éléments de C.

Nous avons traité l'appariement de manière purement combinatoire en proposant deux fonctions d'appariement possibles (les fonctions `App1` et `App2`) et deux fonctions permettant de mettre en relation les appariements $S_A : S_B$ et $S_A : S_C$ (les fonctions `AppApp1` et `AppApp2`). Les fonctions `App1` et `App2` correspondent à deux stratégies différentes pour calculer $S_A : S_B$ ou bien $S_A : S_C$. Les fonctions `AppApp1` et `AppApp2` représentent deux stratégies différentes pour combiner les appariements.

Nous avons qualifié notre traitement de l'appariement de *combinatoire*. En effet, les fonctions `App1`, `App2`, `AppApp1` et `AppApp2` qui permettent d'implémenter cet appariement ne tiennent aucun compte de la nature topologique des éléments à apparier. Évidement, on peut en tenir compte, par exemple pour apparier de manière privilégiée les simplexes de même dimension, ou bien qui ont une connectivité (i.e. une ressemblance) maximale, etc. De nombreuses stratégies diagrammatiques [90] sont possibles pour raffiner et guider le choix combinatoire.

VI.5.4 Le choix d'un chemin

Soit les deux simplexes S_i et S_j correspondant à deux éléments e_i et e_j. Les chemins d'extrémités S_i et S_j correspondent aux transformations que nous envisageons entre e_i et e_j . Il peut exister 0, 1 ou plusieurs chemins entre S_i et S_j. Le cas où il existe un seul chemin ne pose pas de problème.

Dans le cas où il n'y a pas de chemins entre S_i et S_j, nous considérons que la transformation de e_i en e_j est impensable. Il faudrait donc remettre en cause l'appariement qui a associé S_i à S_j. Nous ne traitons pas ce cas dans l'algorithme décrit.

Quand il y a plusieurs chemins entre S_i et S_j, nous avons arbitrairement choisi parmi les chemins de plus grande dimension, un des chemins les plus courts. Ce n'est pas la seule stratégie possible. On peut par exemple conserver tous les chemins possibles pour étudier combinatoirement les différentes solutions qui peuvent être générées, on peut choisir un chemin parmi les plus courts indépendamment de la dimension du chemin, on peut choisir un chemin en tenant compte des autres chemins à choisir, etc.

VI.5.5 La transformation associée à un chemin

Nous associons une transformation à un chemin. Cette association est faite de la manière suivante : un chemin étant constitué de pas élémentaires (un pas étant deux simplexes successifs sur le chemin), la transformation est constituée de la succession des transformations élémentaires associées aux pistes élémentaires.

Il se pose donc la question d'associer une transformation élémentaire à une paire de simplexes (S_i, S_j). En fait, il existe de nombreuses fonctions f telles que $f(S_i) = S_j$. Nous avons proposé quatre choix possibles d'une telle fonction, ce sont les fonctions `DefElem1` à `DefElem4`. D'autres choix sont encore possibles. On peut même mettre en cause ce passage par des transformations élémentaires pour calculer la transformation associée à un chemin. Le tout est de fournir pour chaque chemin, une transformation.

VI.5.6 La formalisation de la similitude des chemins

Quand on a (enfin) obtenu un chemin, disons entre S_i et S_j, on veut considérer ce chemin comme une transformation que l'on va appliquer à S_k. On obtient en résultat un nouveau simplexe S_l. Si $S_l \in C_\Omega$, on dit que S_l est *stable* (c'est un élément de l'univers connu) et sinon on le qualifie de *monstre* (c'est un élément impossible). Selon l'objectif, on acceptera ou pas ces monstres.

Une transformation se traduit par un chemin. Il y a donc un chemin entre S_k et S_l qui est associé à la transformation. Notre idée véritable et diagrammatique est que nous pouvons nous passer de la notion de transformation pour nous intéresser directement à la similitude des deux chemins de S_i vers S_j et de S_k vers S_l.

En effet, l'idée de transformation composée de transformations élémentaires n'est finalement qu'un mécanisme permettant d'exhiber un chemin de S_k vers S_l présentant une similitude de degré 0 avec le chemin de S_i vers S_j . Cette similitude est la suivante :

– les deux chemins ont le même nombre de pas élémentaires,
– chaque pas $(S^p_{i,j}, S^{p+1}_{i,j})$ du chemin de S_i vers S_j est similaire au pas $(S^p_{k,l}, S^{p+1}_{k,l})$ du chemin de S_k vers S_l en ce sens qu'il existe une famille de fonctions f_p telles que $f_p(S^p_{i,j}) = S^{p+1}_{i,j}$ et $f_p(S^p_{k,l}) = S^{p+1}_{k,l}$.

Nous qualifions de degré 0 cette similitude, car elle préserve peu de propriétés topologiques entre les deux chemins. Par ailleurs, afin de rendre compte de la création possible de monstres, il faut voir que l'espace dans lequel le chemin de S_k vers S_l n'est pas le même que l'espace dans lequel chemine S_i vers S_j. En effet, le chemin de S_i vers S_j prend place dans C_Ω alors que celui de S_k vers S_l est envisagé dans le complexe complet de C_Ω (le complexe complet de C_Ω est le simplexe dont les sommets sont les éléments de Ω). Autrement dit, quand on envisage le chemin de S_k vers S_l on a perdu la structure topologique de C_Ω (cela peut avoir un intérêt quand on veut produire des monstres !).

Nous esquissons ici deux approches qui méritent d'être envisagées afin de proposer d'autres notions de similitude :

VI.5.6.0.a Homotopie de chemins. On va supposer que les deux chemins prennent place dans le même espace. Alors une notion de similitude naturelle serait que les chemins soient *homotopes*. Autrement dit, on peut déformer C_Ω dans le chemin de S_i vers S_j en chemin de S_k vers S_l. Pour déterminer S_l, il faut considérer que ce doit être une extrémité commune aux chemins de S_k vers S_l homotopes aux chemins de S_i vers S_j mais aussi aux chemins de S_j vers S_l homotopes aux chemins de S_i vers S_k (on retrouve ici l'idée de symétrie du problème \$A :B : :C :D\$ qui amène à regarder aussi le rapport $A : C$).

VI.5.6.0.b Cobordisme. On peut considérer les deux chemins comme une *chaîne*. On peut alors les voir comme les bords d'un complexe inclus dans C_Ω. On doit donc trouver un complexe dont le chemin de S_i vers $S_j + S_k$, vu comme une chaîne, fait partie du bord. Le reste du bord constitue une chaîne associée à un chemin de S_k vers S_l , privé de S_k. Il reste à trouver le bon S_l dans cette chaîne (on peut donc aussi faire intervenir $A : C$ comme évoqué ci-dessus).

Ces approches ne sont qu'esquissées mais nous aimerions dans la suite du travail pousser leur étude.

VI.6 Conclusions sur ESQIMO

Remarquons tout d'abord que l'approche de ESQIMO ne dépend pas de la nature géométrique du domaine d'application. Nous résolvons en fait un problème abstrait d'analogie, et nous aurions pu choisir de l'illustrer dans un domaine numérique par exemple. Si nous avons choisi de travailler sur des figures géométriques, c'est surtout qu'elles permettent de saisir plus rapidement une conjonction plus grande de propriétés (forme, taille, couleur, texture, position), et sont donc plus parlantes. Mais nous insistons sur le fait qu'à aucun moment nous ne nous appuyons sur la nature géométrique du domaine d'application pour produire l'analogie.

Par ailleurs, ESQIMO utilise un raisonnement diagrammatique [90]. En effet, les données de la tâche à effectuer sont traduites en un espace dont l'organisation reflète la structure du problème. La résolution du problème consiste à trouver un chemin entre deux entités spatiales et à produire un chemin similaire entre deux autres entités. Cette similarité de chemin constitue l'analogie formelle entre les paires de configurations.

Troisième partie

L'adaptation pour la gestion de l'hétérogénéité sémantique entre bases de connaissances en logique de descriptions.

Chapitre VII

Les Complexes Simpliciaux pour Représenter les Relations d'une Base en Logique de Descriptions

Motivations

Nous avons formulé le problème de l'hétérogénéité sémantique entre agents dialogiques en logique des descriptions dans le chapitre III, et nous proposons dans ce chapitre de le résoudre à l'aide d'algorithmes d'adaptation et d'appariement de structures présentés dans la partie 2. Nous avons proposé un modèle de représentation des connaissances naturel pour modéliser le raisonnement par analogies : il est basé sur les Complexes Simpliciaux (CS) et s'appelle CAT.

Afin d'appliquer ces algorithmes de résolution de la HS en utilisant la notion d'adaptation et celle d'appariement de structures, nous envisageons donc dans ce chapitre d'établir une *correspondance* entre les Kbases en LD qui modélisent nos AD, et le formalisme CAT qui permet l'expression de ces algorithmes.

Résumé du chapitre

Le but de ce chapitre est de comparer le formalisme CAT des Complexes Simpliciaux (CS) à la LD pour éventuellement les proposer comme formalisme de représentation des connaissances alternatif et/ou complémentaire afin d'aborder le problème de l'hétérogénéité sémantique entre agents dialogiques. Pour cela, nous utilisons le fait qu'une base LD peut être exprimée en logique des prédicats du Premier ordre , il nous reste alors à savoir si une base en LPO peut s'exprimer sous une forme simpliciale.

Dans ce chapitre, nous commençons par comparer le formalisme des Complexes Simpliciaux, CAT à la LPO. Nous pouvons procéder à cette comparaison selon plusieurs aspects :

- Expressivité en termes de formalisme de représentation des connaissances ;
- Complexité des algorithmes de manipulation des structures de représentation du formalisme ;
- Stockage informatique ;

– Naturalité du formalisme ;
– Facilité de formalisation et procédures pour cette formalisation des connaissances à modéliser.

Notons qu'une fois la correspondance entre LPO et CS établie, nous ne chercherons pas à calculer une correspondance entre la LD et les CS en *passant* par les LPO. Nous utilisons ce résultat seulement pour *confirmer* qu'il est *possible* d'exprimer les connaissances contenues dans une base LD à l'aide de CS. Nous nous attacherons à montrer comment on peut alors exprimer des Tbox et des Abox à l'aide des CS, mais *directement,* sans utiliser la LPO. Pour toutes ces raisons, nous allons maintenant comparer les formalismes LPO et CAT en termes *d'expressivité* seulement *[95].*

Nous présentons quelques rappels sur la LPO, en particulier sur les aspects de sémantique dénotationnelle. Nous proposons ensuite une correspondance avec le formalisme CAT. Nous proposons ensuite une correspondance entre une TBox d'une Kbase en LD et des CS, et entre une ABox et les CS. Nous terminons en reprenant les paramètres de la HS entre AD, mais cette fois en les transposant en termes de CS.

VII.1 La logique des prédicats du premier ordre

VII.1.1 Un formalisme de représentation logique

Chaque Formalisme de Logique se décompose en [57] :

Un langage : défini à l'aide de règles de réécritures ;

Un système de déduction : ensemble de règles d'inférences qui permet de construire de nouvelles formules ;

Des règles de valuation : qui permettent de calculer une valeur pour chaque formule, qualifiées de sémantique.

Une *preuve* est une suite finie de formules telle que toute formule de la suite est soit un axiome, soit obtenue à partir des précédentes à l'aide d'une règle de déduction. Un *théorème* est une formule dont il existe une preuve. Pour établir des preuves, on n'utilise pas toujours les règles d'inférences à l'état brut, mais plutôt des règles dérivées.

VII.1.2 Les entités manipulées en LPO

La logique des prédicats du premier ordre (LPO), permet de parler d'individus. Les formules acceptées en LPO s'appellent les formules bien *formées*, nous les notons F. Une formule bien formée peut se construire à partir des *éléments*[1] suivants et des règles qui leur sont associées :

1. On utilise les entités[2]élémentaires constantes, appelées simplement *constantes*, et notées ici a, b, etc.

[1]Le terme "élément" sera réservé pour désigner un symbole élémentaire d'un alphabet formel introduit de façon précise par des règles dans le cadre d'un formalisme de représentation des connaissances quelconque.

2. On utilise les entités élémentaires variables, appelées *variables*, et notées ici x, y, z, etc.
3. On peut utiliser des *quantifications* sur les variables : la quantification *universelle* notée $\forall x$, et la quantification *existentielle* $\exists y$. On ne peut pas quantifier les constantes. On ne peut pas quantifier des prédicats (définis ci-dessous) non plus. On peut quantifier les variables utilisées dans une formule. Par exemple, $F = \forall x \ P(x)$, la quantification porte sur le prédicat qui suit, ou sur le bloc entre parenthèses qui suit, c'est sa *portée*. Attention, on ne peut pas quantifier des variables à l'intérieur des prédicats (i.e. $P(\forall x)$ est interdit).
4. F peut être égale à un *prédicat*. Les prédicats ont une *arité* associée n, leur nombre *d'arguments*. Les arguments sont chacun soit une et une seule variable, soit une et une seule constante à la fois. Toutes les formules : $F = P(x)$; $F = P(a)$; $F = R(x, y)$; $F = R(a, z)$ sont bien formées.
   Attention, on ne peut pas donner un prédicat en argument d'un prédicat (comme par exemple $P(P(x))$), et un prédicat n-aire prend obligatoirement n arguments exactement.
5. On peut former des formules *composées* en connectant les prédicats entre eux à l'aide des *quatre connecteurs* : $\neg, \vee, \vee, \rightarrow$. Les connecteurs ne lient que des prédicats, par exemple F = $P(x) \wedge Q(y)$, $F = P(x) \vee Q(x)$, $F = P(x) \rightarrow R(x, y)$, la négation ne porte que sur un prédicat $F = \neg P(x)$. On peut ainsi former des formules composées par récurrence. Attention, on ne peut pas connecter des variables ou des constantes pour en faire des arguments composés dans un prédicat.

VII.1.2.0.c Remarque : La LPO permet d'exprimer des connaissances stables, et considérées comme des entités séparées (aux contours définis), dans un univers complet [57]. Cela pose donc des problèmes pour représenter des connaissances dont on peut difficilement dissocier les composants (problèmes de granularité) et qui évoluent dans le temps (dynamiques).

VII.1.3 Sémantique classique de la LPO

Dorénavant, on ne considère plus que les formules bien formées.

VII.1.3.1 Modèle

L'hypothèse dans laquelle on se place en général en représentation des connaissances, est que pour une classe d'applications donnée, il existe une approximation du monde qui convient pour *toutes* les opérations à envisager[3].

Définition	*Les représentations que nous exprimons s'interprètent dans une approximation unique appelée modèle.*

[2]Le terme "entité" sera réservé pour désigner un objet élémentaire non abstrait, le mot "objet" lui-même étant traditionnelment réservé à d'autres usages en informatique. Le fait de parler d'entité élémentaire ici est donc un pléonasme pédagogique.

[3]L'alternative étant de considérer que différentes connaissances accessibles (et nécessaires) simultanément proviennent de différentes approximations.

La *compositionnalité* consiste à dire que les symboles avec lesquels on forme les représentations ont eux-aussi une interprétation dans le modèle[4].

VII.1.3.2 Sémantique dénotationnelle

Cependant, la notion classique d'interprétation présuppose que le modèle est constitué *d'objets*, et que parmi les *symboles* de représentation, il en est qui s'interprètent comme objets du modèle. Les symboles *dénotent* alors des objets, on parle de *sémantique dénotationnelle*.

Les formules dénotent des sous-ensembles d'éléments, les relations entre prédicats en termes de connecteurs et de quantification ont une sémantique traduite en termes de relations entre éléments d'ensembles. La sémantique d'une formule est donc une *extension* dénotée par elle. On représente les dénotations des formules par des diagrammes de Venn.

VII.1.3.3 Interpréter une formule

Plus concrètement, interpréter une formule de LPO, c'est trouver un *monde* possible dont la formule soit une *description :*

Définition | *Interpréter une formule de LPO, c'est en produire un* modèle.

Produire un modèle, revient alors à imaginer puis décrire une *situation* dans laquelle chaque élément de la formule ait une entité correspondante dans cette situation.

On considère que la formule est une *représentation* de la situation et des entités du modèle. Il faut donc aussi que chaque entité du modèle soit formulable en LPO.

VII.1.3.4 Base de connaissances en LPO

Une base de connaissances en LPO est un ensemble de formules tel que toutes les formules soient cohérentes : entre elles, et avec le modèle.

Définition | *Une Kbase KB de LPO vérifie :*
1. On n'a pas simultanément : $KB \models P(a)$ et $KB \models \neg P(a)$;
2. Le modèle est valide pour chacune des formules de KB.

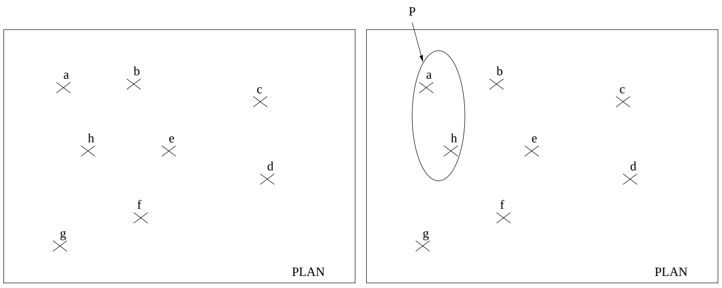
VII.1.4 Recensement des interprétations ensemblistes de structures de la LPO

VII.1.4.1 Constantes : domaine d'interprétation

Pour produire un modèle, nous devons lister un ensemble d'entités sur lesquelles va s'appuyer la représentation des formules. Dans le cadre des modèles ensemblistes, de telles entités sont

[4]Ce principe n'est pas respecté dans la plupart des représentations naturelles (exemple de "pomme de terre").

FIG. VII.1 – Domaine d'interprétation pour des formules de LPO avec $D = \{1, 2, 3, 4, 5, 6, 7, 8\}$, et domaine d'interprétation pour $P(x)$, $D_P = \{a, h\}$



appelées des *individus*. L'ensemble des individus du modèle constitue le *domaine d'interprétation*.

En général, les individus du domaine d'interprétation sont représentés abstraitement par des croix, que l'on va ensuite inclure dans des diagrammes de Venn pour constituer les ensembles du modèle.

Par exemple, si nous choisissons le domaine d'interprétation D de 8 objets *abstraits*, nommés a à h, il correspond à l'ensemble :

$$D = \{a, b, c, d, e, f, g, h\}$$

que l'on représente de manière ensembliste par 8 croix réparties au hasard dans le plan, comme montré sur la figure VII.1.

Définition | *Chaque individu possède un nom unique qui le différencie des autres, c'est son identificateur.*

VII.1.4.2 Variables

Les variables sont aussi des individus, et sont donc elles aussi représentés abstraitement par des croix.

Définition | *Parmi les individus possibles d'un monde, ceux qui sont concrètement observés dans une situation sont désignés comme les individus effectifs.*

Les constantes ont toutes un identifieur, les variables sont des individus sans identifieur. Notons qu'en LD, un individu aura forcément un identifieur, puisqu'il n'est introduit que lors d'une assertion à son sujet.

VII.1.4.3 Quantifications

La formule $\forall x P(x)$ se modélise par un domaine $D_P = D$ puisque dans ce cas $P(x)$ est valide pour toute entité de D. Et la formule $\exists x P(x)$ se modélise par une situation où D_P englobe au moins une croix, n'importe laquelle, en absence d'autres informations contraignant son identité, et peut-être plus d'une croix.

Cette deuxième formule exprime potentiellement plusieurs situations, qui sont soit sélectionnées ensuite par d'autres formules, soit restent toutes possibles. Il faut en *choisir* une comme modèle, car on ne peut pas représenter toutes les situation d'un coup sur un dessin. En effet, un dessin correspond à un choix de :

– situation possible pour illustrer la configuration que représente la formule,
– instanciation de la catégorie de mondes décrits par la formule,
– représentation en *extension* des formules.

tous ces points de vue sont *équivalents*.

VII.1.4.4 Domaine de validité d'un prédicat

Donner une interprétation pour un *prédicat unaire* $P(x)$, sur un domaine, correspond à donner l'ensemble D_P, des éléments pour lesquels le prédicat est *valide*. Par exemple, si nous devons représenter $P(x)$ tel que seulement $P(a) = v$ et $P(h) = v$, le domaine de validité est alors $D_P = \{a, h\}$, et on le *représente* comme sur la figure VII.1.

L'interprétation d'un prédicat binaire $R(x, y)$ doit représenter l'ensemble D_R des couples $(x, y) \in D \times D$ tels que R soit valide pour eux. On peut soit représenter ces couples par des flèches allant des x vers les y sur le même diagramme que pour les prédicats unaires, soit en dessinant des diagrammes de Venn sur le plan des couples de $D \times D$.

De même, l'interprétation des prédicats n-aires, se fait soit en prenant une convention permettant de lier les n variables de chaque n-uplet tel que le prédicat est valide pour lui, soit en dessinant des diagrammes de Venn sur le plan des n-uplets dans D^n.

VII.1.4.5 Domaine de validité des formules composées

Pour donner le domaine de validité de toute formule, il suffit de savoir calculer la valeur de vérité de toute formule en fonction de ses composants et de la manière dont ils sont connectés. Cela revient à savoir construire le domaine de validité pour chaque type de *connexion* (de connecteur) en fonction des domaines des prédicats connectés.

Par exemple, donnons-nous maintenant un deuxième prédicat $Q(x)$, tel que $D_Q = \{a, b\}$ et cherchons le domaine de $F(x) = P(x) \wedge Q(x)$: c'est l'ensemble des points pour lesquels cette formule est valide. Pour le savoir, nous écrivons la *table de vérité* de F en fonction de celles de P et Q :

$P(x)$	$Q(x)$	F
v	v	v
v	f	f
f	v	f
f	f	f

où "v" signifie vrai et "f" signifie faux. F est valide uniquement quand P et Q le sont en même temps toutes les deux, en termes de domaines de validité cela veut dire que $D_F = D_P \cap D_Q$.

FIG. VII.2 – a) Domaine d'interprétation pour $F(x)$, $D_F = \{a\}$, pour $F(x) = P(x) \wedge Q(x)$. b) représentation d'une situation vérifiant $P(a) \wedge R(a, b)$ sur le même domaine d'interprétation.

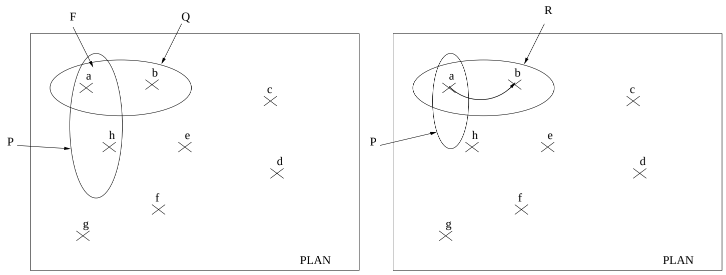


FIG. VII.3 – Table de vérité de l'implication en LPO

$P(x)$	$Q(x)$	$H(x)$
v	v	v
v	f	f
f	v	v
f	f	v

Dans notre exemple, $D_P = \{a, h\}$ et $D_Q = \{a, b\}$ donc $D_F = \{a\}$, et cela se représente comme sur la figure VII.2.

Si les prédicats connectés sont n-aires, avec $n > 1$, il faut calculer les intersections et les unions d'ensembles dans l'espace D^n.

Si le connecteur lie deux prédicats d'arités différentes, n et m, avec $n < m$, on se ramène à l'espace D^n et on représente le domaine du prédicat d'arité m par une autre convention. Classiquement, pour une différence de dimensions $(m - n) = 1$, on utilise des arrêtes pour représenter le domaine de celui dont l'arité est plus grande. Par exemple $P(a) \wedge R(a, b)$ se représente comme sur la figure VII.2 b).

Pour calculer le domaine de validité d'une formule :

– Avec un connecteur $\wedge$, il faut prendre l'intersection des domaines de validité des formules connectées ;
– Pour calculer le domaine d'un $\vee$, il faut prendre l'union des domaines des formules connectées ;
– Pour une négation, il faut prendre le complément ;
– Pour une implication, le calcul est moins intuitif : les cas possibles pour que la formule $F = P(x) \rightarrow Q(x)$ soit valide sont, soit $P(x)$ est fausse, soit $P(x)$ est valide en même temps que $Q(x)$ (voir table de vérité, figure VII.3). Le domaine pour F dans ce cas est donc donné par $D_F = D_{\neg P} \cup (D_P \cap D_Q)$.

Ces correspondances entre opérations ensemblistes et connecteurs sont résumées sur la table VII.1.

TAB. VII.1 – Calcul du domaine de validité ensembliste d'une formule de LPO en fonction de ses composants pour chaque type de connecteur impliqué.

Connecteur	Opération Ensembliste
$P(x) \wedge Q(x)$	$D_P \cap D_Q$
$P(x) \vee Q(x)$	$D_P \cup D_Q$
$\neg P(x)$	$D \setminus D_P$
$P(x) \to Q(x)$	$D_{\neg P} \cup (D_P \cap D_Q)$

VII.2 Modèles pour la LPO en CS

Nous allons maintenant essayer de donner un modèle d'interprétation pour chaque structure exprimable dans les formules en LPO, mais avec une représentation à base de CS. Cette idée vient du fait que les modèles donnés usuellement aux formules de LPO sont ensemblistes, et que les CS sont définissables de manière ensembliste aussi. Nous cherchons à étudier les formules de la LPO et à vérifier si elles sont exprimables à l'aide de CS [95]. Nous analyserons ensuite les différences avec les modèles ensemblistes présentés ci-dessus.

VII.2.1 Représentation des constantes et variables

Les constantes et variables de la LPO sont interprétées comme des entités élémentaires, les individus. Dans les modèles ensemblistes, nous avons utilisé des croix pour modéliser les individus. Dans le formalisme des CS, nous modéliserons (par analogie) les individus par les sommets, c'est-à-dire les simplexes de dimension 0.

Ainsi, le domaine D présenté en figure VII.1, se dessinera à l'aide des CS avec des sommets plongés dans un *espace* algébrique, au moins de la dimension n du complexe le plus grand à représenter (voir figure VII.4).

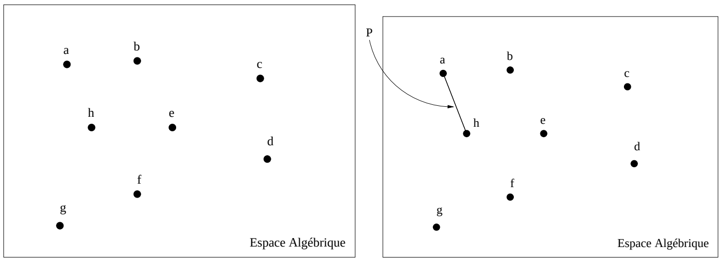
VII.2.2 Représentation des quantificateurs

Une fois que nous avons modélisé les variables, nous cherchons une correspondance en termes de modèle pour l'utilisation de quantificateurs dans les formules de la LPO.

VII.2.2.1 Quantification universelle

La formule $\forall x P(x)$, nous l'avons dit, désigne le domaine de validité D_P égal à D puisque le prédicat P est valide pour tout individu de D. Nous proposons que sa modélisation à l'aide de CS corresponde au simplexe *maximal* formé à partir de tous les sommets de D. Dans le cas présenté sur la figure VII.4, le simplexe sera de dimension 7 (nombre de sommets -1) et n'est pas représentable sur un plan.

FIG. VII.4 – a)Modélisation du domaine $D = \{a, b, c, d, e, f, g, h\}$ à l'aide du formalisme des CS : les individus sont des simplexes de dimension 0, c'est-à-dire des sommets. b)Modélisation du prédicat P tel que $D_P = \{a, h\}$, c'est un 1-simplexe $\langle a, h\rangle$ s'appuyant sur les sommets a et h.



VII.2.2.2 Quantification existentielle

La formule $\exists x P(x)$ est modélisée par un domaine D_P contenant au moins un individu, non identifié. Sa modélisation en termes de CS, correspond à un simplexe de dimension au moins 0 ; c'est donc au moins un sommet. En l'absence d'informations supplémentaires, on ne peut le figurer plus précisément, sauf en commettant des choix (comme pour la modélisation classique avec des ensembles).

VII.2.3 Représentation des prédicats du premier ordre

VII.2.3.1 Une première proposition : représenter avec des simplexes

A partir de la modélisation des individus, nous pouvons dessiner les domaines d'extension pour un prédicat unaire, selon le même procédé que pour les ensembles classiques.

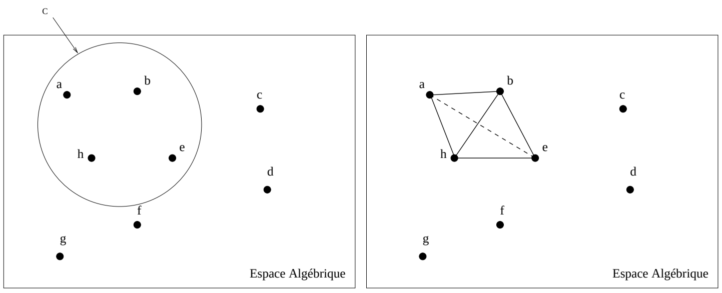
Reprenons l'exemple précédent (voir page 168), où $D_P = \{a, h\}$. Ce domaine englobe les deux individus a et h. Dans le formalisme des CS, englober deux sommets, revient à créer un simplexe de dimension supérieure, s'appuyant sur ces sommets.

Ici, il s'agit de deux sommets, a et h, nous introduisons donc le 1-simplexe $\langle a, h\rangle$ pour modéliser D_P, comme le montre la figure VII.4.

Avec les CS, nous disposons de simplexes de dimensions arbitraire n, pour représenter des domaines aussi grands que désiré. Par exemple, le domaine d'un prédicat $C(x)$ tel que $D_C = \{a, b, h, c\}$ sera modélisé en CS, par un tétraèdre, alors qu'il ne serait modélisé en ensembles classiques que comme un plus gros diagramme de Venn à 4 croix, comme le montre la figure VII.5.

Mais si les dimensions supérieures sont réservées pour les domaines étendus, comment représenter les domaines des prédicats d'arité supérieure à 1 ? Nous allons donner une seconde proposition, permettant de répondre à cette question.

FIG. VII.5 – Modélisation du prédicat C tel que $D_C = \{a, b, h, c\}$, c'est un 3-simplexe $\langle a, b, h, c\rangle$, c'est-à-dire un tétraèdre, s'appuyant sur les sommets a, b, h et c.



VII.2.3.2 Une deuxième proposition : représenter avec des complexes

Pour un prédicat P d'arité n quelconque, on représente chaque n-uplet de son domaine D_p comme un simplexe de dimension n. Le domaine D_p est alors le complexe (composé des simplexes de dimension n qui vérifient le prédicat (et non plus le simplexe de dimension n). Ainsi, le domaine de l'exemple ci-dessus se représente cette fois par le complexe $\langle\langle a\rangle, \langle b\rangle, \langle c\rangle, \langle d\rangle\rangle$. Et un prédicat R d'arité 2 par exemple aura un domaine représenté par un complexe dont les composantes, non forcément connexes, seront des simplexes à 2 sommets, c'est-à-dire des arêtes.

VII.2.4 Connecteurs

Pour savoir modéliser n'importe quelle formule de LPO, il suffit de savoir modéliser toute formule composée des *connecteurs* de la LPO, puisque c'est ainsi que l'on définit toutes les formules *valides* de la LPO.

VII.2.4.1 Connecteur $\neg$

Soit la formule de LPO composée de l'unique prédicat $\neg P(x)$. Cette formule a pour domaine de validité, le *domaine complémentaire* de D_P dans D, on le note $D_{\neg P} = D \setminus D_P$. Mais il reste à déterminer *quelle* modélisation on peut associer au *complémentaire* d'un CS dans un autre. En effet, on peut soit :

a) Modéliser directement le simplexe complément de celui modélisant D_P dans D : mais alors se pose la question de prévoir ce qu'est le complémentaire d'un simplexe par rapport à un autre ?

b) Modéliser le simplexe complément comme celui s'appuyant sur l'ensemble des sommets n'étant pas inclus dans la modélisation de D_P.

VII.2.4.2 Degré de complémentarité entre CS

Soit $D = \{a, b, c, d\}$ et soit $D_P = \{a, b, c\}$ déterminons le complémentaire $D \setminus D_P$.

Le simplexe S_D représentant D s'écrit en écriture maximale : $S_D = \langle a,b,c,d \rangle$. Il contient toutes ses faces, par définition, c'est-à-dire :

$$S_D = \langle \langle a,b,c,d \rangle, \langle a,b,c \rangle, \langle b,c,d \rangle, \langle a,b,d \rangle, \langle a,c,d \rangle, \langle a,b \rangle, \langle a,c \rangle, \langle a,d \rangle, \langle b,c \rangle, \langle b,d \rangle, \langle c,d \rangle, \langle a \rangle, \langle b \rangle, \langle c \rangle, \langle d \rangle \rangle$$

VII.2.4.2.a Proposition a) On peut en prendre le complémentaire, en enlevant tout simplement la face $\langle a,b,c \rangle$ correspondant au simplexe S_P représentant D_P. Cependant, par définition, un simplexe doit contenir toutes ses sous-faces, donc on ne peut pas conserver le volume $\langle a,b,c,d \rangle$ sans conserver toutes les faces bidimensionnelles. Ainsi, si on enlève $\langle a,b,c \rangle$, on est aussi obligé d'enlever le volume du tétraèdre qui était refermé par ce simplexe.

En conséquence, par définition (voir chapitre IV), lorsque l'on veut prendre le complémentaire d'un simplexe dans un autre, dès que l'on perd une face, on perd le volume qu'elle bordait. Ainsi, le complémentaire dans notre exemple est égal à :

$$S_{\neg P} = \langle \langle b,c,d \rangle, \langle a,b,d \rangle, \langle a,c,d \rangle \rangle$$

Cette expression n'est évidement pas équivalente au simplexe formé de tous les sommets n'appartenant pas au simplexe S_P.

VII.2.4.2.b Proposition b) En effet, les sommets de D qui n'appartiennent pas à D_P dans notre exemple se réduisent à d. Ainsi, le simplexe formé des sommets de D n'appartenant pas à S_P est seulement le 0-simplexe $\langle d \rangle$.

VII.2.4.2.c Synthèse des propositions a) et b) Mais ces deux points de vues représentent les *deux extrémités* d'une conception de $D_{\neg P}$ à toutes les dimensions k possibles, compte tenu de la dimension des simplexes en jeu.

Définition | *Nous pouvons prendre le complémentaire d'un simplexe S_P dans un autre S_D à toutes les dimensions k intermédiaires entre la dimension m du simplexe S_P, et la dimension m' du simplexe formé de tous les sommets de S_D qui n'appartiennent pas à S_P.*

Dans notre exemple, $m = dim\ S_P = 2$ et $m' = 0$ nous pouvons donc prendre le complémentaire aux dimensions 2, 1 et 0 :

– le complémentaire en dimension $k = 2$ est ici égal à $S^2_{\neg P} = \langle \langle b,c,d \rangle, \langle a,b,d \rangle, \langle a,c,d \rangle \rangle$,
– le complémentaire en dimension $k = 0$ est $\langle d \rangle$,
– mais nous pouvons également définir le complémentaire en dimension $k = 1$.

En effet, à partir du complémentaire en dimension 2, nous pouvons continuer d'éplucher" S_D. Il faut alors en enlever aussi les sous-faces de S_P, c'est-à-dire les 1-simplexes $\langle a,b \rangle$, $\langle b,c \rangle$ et $\langle a,c \rangle$; nous obtenons alors :

$$S^1_{\neg P} = \langle \langle a,d \rangle, \langle b,d \rangle, \langle c,d \rangle \rangle$$

toujours en notant les simplexes maximaux, c'est-à-dire que les sommets sont toujours inclus dans $S^1_{\neg P}$.

VII.2.4.2.d Les degrés pour des requêtes différentes Les complémentaires de dimension k quelconque peuvent être utilisés pour répondre à différents types de requêtes sur des Kbases exprimées en CS. En effet, les individus étant dénotés par des simplexes de dimension 0, si nous recherchons l'ensemble de tous les individus pour lesquels le prédicat P n'est pas vérifié, (comme pour les modèles de LPO), alors il faut répondre par le complément du simplexe en dimension 0.

Dans notre exemple, cela correspond à $\langle d \rangle$. Cependant les dimensions supérieures correspondent aux dénotation des types effectifs, et l'on peut s'intéresser aux "autres" types que celui défini par le prédicat P. On répondra alors par les complémentaires de dimension supérieure.

VII.2.4.3 Connecteur $\wedge$

Soit la formule de LPO, $F(x) = P(x) \wedge Q(x)$, dans un modèle ensembliste, nous avons établi que $D_F = D_P \cap D_Q$. Mais, comme pour la négation, il faut établir la signification de l'intersection en termes de simplexes. Soit S_P le simplexe modélisant D_P et S_Q le simplexe modélisant D_Q, alors $S_{P \cap Q}$ sera la *frontière* entre S_P et S_Q. En effet, que l'on cherche à établir :

- directement le simplexe intersection des deux simplexes (le simplexe frontière),
- le simplexe formé des sommets étant dans l'intersection des domaines (pris comme ensembles de sommets)

ces deux points de vues donnent cette fois-ci la même représentation, le même simplexe, appelé *frontière*.

VII.2.4.4 Connecteur $\vee$

Soit la formule de LPO $G(x) = P(x) \vee Q(x)$, dans un modèle ensembliste, nous avons établi que $D_G = D_P \cup D_Q$. Cette union peut s'envisager de deux façons du point de vue de CS :

- Soit l'union des simplexes S_P et S_Q donnant un complexe simplicial formé de deux faces non nécessairement connexes, sa dimension est alors égale à celle du plus grand des deux ;
- Soit le simplexe formé de tous les sommets appartenant soit à D_P soit à D_Q, sa dimension est alors supérieure à celle du plus grand des deux.

Ces deux points de vue rejoignent les deux points de vue possibles sur la négation, et nous réutilisons la notion de degré de complémentarité, mais cette fois pour l'union.

VII.2.4.5 Connecteur $\rightarrow$

Soit $H(x) = P(x) \rightarrow Q(x)$. Pour établir l'équation qui relie D_H à D_P et D_Q, écrivons la table de vérité du connecteur $\rightarrow$. Le domaine de validité de $H(x)$ est celui de $P(x) \cap Q(x)$ auquel on ajoute celui de $\neg P(x)$, ainsi, $D_H = (D_P \cap D_Q) \cup D_{\neg P}$. Nous avons vu que le simplexe correspondant à $D_P \cap D_Q$ était le simplexe frontière $S_{P \cap Q}$ entre S_P et S_Q, alors que l'union entre ce simplexe frontière $S_{P \cap Q}$ et le simplexe $S_{\neg P}$ peut s'envisager de deux manières différentes correspondant aux deux points de vue cités dans le paragraphe précédent.

VII.2.5 Conclusion

Nous avons proposé une modélisation à l'aide de CS pour **chaque** structure exprimable dans les formules de la LPO. Tous les éléments de la LPO sont donc modélisables à l'aide des CS, comme on le fait classiquement avec les diagrammes de Venn. Ce résultat intermédiaire est important pour deux raisons :

1. Nous savons comment effectuer des raisonnements logiques sur des connaissances représentées en CS ; il suffit de ramener des Kbases de LPO à un expression en CS, pour leur appliquer les algorithmes développés pour les formalisme CAT dessus.
2. Nous pouvons utiliser les CS pour exprimer des Kbases écrites en LD, puisque nous savons qu'il est possible d'écrire toute formule de LD en LPO[5].

VII.3 Exprimer une base de LD en CS

VII.3.1 Les éléments d'une LD

Nous rappelons (voir chapitre II) que les deux éléments constitutifs d'une Kbase en LD sont (voir [87]) :

La Abox : ensemble d'assertions sur les individus effectifs d'une situation, les concepts utilisés pour ces assertions sont appelés effectifs ;

La Tbox : ensemble de définitions de concepts et rôles, appelées des descriptions ; on utilise pour cela la relation de subsomption (is-a), les autres rôles, des constructeurs de concepts, des constructeurs de rôles.

Afin d'appliquer des algorithmes classiques sur des Kbases, certains chercheurs ont déjà proposé de les augmenter avec ce qu'ils appellent les graphes de descriptions (GD) [58][1]. Cette traduction des descriptions d'une Tbox en graphes permet d'utiliser les algorithmes connus sur les graphes qui ont le mérite d'être complets. Afin de donner une correspondance des ABox et TBox en CS, nous allons d'abord étudier les structures exprimées GD.

VII.3.2 Les graphes de description

Küesters [58] et Borgida [1] introduisent les graphes de descriptions (GD) dans deux articles différents en utilisant des notations légèrement différentes, mais en se basant sur des définitions fondamentalement semblables. La définition que nous prenons comme référence ici est celle donnée par Borgida [1] pour la LD simple du système CLASSIC.

Notons que les GD ne sont calculés que pour des *descriptions* de concepts, c'est-à-dire des Tbox.

[5]La LPO est même plus expressive, puisqu'il est impossible d'écrire des prédicats à plus de 3 variables en LD. Nous n'avons donc besoin de représenter que des prédicats d'arité 3.

A *une* description correspond *un et un seul* graphe. Une Tbox étant un *ensemble* de descriptions, elle est donc représentée par un *ensemble* de graphes de descriptions.

Lorsque l'on procède à *l'appariement* de structures entre deux descriptions (une requête et un candidat au matching par exemple, voir chapitre III) c'est donc toujours *deux* graphes que l'on compare.

Les problèmes d'appariement et d'adaptation qui nous intéressent, sont donc ramenés dans le cas des GD à un problème de mapping entre deux graphes individuels, non plongés dans un graphe plus grand. Cette remarque nous inspirera dans la section suivante lorsque nous ferons de l'appariement entre CS.

Mais avant de poursuivre, rappelons la définition formelle des GD.

VII.3.2.1 Définition d'un graphe de descriptions

Un *graphe* de *description* est un graphe *connecté*, *étiqueté* et *orienté* qui reflète la *structure* syntaxique des descriptions de concepts en LD. Les noeuds sont étiquetés et correspondent aux descriptions, les arêtes correspondent aux restrictions sur les rôles ou les attributs.

Voici la définition récursive d'un graphe de descriptions :

1. il existe un noeud r appelé racine, connecté à une île de noeuds par des arêtes étiquetées par des attributs ;
2. les noeuds de cette île peuvent avoir 0 ou plusieurs arêtes étiquetées par des rôles ; ces arêtes quittent ces noeuds et pointent vers des noeuds d'autres graphes ;
3. ces autres graphes n'ont aucun noeud ou arête en commun avec les autres, ni avec les noeuds des îles qui pointent vers eux.

Plus formellement, un graphe de description est un triplet (N, E, r) où :

- N est un ensemble de noeuds ;
- E est un sac d'arêtes appelées a-edges, étiquetées avec des noms d'attributs ;
- r est un noeud particulier de N appelée racine.

Les éléments de E sont des triplets (n_1, n_2, A), où :

- n_1 et n_2 sont des noeuds ;
- A est un nom d'attribut.

Un noeud est une paire (C, H) telle que :

- C est un ensemble de noms de concepts, les atomes ;
- H est un sac de n-uplets, les r-edges du noeud

Une r-edge est un n-uplet (R, m, M, G) où :

- R est un nom de rôle ;
- m est un entier non négatif, appelé min ;
- M est un entier non négatif ou l'infini ∞ , appelé max ;
- G est un graphe de description qui représente la restriction sur les fillers du rôle R, ce graphe est souvent appelé le graphe de restriction du noeud.

VII.3.2.2 Sémantique

La sémantique d'un GD est donnée par une fonction d'interprétation I sur un domaine Δ. Dans tous les cas, l'extension d'un GD est la même que celle de la description qu'il représente. Pour traduire une description quelconque C de LD dans le système CLASSIC en son graphe de description correspondant, il suffit de savoir traduire chaque construction autorisée dans CLASSIC. Prenons donc les constructeurs dans l'ordre :

1. Un nom de concept est transformé en un noeud simple sans a-edge. Les atomes du noeud contiennent seulement le nom du concept. Le noeud n'a pas de r-edge non plus.
2. Une restriction du type $\geq n.R$, est transformée en un noeud sans a-edge. Les atomes du noeud sont des CLASSIC-THING, et il a une r-edge contenant le rôle R, le nombre min m vaut $m = n$ le nombre max M vaut $M = \infty$, et le graphe de restriction est G_{THING}.
3. Une restriction du type $\leq n.R$, est transformée en un noeud sans a-edge. Les atomes du noeud sont des CLASSIC-THING, et il a une r-edge contenant le rôle R, le nombre min m vaut $m = 0$ le nombre max M vaut $M = n$, et le graphe de restriction est G_{THING}.
4. Une restriction du type $\forall R : C$, est transformée en un noeud sans a-edge. Les atomes du noeud sont des CLASSIC-THING, et il a une r-edge contenant le rôle R, le nombre min m vaut $m = 0$ le nombre max M vaut $M = \infty$, et le graphe de restriction est le graphe de description du concept C, noté G_C.
5. La construction $C \sqcap D$ a pour graphe de description le résultat du merge entre G_C et G_D.
6. Une restriction du type $\forall A : C$, où A est un attribut, est transformée en un graphe de description G qui s'exprime en fonction de G_C. Soit $G_C = (N_C, E_C, r_C)$, alors $G = (N_c \cup \{t\}, E_C \cup \{(t, r_C, A)\}, t)$ où t est le noeud $(\{CLASSIC - THING\}, \{\})$.
7. Le graphe de description de $A_1 o ... o A_n = B_1 o ... B_m$

Remarque : Les restrictions *d'égalité* sont la seule source de création de structures non linéaires dans le graphe. Remarquons que les descriptions canoniques utilisées par Borgida dans [13] pour calculer une subsomption généralisée (pour matcher des requêtes à une LD), correspondraient à des arbres dans le formalisme des graphes de description. Si le graphe de descriptions est un arbre, il n'y a aucune différence avec une notation *linéaire* [1].

VII.3.3 Un Modèle pour les ABox en CS

Pour construire un modèle en CS pour les ABox d'une Kbase en LD, nous devons nous poser les question suivantes :

1. Peut-on donner une représentation graphique des ABox à la manière des GD ?
2. Si oui, qu'est-ce que cela apporte ?
3. A quoi correspondent les dimensions supérieures disponibles avec des CS par rapport à des GD ?
4. Et les duaux des CS ?

VII.3.3.1 Représenter la relation d'instanciation avec des CS

Notations :

- Soit I l'ensemble des individus effectifs de la ABox à représenter.
- Soit C_e l'ensemble des concepts effectifs dont les individus peuvent être des instances.
- Soit λ la relation binaire d'instanciation, aussi appelée *membre-de*.

Nous pouvons alors écrire la matrice d'incidence de $\lambda : I \rightarrow C_e$ et chaque ligne (ou chaque colonne) de cette matrice est alors représentable comme un simplexe appartenant au complexe (ou le dual) représentant λ (ou λ^{-1}).

Reprenons l'exemple de la petite équipe du chapitre II, page 54. Nous avons la Tbox T_0 suivante :

$Humain \leq ANYTHING$

$Homme \leq Humain$

$Femme \leq Humain$

$Ensemble \leq ANYTHING$

$membre \leq ANYRELATION$

$chef \leq membre$

$(DISJOINT\ Homme\ Femme)$

$Equipe \doteq (AND\ Ensemble\ (ALL\ membre\ Humain)\ (ATLEAST\ 2\ membre))$

$PetiteEquipe \doteq (AND\ (Equipe\ (ATMOST\ 5\ membre))$

$EquipeModerne \doteq (AND\ Equipe\ (ATMOST\ 4\ membre)\ (ATLEAST\ 1\ chef)\ (ALL\ chef\ Femme))$

A partir de laquelle nous définissons la Abox A_0 suivante :

$(EquipeModerne\ \#equipeA)$

$(Homme\ \#joe)$

$(Homme\ \#harry)$

$(Humain\ \#mary)$

$(membre\ \#equipeA\ \#joe)$

$(membre\ \#equipeA\ \#harry)$

$(chef\ \#equipeA\ \#mary)$

L'ensemble des individus effectifs est alors :

$$I(A_0) = \{\#joe, \#harry, \#mary, \#equipeA\}$$

et l'ensemble des concepts effectifs est :

$$C_e(A_0, T_0) = \{EquipeModerne, Homme, Humain\}$$

Et la relation $\lambda : I \to C_e$ est donnée par la matrice d'incidence :

λ	$EquipeModerne$	$Homme$	$Humain$
$\#joe$	0	1	x
$\#harry$	0	1	x
$\#mary$	0	0	1
$\#equipeA$	1	0	0

Les "x" représentent des "1" qui sont déduits par le moteur de déduction attaché à la Kbase. Ces déductions sont faites à partir de la relation de subsomption (donc suite à une étape de raisonnement) qui existe entre les concepts Humain et Homme.

VII.3.3.2 Analyse de la représentation en CS d'une ABox.

Les individus effectifs sont des simplexes de dimension 0. Les concepts effectifs sont les simplexes qui s'appuient sur ces sommets. En fait, ils représentent *l'extension* de chaque type, les différentes occurrences connues de ce type. C'est la sémantique du concept en termes ensemblistes, comme vu à la section VII.1, page 164.

Plus il y a d'individus représentant le concept à représenter, plus le simplexe qui représente ce concept a de dimensions.

La frontière entre les simplexes représentant deux concepts, est formée par le simplexe des éléments appartenant aux deux concepts à la fois, ou plus précisément *héritant* des deux.

VII.3.4 Une représentation des TBox en CS

Pour fournir une représentation en CS d'une TBox pour une Kbase en LD, nous devons nous poser les questions suivantes :

1. Peut-on utiliser des CS à la place des graphes de description ?
2. Si oui, qu'est-ce que cela apporte ? c'est-à-dire, à quoi correspondent, les dimensions supérieures ?
3. Quels liens y-a-t-il entre les CS pour les ABox et ceux pour les TBox, si il y en a ?

Pour répondre à la première question, nous considérons les liens entre CS et GD. Pour répondre à la seconde question, nous étudions les relations d'une TBox pouvant nécessiter une représentation avec des dimensions supérieures à 1 (les graphes sont des CS de dimension 1, voir chapitre IV). Enfin, nous présentons une synthèse des représentations que nous proposons pour les ABox et TBox d'une Kbase en LD.

VII.3.4.1 Liens entre CS et GD

Un GD est un cas particulier des CS, il est donc trivial d'en établir la correspondance en termes de CS. Cependant, l'intérêt d'utiliser des CS se situe essentiellement au niveau de l'exploitation des dimensions supérieures. Dans quels cas peut-on avoir besoin de dimensions supérieures pour un GD ? Il existe deux situations à étudier :

TAB. VII.2 – Relation Part-of en LD

I	**F**	**H**	relation	ex(partie,tout)
0	0	0	stuff	(acier,vélo)
0	0	1	lieu	(oasis,désert)
0	1	0	opération	(avaler,manger)
0	1	1	?	
1	0	0	membre	(arbre,forêt)
1	0	1	portion	(part,tarte)
1	1	0	composant	(roue,voiture)
1	1	1	?	

– les relations de structure, de la famille de part-of par exemple ;
– les relations n-aires.

VII.3.4.2 Les relations part-of en LD

Il existe différents types de relations part-of qui peuvent se classer en fonction d'une décomposition en quelques propriétés [3] :

H homéomorphisme des parties par rapport au tout,

F fonctionnalité de la partie,

I caractère indispensable de la partie.

Nous listons dans la table VII.2, toutes les combinaisons possibles de ces propriétés, sous forme de vecteurs de 0 et de 1 : 0 lorsque la propriété n'est pas vérifiée, et 1 lorsqu'elle est vérifiée. Pour chaque combinaison, nous caractérisons le type de relation part-of par un nom et un exemple donné par un couple (partie,tout).

On se demande ensuite, s'il est possible de trouver une relation part-of telle que la partie remplisse une fonction et soit homéomorphe au tout. Par ailleurs, la relation appelée stuff, est apparentable à la notion d'attribut (voir chapitre II). On se demande également alors "est-ce que les attributs des concepts en sont des parties ?". La réponse que nous proposons est de considérer les attributs comme une classe différente de relations (voir section VII.3.4.4).

Nous remarquons que toutes ces relations part-of sont binaires puisqu'elles se définissent en termes de la relation d'une partie à son tout. Elles sont donc représentables par des rôles classiques en LD en introduisant des constructeurs spécialisés. C'est d'ailleurs ce que font [3][39]. Ce n'est pas par ce biais que nous aurons besoin de dimensions supplémentaires pour représenter des choses dans des GD.

VII.3.4.3 Les agrégats

Dans le cadre de l'étude des relations part-of nous devons réserver une attention particulière à la notion d'agrégats [39]. En effet, les agrégats sont le résultat du nommage d'une *structure*

FIG. VII.6 – Relation part-of abstraite, notée + liant trois parties h, i, j au tout y, et sa représentation simpliciale.

y	h	i	j
$+$	1	1	1

FIG. VII.7 – Relation part-of abstraite, notée + liant deux parties x, y au tout a, et sa représentation simpliciale.

a	x	y
$+$	1	1

construite à l'aide des relations part-of listées dans la table VII.2. Les agrégats, contrairement aux relations part-of (briques élémentaires pour les construire) peuvent nécessiter l'introduction de représentations à des dimensions plus grandes que les GD. Par ailleurs, des parties composées à l'aide de relations part-of vont non-seulement entretenir des relations avec le tout, mais aussi entre elles. Les relations entre parties découlent des propriétés des relations qui les lient au tout. Le fait d'introduire et de représenter un agrégat correspond à *expliciter* le tout [43], pour y faire ensuite référence.

Mais le problème principal posé par la représentation graphique des agrégats, est le problème de la *granularité*. En effet, comment représenter un agrégat composé d'agrégats ? Et comment représenter des variations de granularité ? La solution que nous proposons est de représenter la relation binaire entre les parties P_i d'un agrégat A et les relations r_i qui les lient à cet agrégat A. Si ces parties p_i sont elles-mêmes des agrégats, nous introduisons alors la relation $p_i \in \{r_i\}$ et nous considérons les liens entre cette partie et les parties *élémentaires*.

Afin de préciser notre propos, prenons un exemple. Soit la portion de Tbox suivante :

$a := x + y$

$y := h + i + j$

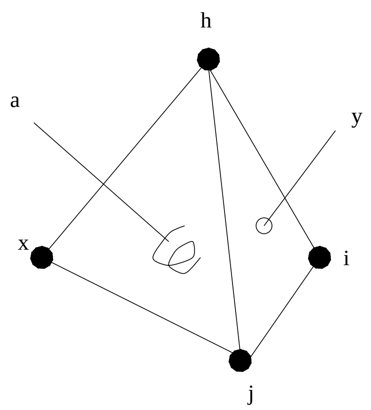
où l'opérateur + symbolise n'importe quelle relation part-of (pour l'instant). Pour représenter cet agrégat, nous pouvons dresser la table d'incidence des parties de y avec les relations qui les lient, comme montré sur la figure VII.6.

Cette relation se représente simplicialement par une face, un 2-simplexe, $\langle h, i, j\rangle$, appuyé sur les sommets h, i, j, (voir aussi figure VII.6). Par ailleurs, l'agrégat a se construit en dressant la table d'incidence de la figure VII.7.

Cette relation se représente par le 1-simplexe $\langle x, y\rangle$, (voir aussi la figure VII.7). Mais comment reporter la composition de y sur ce simplexe représentant a ? On peut choisir d'appuyer la représentation de a sur la face $\langle h, i, j\rangle$ d'une part, et sur x d'autre part. Mais cela donne :

1. soit quelque chose qui n'est pas un CS : un lien 1-D entre une face 2-D et un sommet ;
2. soit un simplexe formé des 4 sommets, c'est-à-dire de dimension supérieure, noté $\langle h, i, j, x\rangle$;
3. soit un complexe composé des deux simplexes $\langle\langle h, i, j\rangle, \langle x\rangle\rangle$.

FIG. VII.8 – a) Représentation simpliciale de la relation abstraite part-of + entre les parties h, i, j en un sous-agrégat y lui même sous-partie de l'agrégat a contenant par ailleurs x. La spirale représente une flèche qui pointe sur le volume, tandis que la flèche en cercle pointe sur la face 2D. Nous donnons également la matrice d'incidence correspondante.



a	h	i	j	x
$+$	1	1	1	1
y	1	1	1	0

Ces cas sont tous insatisfaisants à eux seuls car :

1. Le premier cas est exclu car cela nous éloigne des CS, et que le but est ici d'utiliser les complexes simpliciaux et en particulier leurs caractéristiques dimensionnelles ;
2. dans le deuxième cas, nous perdons l'information intéressante qui consistait à nommer le sous-agrégat h, i, j par y.
3. enfin, ce troisième cas nous oblige à lier les parties h i et j par une opération topologique, et les parties x et y par une autre, différente ; alors qu'a priori, la loi de composition (relation part-of) était la même au départ.

La solution que nous proposons est d'introduire la notion de "relation" y . Si une partie de l'agrégat a est nommée, distinguée, nous introduisons une *relation abstraite*, appelée aussi y (comme le sous-agrégat à représenter), par laquelle toutes les parties élémentaires de a qui sont regroupées en un sous-agrégat y sont liées : on dira que "h,i et j sont y-reliés". Cette nouvelle façon de concevoir l'agrégat a se traduit par la matrice d'incidence globale de la figure VII.8.

Les parties h,i,j sont toujours liées par la relation $+$, de même que x, et donnent une représentation en simplexe 3D (voir figure VII.8), un tétraèdre, noté $\langle h, i, j, x \rangle$; comme dans le cas 2 précédent. Cependant, nous représentons le simplexe supplémentaire, correspondant au sous-agrégat y, comme une face de ce tétraèdre (voir figure VII.8), notée $\langle h, i, j \rangle$. Sur cette représentation simpliciale, nous représentons de manière spatiale les sous-parties d'un agrégat à l'aide de la métaphore spatiale des relations d'inclusion. Cette représentation nous permet de ne pas perdre l'information sur les sous-agrégats tout en explicitant le "sous-tout" y. Le fait de l'expliciter nous permet d'y faire explicitement référence (référence directe, par opposition à par le contenu [88]). Par extension de cette méthode, nous pouvons alors faire référence à tout sous-agrégat de a, par exemple $\langle x, h \rangle$, bien qu'il n'ait pas été explicitement introduit dans la Tbox. Grâce à l'étiquetage des arêtes par le type de relation part-of qui les lie, nous pouvons en déduire ce qui lie ici x à h, par exemple. Bien sûr, dans cet exemple, toutes les relations étaient de même nature et notées $+$, mais nous pouvons imaginer que deux relations part-of différentes entrent dans la description.

Par exemple, voici la description d'un $Orchestre$:

$$Orchestre := \exists membre : Musicien \sqcap \exists membre : ChefOrchestre$$

dont on donne par ailleurs la description d'un $Musicien$:

$$Musicien := \exists composant : Instrument \sqcap \exists composant : Partition$$

où l'on veut mentionner que le musicien est membre de l'orchestre et que l'instrument (de musique) est un composant du musicien (il remplit une fonction). Les représentations simpliciales de la première description se traduisent par un simplexe de dimension 1, de même que celle de la deuxième description. Si l'on intègre les deux descriptions dans la représentation globale de l'agrégat orchestre, nous obtenons un simplexe de dimension 3 dont une arête est la représentation du musicien.

Remarque : il reste des questions ouvertes à ce sujet, et il serait intéressant de les développer dans un travail futur. En effet, elles ne se rapportent pas directement au problème des agents sémantiquement hétérogènes. Citons, par exemple :

- Que peut-on dire au sujet de l'arête $\langle Partition, ChefOrchestre \rangle$?
- Pouvons nous établir des lois de composition des différentes relations part-of ?
- Si un concept est composant d'un concept qui est membre, est-il membre ?
- ou composant ?
- ou autre chose ?

VII.3.4.4 Toutes les relations d'une Tbox

Les relations d'une LD se répartissent alors en trois familles de relations utilisables dans une Tbox :

1. Relation de spécification de type des concepts : relation "is-a" ;
2. Relation de spécification de structure : relation "part-of", plus précisément "lieu", "opération", "membre", "portion" ou "composant".
3. Relation de spécification d'un attribut de concept : relations de traits comme "couleur", "forme", "fait-de".

Ces trois types de relations peuvent être classés selon les implications aux niveau de l'héritage. Voici le récapitulatif de ces caractéristiques, où les lettres H, F, I désignent respectivement l'homéomorphisme, la fonction et le caractère enlevable comme précédemment :

	A is-a B	part-of					attribut
		lieu	opération	membre	portion	composant	
B hérite	attributs de A	H de A	F de A	I de A	I,H de A	I,F de A	rien de A

VII.3.4.5 Une représentation en CS par type de problème

Nous proposons de considérer les trois familles de problèmes à résoudre avec une Tbox, qui font intervenir des types de connaissances différentes :

– Héritage : Concepts $\times$Concepts ;
– Référence : Concepts $\times$Relations (dont part-of avec constructeurs dédiés) ;
– Monde effectif : Individus $\times$Concepts (effectif seulement).

sachant que nous pouvons construire les CS correspondant aussi aux liens Relations (effectives) $\times$ Individus, Relations $\times$ Relations, Individus $\times$ Individus, etc ; mais que ces CS vont présenter peu d'intérêt pour résoudre des problèmes, et notamment des requêtes. Pour résoudre chaque catégorie de problème, on utilise une représentation simpliciale différente.

VII.4 Conclusions du chapitre

Nous avons établi une modélisation des Kbases en LD à l'aide de CS. Cette modélisation repose sur la définition ensembliste des CS, ainsi que sur la sémantique dénotationnelle ensembliste elle-aussi, associée aux LD. Un certain nombre de questions sont encore ouvertes pour achever cette correspondance notamment au niveau des raisonnement possibles sur une telle modélisation.

Cependant, ces résultats sont très prometteurs et nous poussent à croire qu'il est possible d'utiliser les algorithmes d'adaptation proposés par ailleurs et basés sur le formalisme des CS, pour résoudre une certaine catégorie de problèmes inclus dans la problématique des agents hétérogènes, formalisé en LD.

Le travail qui reste à accomplir se situe surtout au niveau du traitement des requêtes, par le biais de matchings entre structures : la structure représentant la requête, et la structure représentant la Kbase interrogée. C'est lors de tels traitements qu'interviendront les algorithmes d'adaptation dont nous avons parlé. Les algorithmes de matching, par ailleurs, vont également utiliser avec profit le modèle de résolution d'analogies développé pour les CS, puisqu'il y avait également une étape d'appariement de structures [89].

Chapitre VIII

Analyse pour une intégration des outils et exemples

Introduction

Après avoir formalisé le problème de l'hétérogénéité sémantique entre agents dialogiques à l'aide de la logique de descriptions, nous avons cerné un sous problème crucial à résoudre qui était : la gestion d'un terme hétérogène lors d'un pas d'interaction par l'agent récepteur.

Nous avons proposé de résoudre ce sous-problème en utilisant la notion *d'adaptation* pour laquelle nous avons proposé un algorithme *topologique*. Cet algorithme topologique est entièrement basé sur une représentation du problème à l'aide d'une structure simple de la topologie algébrique : les complexes simpliciaux. Nous avons donc montré dans le chapitre précédent qu'il était possible d'établir une correspondance entre une base écrite en logique des descriptions et modélisant un agent, et une représentation simpliciale des connaissances. En effet, nous pouvons espérer *extraire* des complexes simpliciaux d'une base afin de pouvoir appliquer cet algorithme topologique sur les données concernant des agents qui communiquent dans InterViews.

Résumé du chapitre

Le but de ce chapitre est d'abord d'illustrer les correspondances entre Kbases en logique de descriptions et représentation simpliciales des connaissances. Nous allons donc nous concentrer dans un premier temps sur l'illustration des processus d'extraction de tels complexes simpliciaux à partir des bases de connaissances, ce qui constitue une application directe du chapitre précédent. Ces extractions se feront de différentes manières selon le type de connaissance envisagé (concept, individu, rôle) et le type de relation (inclusion, définition).

Une fois ces processus d'extraction illustrés, nous pourrons utiliser ces représentations simpliciales pour traiter explicitement les différentes requêtes qu'il est possible de poser à un agent. Pour cela, nous commençons par rappeler toutes les requêtes qu'il est possible de poser à une Kbase en LD. Nous pourrons alors étudier le problème qui nous intéresse : à savoir la gestion de telles requêtes à l'aide des complexes extraits précédemment. Nous verrons que répondre à de telles requêtes revient à opérer différentes opérations sur les représentation simpliciales, en fonction du type du pattern contenu.

VIII.1 Procédure générale de résolution d'une interaction hétérogène

VIII.1.1 Traitement général : rappels

Une interaction dialogique entre deux agents dans InterViews est modélisée comme un *cycle* (voir chapitre III) de pas d'interaction i, chaque pas est ensuite décomposé en :

1. Une phase d'émission d'une requête $Q^i_{e \to r}$ par la Kbase Kb_e (Kbase envoyeur) ;
2. Une phase de gestion de cette requête par la Kbase Kb_r (receveur)
3. Une phase d'émission de réponse $R^i_{r \to e}$ pour la requête $Q^i_{e \to r}$ par Kb_r lorsqu'elle a trouvé une interprétation valide de la requête reçue.

Cette décomposition est schématisée par la figure VIII.1, où l'étape 1 est symbolisée par Q_i, l'étape 2 se produit *dans* KB_r (non visible), et l'étape 3 est symbolisée par R_i. Rappelons que pour le traitement d'un cycle complet d'interaction, nous distinguons les fonctions suivantes (voir chapitre III) :

A1 : traitement du cycle complet ;

A2 : traitement d'un seul pas d'interaction ;

A3 : gestion de la requête, que nous étudions dans ce chapitre ;

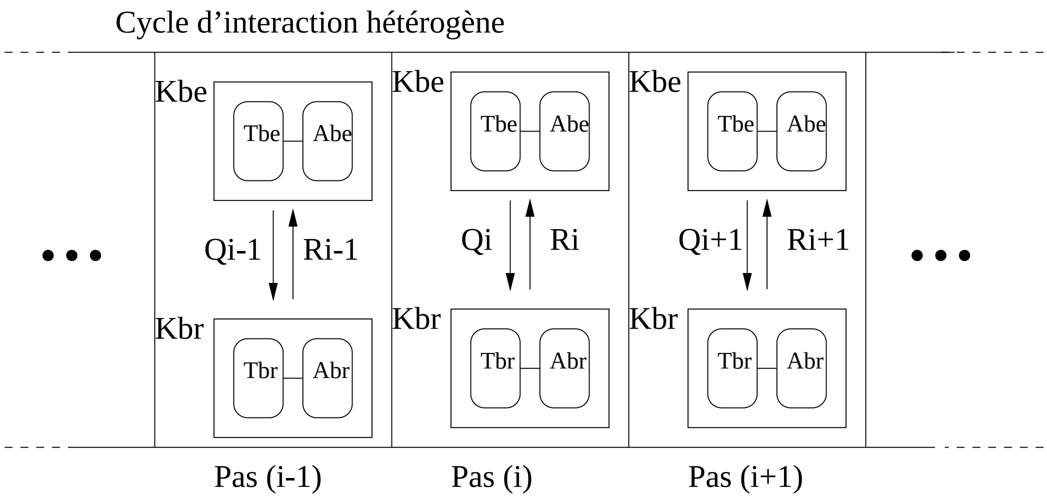
A4 : gestion des patterns de la requête, et le cas échéant des termes *hétérogènes*, que nous étudions dans ce chapitre.

VIII.1.1.1 Gestion d'un cycle d'interaction.

L'interaction est gérée par **A1**, de manière basique, selon le schéma suivant :

	`GererInteraction(KB1,KB2)`
pas d'interaction :	`incr i`
initialisations :	`initi(KBe,KB1,i)`
	`initi(KBr,KB2,i)`
appel de fonction :	`GestionPasInteraction(i,KBe,KBr)`
fin :	`Validation`

FIG. VIII.1 – Interaction dialogique entre deux agents modélisés par des Kbases dans InterViews.



VIII.1.1.2 Gestion d'un pas d'interaction

Un pas d'interaction est géré par **A2** selon le schéma suivant :

	`GererPasInteraction(i,KB1,KB2)`
émission de requête :	`Q=EmissionRequete(i,Requete(KBe))`
gestion de requête :	`R=GererRequete(Q,KBr)`

VIII.1.1.3 Gestion de la requête

Une requête est gérée par **A3** la Kbase réceptrice selon l'algorithme suivant :

	`GererRequete(Q,KBr)`
fct sur le nb de patterns de Q :	`for a=0 to Nb_patterns(Q)`
test de type sur patterns Q :	`T=TypePattern(a,Q)`
cas terminologique :	`if T in (concept,role) P=GererPatternTbox`
cas référentiel :	`else P=GererPatternAbox`
production d'une réponse	`R= ProduireReponse(P)`

Les patterns sont gérés par des fonctions qui appliquent soit des algorithmes de matching, soit des algorithmes d'adaptation dans le cas de termes hétérogènes, soit sur les représentations simpliciales, soit sur les expressions de LD. Selon le type du pattern, l'information nécessaire au matching et à l'adaptation sera contenue soit dans la Tbox soit dans la Abox. Ce sont les fonctions de gestion des patterns qui nous intéressent dans ce chapitre.

VIII.1.1.4 Gestion des patterns de la requête

La gestion des patterns d'une requête par les fonctions `GererPattern` (d'une Tbox ou d'une Abox) se fait sur un pattern à la fois. Le but de ce traitement est de produire une substitution valide pour chaque pattern, ce qui constitue l'information inclue dans la réponse que KB_r donne à Kb_e en fin de pas d'interaction. Pour produire cette substitution valide, voici les étapes qu'il faut appliquer aux patterns d'une requête :

- Détermination du type des patterns, et appel de la fonction de gestion correspondante en fonction du type, pour le traitement d'un seul pattern à la fois : cette étape préliminaire est prise en charge par la fonction ci-dessus de gestion de la requête.
- Traitement d'un seul pattern à la fois par matching, et adaptation préalable des termes hétérogènes si nécessaire : ceci est détaillé au paragraphe VIII.1.1.4.a.
- Sélection d'une substitution valide pour chaque pattern traité, et construction de la réponse envoyée à KB_e : voir paragraphe VIII.1.1.5.

VIII.1.1.4.a Appariement et adaptation pour le traitement d'un pattern Quel que soit le type du pattern (concept, rôle, individu), la requête peut contenir ou non des termes hétérogènes. La gestion d'un terme hétérogène se fait par adaptation, en utilisant une représentation simpliciale des connaissances. Le but de ce chapitre est de détailler les algorithmes du traitement d'une requête hétérogène, que nous avions regroupés sous la désignation **A4** dans le chapitre III. Pour cela, nous commençons par aborder également le cas des requêtes non hétérogènes.

Les algorithmes inclus dans **A4**, doivent donc effectuer toutes les étapes nécessaire au traitement d'une requête. Ces étapes sont :

1. *Extraction* des complexes à partir des descriptions en LD contenues dans la Kbase réceptrice qui doit gérer la requête (notée Kb_r). L'extraction de telles représentations simpliciales se fera par un algorithme général, que nous appelons A_X. Cet algorithme s'instanciera en sous-fonctions selon que l'on cherche à extraire un complexe de concepts, de rôles ou d'individus. Nous appelerons ces sous-fonctions respectivement $A_{X:concepts}$, $A_{X:rôles}$ et $A_{X:individus}$. Nous verrons que ces fonctions vont se diviser à leur tour en sous-fonctions selon le type de relations que l'on étudie pour les concepts, les rôles et les individus. Nous analysons cette étape dans la section VIII.2.
2. Gestion du *matching* du pattern de la requête avec les informations nécessaires de la Kbase Kb_r. Nous verrons que ce matching peut s'opérer soit sur les données explicites de la Kb_r, soit sur des données déduites de KB_r. En effet, cette Kbase peut contenir des informations non explicitées nécessaires à la production d'une substitution valide du pattern de la requête. Nous analysons cette étape dans la section VIII.3.
3. Application de l'agorithme *d'adaptation* pour donner une interprétation aux termes hétérogènes et ramener une requête avec des termes hétérogènes, à une requête avec uniquement des termes connus et procéder au matching selon l'étape ci-dessus. L'algorithme d'adaptation que nous utilisons est celui du système ESQIMO présenté au chapitre VI, que nous devrons néanmoins modifier pour l'utiliser dans notre problème d'hétérogénéité entre agents.

VIII.1.1.5 Substitution valide et production d'une réponse

Une fois la représentation simpliciale correspondant au type du pattern produite, nous pouvons l'utiliser pour trouver un matching et donc produire une substitution valide pour la requête.

Cette étape passe par la traduction des opérations de matching sur des expressions de LD en une recherche sur le complexe d'une structure topologique correspondant à la requête. Plus précisément, pour chaque structure de requête et pour chaque type de requête, il correspond la recherche d'une sous-structure topologique dans la représentation simpliciale étudiée (comme nous le verrons dans la section VIII.3).

La recherche de telles sous-structures se fera à l'aide d'un autre algorithme, que nous appelons $A_{\div}$. Encore une fois, $A_{\div}$ va se décomposer en plusieurs sous-fonctions selon le type du pattern à matcher et la relation représentée simplicialement. La réponse fournie par cet algorithme pourra alors être reprise par une autre fonction A_R qui produira la réponse sur la base des substitutions valides pour la requête fournies par $A_{\div}$.

VIII.2 Extraction de complexes à partir d'une base de connaissances en LD

Nous nous concentrons, dans cette section, sur l'illustration des processus *d'extraction* par la fonction A_X de complexes simpliciaux à partir des expressions de LD contenues dans les bases de connaissances en interaction, ce qui constitue une application directe du chapitre VII.

VIII.2.1 Les différentes fonctions d'extraction

Dans une terminologie appartenant à une base de connaissances, nous distinguons deux types de descriptions : les inclusions de concepts, introduites avec le symbole $\leq$; et les équations de concepts, introduites avec le symbole $\doteq$. Pour ce qui concerne la Abox associée, nous avons la relation d'instanciation, notée $\in$.

Les concepts peuvent être introduits soit par une inclusion, soit par une équation. Les rôles aussi en théorie, mais dans le langage de LD simple que nous utilisons dans ce document, il n'y a pas de constructeurs de rôles, et nous aurons tout au plus des relations d'inclusion entre rôles. Enfin, les individus ne pourront être introduits que par des relations d'instanciation dans la Abox. Ainsi, la fonction A_X se décomposera en :

- $A_{X:concepts:\leq}$: extrait une représentation simpliciale à partir des relations d'inclusion $\leq$ de la Tbox, pour le traitement d'un pattern de type concept (noté $?X$) ;
- $A_{X:concepts:\doteq}$: extrait une représentation simpliciale à partir des équations de concepts introduites par la relation $\doteq$ de la Tbox, pour le traitement d'un pattern de type concept $?X$;
- $A_{X:role}$: extrait une représentation simpliciale à partir des relations d'inclusion $\leq$ de la Tbox, pour le traitement d'un pattern de type role (noté $?r$) ;
- $A_{X:individu:\in}$: extrait une représentation simpliciale à partir des relations d'instanciation $\in$ de la Abox, pour le traitement d'un pattern de type individu (noté $?x$) ;

Par commodité et puisque toute confusion est impossible, nous noterons dans la suite ces quatre fonctions respectivement : $A_{X:\leq}$, $A_{X:\doteq}$, A_{role}, $A_{\in}$.

VIII.2.2 Extraction des complexes d'inclusion par $A_{X:\leq}$.

Pour extraire les complexes simpliciaux représentant les concepts reliés par la relation d'inclusion de concepts dans une Tbox, nous devons commencer par isoler les descriptions qui introduisent des inclusions. Nous travaillerons donc ensuite sur une sous-partie de la Tbox : le système de descriptions d'inclusions. Pour une Kbase KB_i, nous notons sa Tbox Tb_i et nous noterons son système de descriptions d'inclusion Inc_i.

Si nous reprenons l'exemple abstrait du chapitre III, avec les Tboxes envoyeur Tb_e et receveur Tb_r suivantes :

$$
\begin{array}{ll|ll}
Tb_e = & \{A \doteq E \wedge C & Tb_r = & \{F \doteq \forall s.C \\
 & D \doteq \forall r.C & & G \leq C \\
 & B \leq E & & E \leq G \wedge F \\
 & A \leq C\} & & A \doteq B \wedge C \\
 & & & A \leq C\}
\end{array}
$$

Nous travaillerons sur les systèmes de descriptions :

$$
\begin{array}{l|l||l|l}
Incl_e = & B \leq E & Incl_r = & G \leq C \\
 & A \leq C & & E \leq G \wedge F \\
 & & & A \leq C
\end{array}
$$

Nous nous appliquons maintenant à construire la représentation simpliciale correspondant à chacun de ces sytèmes, nous les appelons CS_e et CS_r. Rappelons que nous construisons *une* représentation CS_i pour un *système* de descriptions. Chaque nouvelle description ajoute donc de l'information à la même représentation.

VIII.2.2.1 La relation d'inclusion $\leq$.

Pour proposer un algorithme général de $A_{X:\leq}$, nous devons commencer par étudier les relations d'inclusion de concepts plus en détail, et en particulier nous allons examiner quelles sont les configurations simpliciales qui correspondent à une situation d'inclusion donnée. Pour cela, reprenons le formalisme de représentation simpliciale présenté au chapitre V où nous avions proposé de représenter les hiérarchies de classes par le biais de leur caractérisation par des *traits*.

VIII.2.2.1.a Hiérarchies de classes Nous avons vu au chapitre V, que puisque les complexes sont une généralisation des graphes à plusieurs dimensions, nous pouvons représenter des hiérarchies de classes à l'aide de CS. Pour cela, nous avons appuyé notre discours sur une correspondance avec les langages orientés objets en informatique :

- Les objets sont définis avec des attributs qu'ils héritent de leurs parents, et des attributs qui leur sont propres.

– Une classe est alors un ensemble d'objets qui partagent un ensemble d'attributs.
– La représentation d'une classe se fera tout naturellement par le biais de la représentation des attributs qui la caractérisent.
– La relation d'héritage des classes est alors la relation d'inclusion des complexes.

VIII.2.2.1.b Inclusions de concepts Afin d'appliquer ce point de vue aux concepts introduits dans une terminologies par le biais de la relation d'inclusion, nous proposons de :

– Voir un concept comme une classe dans une hiérarchie objet ;
– Un concept est donc représentable par le biais des attributs qui le caractérisent en tant que classe
– La relation d'inclusion des concepts est la relation d'inclusion des simplexes inversée.

Précisons ce dernier point. Soit un concept C_1 défini par un seul attribut, il sera représenté par un 0-simplexe S_{C_1}. Soit un concept C_2 inclu dans C_1 (héritant de celui-ci), sera spécialisé par rapport à C_1 par un deuxième attribut. Sa représentation simpliciale sera donc un 1-simplexe noté S_{C_2}. Le simplexe S_{C_2} contiendra le simplexe S_{C_1} alors qu'en termes d'inclusion, nous avons l'inverse : $C_2 \leq C_1$.

Ainsi, de même que nous avions noté au chapitre V que les classes les plus hautes dans un graphe d'héritage ont un représentation simpliciale contenue dans celles de toutes les autres, nous savons qu'un concept C_2 inclus dans un concept C_1 aura une représentation simpliciale S_{C_2} contenant celle de C_1 : S_{C_1}.

VIII.2.2.2 Algorithme de $A_{X:\leq}$

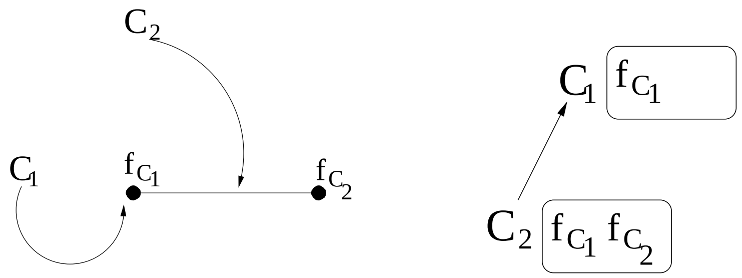
Mais comment établir automatiquement la représentation simpliciale de concepts en relation d'inclusion si nous ne disposons pas des attributs qui les caractérisent ? En particulier, le langage de LD que nous utilisons dans tous nos exemples n'utilise pas les attributs (alors que c'est possible dans plusieurs LD).

VIII.2.2.2.a Définition Nous appellons *trait*, pour le distinguer de la notion *d'attribut*, un attribut abstrait qui permet de caractériser un concept et en particulier de le spécialiser par rapport aux concepts dans lesquels il est inclu, on le note f_i (f pour *feature*). Il est dit abstrait pour deux raisons :

1. Nous ne cherchons pas à déterminer la valeur de tels attributs, seule leur *existence* est utilisée.
2. De plus, il se peut qu'ils ne soient pas uniques, c'est-à-dire que plus d'un attribut permette de spécialiser un concept par rapport à un autre, mais nous symboliserons un tel ensemble éventuel d'attributs par un seul symbole d'attribut.

Cette notion de trait est issue de la terminologie objet.

FIG. VIII.2 – Représentation simpliciale du système E_1, ainsi que la représentation sous forme de graphe de cette inclusion prise comme une hiérarchie de classes.



VIII.2.2.2.b Premier exemple Soit le système E_1 d'une seule équation, où le concept C_2 est inclus dans un concept C_1 :

$$E_1 = \{C_2 \leq C_1\}$$

nous dirons que le concept C_1 est caractérisé par l'attribut f_{C_1}, qui est un attribut abstrait associé à C_1. Le concept C_2 possède forcément ce trait et en est spécifié par un trait supplémentaire, disons f_{C_2}.

Ainsi, pour représenter la relation d'inclusion entre ces deux concepts, nous pouvons dresser la matrice d'incidence :

	f_{C_1}	f_{C_2}
C_1	1	0
C_2	1	1

Pour laquelle, la représentation simpliciale est construite en représentant les traits comme des sommets et les concepts comme des simplexes s'appuyant dessus, cette représentation est donnée par la figure VIII.2.

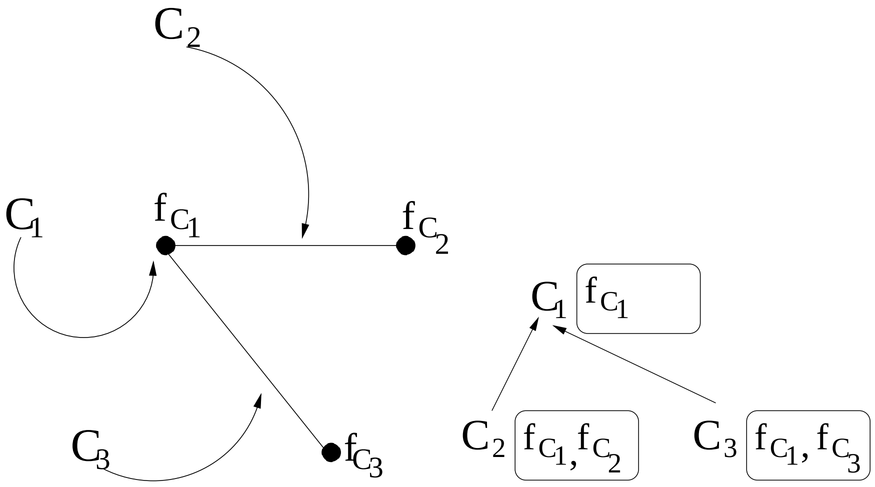
VIII.2.2.2.c Deuxième exemple Si un autre concept C_3, différent de C_2 est lui-aussi inclus dans C_1, on obtient alors le système E_2 :

$$\begin{cases} C_3 \leq C_1 \\ C_3 \neq C_2 \end{cases}$$

nous le spécifierons par le trait f_{C_3} en plus du trait f_{C_1} dont il hérite (par inclusion). La représentation sera un autre 1-simplexe appuyé sur f_{C_1} mais différent de celui représentant C_2 (voir figure VIII.3).

VIII.2.2.2.d Troisième exemple Si par contre ce dernier concept C_3 est maintenant inclus dans C_2, le système devient :

FIG. VIII.3 – Représentation simpliciale du système E_2 contenant un troisième concept, tel que $C_3 \leq C_1$.



$$E_3 = \{C_3 \leq C_2$$

alors il est spécifié par f_{C_3} mais hérite des deux traits f_{C_1} et f_{C_2}, il sera alors représenté par une face (voir figure VIII.4) prenant appui sur ces trois traits.

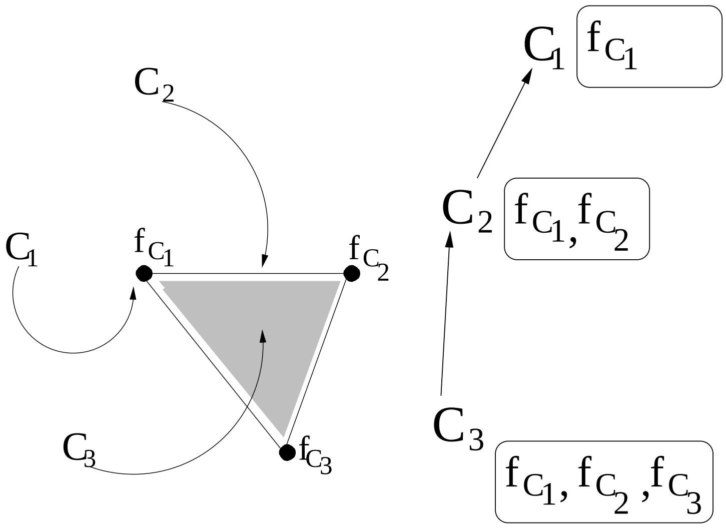
Remarque : Cette dernière représentation fait déjà intervenir une face de dimension 2 et présente donc une structure plus *riche* topologiquement que les 1-simplexes obtenus dans les deux exemples précédents. Le formalisme des graphes aurait alors des difficultés à représenter cette notion.

VIII.2.2.3 Procédure d'extraction automatique des complexes d'inclusion

Afin d'automatiser le processus de construction de représentations simpliciales d'inclusion à l'aide de la notion de traits, voici l'algorithme $A_{X:\leq:traits}$ qui constitue les listes de traits associées à chaque concept de Inc_i :

$A_{X:\leq:traits}$:
Pour une base de connaissances KB_o :
    Pour chaque description D_i introduite par $\leq$:
        Pour chaque concept C_i à gauche du $\leq$:
            Ajouter D_i à la liste $Incl_o$
            Associer une liste de traits l_i au concept C_i,
            Initialiser l_i avec le trait f_i ;
            Pour chaque description D_j déjà dans Inc_o :
                Mettre à jour la liste l_i
                Mettre à jour la liste l_j

FIG. VIII.4 – Représentation simpliciale du système E_3, tel que $C_3 \leq C_2$.



Une fois ces listes constituées, elle serviront à constituer la matrice d'incidence *concepts* $\times$ *traits*, c'est-à-dire $C_i \times f_i$ qu'utilise la procédure de représentation simpliciale par catégorisation, présenté au chapitre V. Nous effectuons maintenant cette procédure de représentation à l'aide de l'algorithme A_{cs}, qui représente une relation λ entre deux ensembles A et B :

> A_{cs} :
> Soient A et B deux ensembles
> Pour chaque élément a de A,
>   Représenter un sommet σ_a distinct,
> Pour chaque élément b de B,
>   Pour chaque élément a_b de A avec le quel il est en
>   relation par λ $((a,b) \in \lambda)$.
>   étiqueter le sommet σ_{a_b} par b
> Former les simplexes S_b de chaque élément b de B,
> avec tous les sommets étiquetés de leur nom

Ainsi la fonction $A_{X:\leq}$se ramène à l'appel successif de $A_{X:\leq:traits}$ et de A_{cs} :

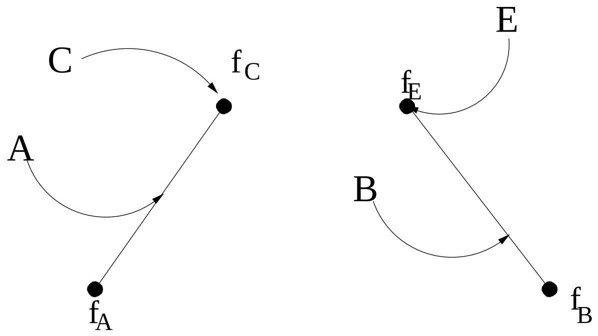
$$A_{X:\leq} = A_{X:\leq:traits} \circ A_{cs}$$

VIII.2.2.4 Illustration de l'algorithme $A_{X:\leq}$

Reprenons l'exemple de la page 190 :

$$\begin{array}{ll|ll} Incl_e = & B \leq E & Incl_r = & G \leq C \\ & A \leq C & & E \leq G \wedge F \\ & & & A \leq C \end{array}$$

FIG. VIII.5 – Représentation simpliciale des relations d'inclusion du système $Incl_e$.



VIII.2.2.4.a Le Système $Incl_e$ En appliquant l'algorithme $A_{X:\leq:traits}$ à $Incl_e$, on obtient les listes de traits :

$$\left| \begin{array}{l} E : \{f_E\} \\ B : \{f_E; f_B\} \\ C : [f_C\} \\ A : [f_C; f_A\} \end{array} \right.$$

Qui mène à la matrice d'incidence suivante :

	f_E	f_B	f_C	f_A
E	1	0	0	0
B	1	1	0	0
C	0	0	1	0
A	0	0	1	1

En appliquant l'algorithme A_{cs} à cette matrice, on obtient la représentation simpliciale :

$$\begin{array}{l} S_E = \langle f_E \rangle \\ S_B = \langle f_B, f_E \rangle \\ S_C = \langle f_C \rangle \\ S_{A=} \langle f_A, f_C \rangle \end{array}$$

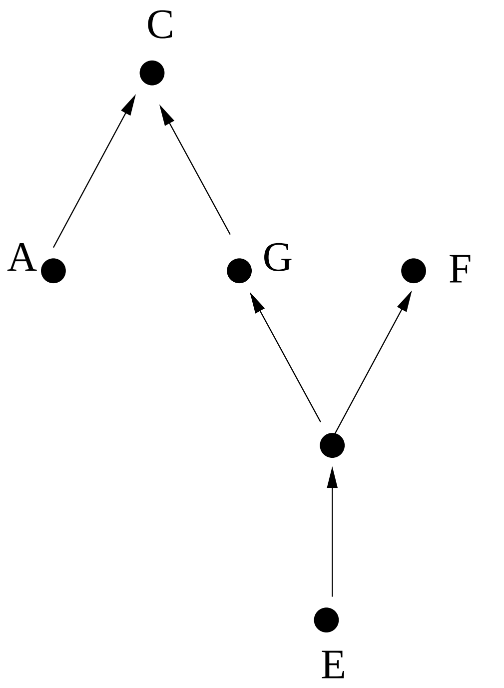
La représentation simpliciale d'inclusion extraite de $Incl_e$ se réduit donc à deux 1-simplexes qui correspondent au graphe d'héritage inversé et déconnecté, voir la figure VIII.5.

VIII.2.2.4.b Le Système $Incl_r$ Nous nous attendons à ce que le complexe représentant les relations d'inclusion de $Incl_r$ soit donc formé d'une seule composante connexe puisque tous les éléments appartenant à des relations d'inclusion sont liés entre eux dans cette sous-Tbox. Le graphe d'héritage des classes correspondant aux concepts en inclusion, est donné par la figure VIII.6.

Nous détaillons les étapes d'application de l'algorithme $A_{X:\leq:traits}$ appliqué à Inc_r :

– Soit le concept C caractérisé par le trait f_C ;

FIG. VIII.6 – Graphe d'héritage correspondant aux relations d'inclusion entre concepts du système $Incl_r$.



- Le concept G est inclus dans C, il a donc lui aussi ce trait f_C mais il se différencie du concept C disons par au moins le trait f_G ;
- Le concept F est caractérisé par le trait f_F ;
- Le concept $F \wedge G$ est inclus dans F et dans G par définition de la conjonction d'ensembles, ainsi ce concept aura les traits de F c'est à dire f_F , ceux de G c'est-à-dire f_C et f_G .
- Enfin, le concept F est inclus dans $F \wedge G$ et aura les traits f_C, f_G, f_F, et un trait propre, appelé f_E.

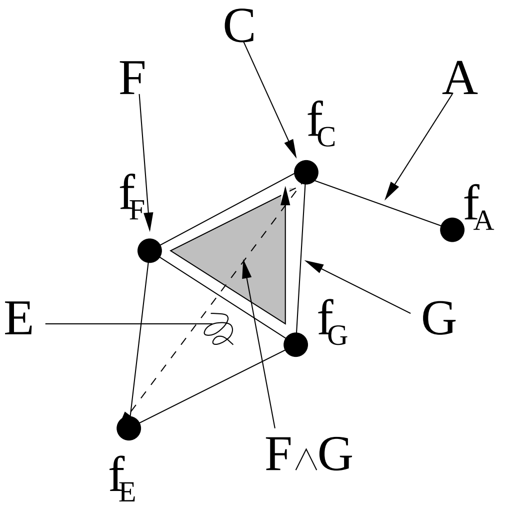
Toutes ces définitions nous mènent à la matrice d'incidence suivante pour la relation d'inclusion :

$\leq$	f_C	f_G	f_F	f_E
C	1	0	0	0
G	1	1	0	0
F	0	0	1	0
$F \wedge G$	1	1	1	0
E	1	1	1	1

qui se représente par les simplexes de la figure VIII.7 :

$$\begin{array}{l} S_C = \langle f_C \rangle \\ S_G = \langle f_C, f_G \rangle \\ S_F = \langle f_F \rangle \\ S_{F \wedge G} = \langle f_C, f_F, f_G \rangle \\ S_E = \langle f_C, f_G, f_F, f_E \rangle \end{array}$$

FIG. VIII.7 – Représentation simpliciale de la relation d'inclusion de la Tbox Tb_r. Afin d'alléger les notations, nous avons désigné directement par le nom du concept la représentation simpliciale de chaque concept. C'est-à-dire qu'au lieu d'indiquer le simplexe S_A représentant A, nous l'avons directement fléché avec l'étiquette A.



VIII.2.2.5 Utilisation de constructeurs dans les relations d'inclusion

Notons que des constructeurs de concepts peuvent intervenir dans les relations d'inclusion, comme par exemple dans la description $E \leq F \wedge G$ de $Incl_r$ dans l'exemple précédent. Dans ce cas, nous avons alors affaire à un concept composé. Qu'un concept soit composé ou atomique, il est cependant traité par $A_{X:\leq:traits}$ comme un symbole atomique auquel il associe simplement une liste de traits. En effet, toujours dans l'exemple précédent, le concept composé $F \wedge G$ est pris comme un symbole atomique, il est à ce titre représenté comme un simplexe et a une place identique à celle des autres concepts (réellement atomiques) dans la matrice d'incidence.

Mais les choses ne sont pas si simples. En effet, dans l'exemple ci-dessus, pour déterminer la liste des traits de $F \wedge G$ nous avons utilisé la connaissance :

$$F \wedge G \leq F \text{ et } F \wedge G \leq G.$$

Cette connaissance *implicite* découle naturellement de la définition de l'intersection de concepts $\wedge$ en LD, mais nous ne l'avons pas fournie *explicitement* à $A_{X:\leq:traits}$! Il faut donc faire appel à un traitement préalable lorsque nous devons traiter des concepts composés. En particulier, remarquons également que non seulement le concept $F \wedge G$ a les traits f_F et f_G découlant de F et G dans lesquels il est inclus, mais qu'il n'en possède pas d'autres dans sa liste, pour le spécialiser.

Afin de clarifier toutes ces connaissances liées aux connecteurs de LD, nous devons donc ajouter une étape de traitement qui affine l'algorithme d'extraction $A_{X:\leq:traits}$:

1. Si le concept à représenter est atomique, alors on lui associe un trait f_i abstrait pour le *spécifier*.
2. Si il ne l'est pas, alors on cherche à déduire les traits qui lui sont associés en fonction des traits de ses composants.

VIII.2.3 Extraction des complexes d'équation avec $A_{\doteq}$

Dans les relations de définition utilisant le symbole $\doteq$, nous pouvons rencontrer des constructeurs de deux sortes :

1. Les connecteurs logiques de la logique du premier ordre : en particulier, dans le langage AL (cf. chapitre II) que nous utilisons, nous ne disposons que de la conjonction $\wedge$, et de la négation $\neg$ (en particulier, on ne traite pas le connecteur $\vee$, de disjonction) ;
2. Les restrictions : de nombre $\leq n$ et $\geq n$, ou de valeur $\forall r : C$.

Nous allons donner une procédure d'extraction de complexes simpliciaux pour chacun de ces cas de définitions.

VIII.2.3.1 Les connecteurs logiques

VIII.2.3.1.a Traitement de la conjonction Soit, par exemple, le concept A défini par conjonction :

$$A \doteq E \wedge C$$

cette équation peut se ramener au traitement de l'intersection dans les relations d'inclusion. En effet, la représentation simpliciale S_A du concept A est alors le simplexe qui englobe la représentation simpliciale S_E de E et la représentation S_C de C. Ainsi, nous pouvons exploiter l'algorithme d'inclusion. Pour cela, nous déduisons de l'équation précédente le système suivant de deux inclusions :

$$\begin{cases} A \leq C \\ A \leq E \end{cases}$$

sur lesquelles il est aisé d'appliquer $A_{X:\leq:traits}$. Cet algorithme donne en sortie les listes de traits suivantes (cf. VIII.2.2.3) :

$$\begin{array}{c} S_E = \langle f_e \rangle \\ S_C = \langle f_C \rangle \\ S_A = \langle f_E, f_C \rangle \end{array}$$

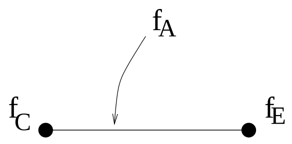
qui à leur tour donnent la représentation simpliciale de la figure VIII.8, par application de A_{cs} (voir page 194).

Remarque : Notons qu'au chapitre VII, nous avions proposé une interprétation de la conjonction sur la représentation simpliciale des concepts en *extension*. Dans le chapitre VII, dès lors que nous nous intéressions aux concepts en extension, une conjonction de concepts se représentait par une *frontière* entre deux complexes en extension. Cela correspondait à l'intersection ensembliste de deux extensions de concepts dans l'interprétation ensembliste de la LD.

Mais ici, nous représentons les concepts à l'aide de l'algorithme d'inclusion, nous sommes en train de représenter les *relations* des concepts *entre eux*, sans entrer dans leur contenu (extension).

Nous présenterons l'extraction de la représentation simpliciale des concepts et extension dans la section VIII.2.4.1, page 201.

FIG. VIII.8 – Représentation simpliciale d'un concept défini par conjonction $A \doteq E \wedge C$.



VIII.2.3.1.b Traitement de la négation Soit, par exemple, le concept A défini par :

$$A \doteq \neg B$$

D'après ce que nous avons vu au chapitre VII, nous pouvons construire le simplexe S_A représentant le concept A, comme le complément de S_B, représentant B dans un domaine Dom. Mais se pose alors la question de savoir représenter ce domaine Dom, et surtout à partir de quelles connaissances.

Remarque : Nous avons vu au chapitre VII, que parler de domaine, nous oblige à introduire les représentations en *extension* des domaines, c'est-à-dire de proposer une *interprétation* en termes de complexes simpliciaux. Nous verrons comment extraire les complexes qui représentent les extensions lorsque nous parlerons de la relation d'instanciation et des Abox, au paragraphe VIII.2.4.1.

VIII.2.3.2 Les restrictions

VIII.2.3.2.a Restriction de valeur $A_{x:\doteq:\forall}$ Reprenons l'exemple VIII.2.2, page 190, considérons le concept D introduit par l'équation :

$$D \doteq \forall r : C$$

ce concept dénote l'ensemble des individus x tels que s'ils entretiennent un lien de rôle r avec un autre individu y, ce dernier appartient obligatoirement au concept C (voir chapitre II) :

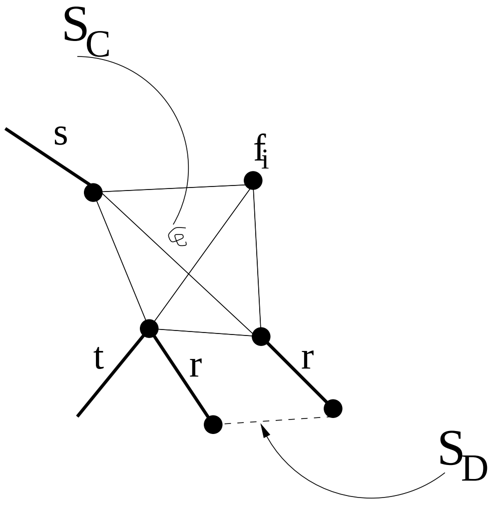
$$I(D) = \{x | \forall y, (x, y) \in I(r) \Rightarrow y \in I(C)\}$$

Où I désigne la fonction d'interprétation (ensembliste) introduite au chapitre II. Quelle est alors la représentation simpliciale correspondante ? Si on note S_r la représentation simpliciale du rôle r et S_C, la représentation simpliciale du concept C, alors la représentation simpliciale de D est donnée par :

$$S_D = \langle f_i | \forall f_j \langle f_i, f_j \rangle \in S_r \Rightarrow f_j \in S_C \rangle$$

ce qui se traduit par le simplexe des sommets liés à des sommets du simplexe S_c via des arcs représentant le rôle r.

FIG. VIII.9 – Exemple de représentation simpliciale d'une équation introduisant une restriction de valeur avec $D \doteq \forall r : C$. Ici le simplexe représentant le concept C en extension est un tétraèdre S_C duquel partent des arcs qui représentent des liens de rôle. Les rôles représentés sont r, s et t. L'ensemble des sommets liés à des sommets de S_C par des arcs r constitue l'extension S_D du concept D.



Remarque : Dans cette construction, nous supposons encore une fois que l'on travaille sur la représentation simpliciale des concepts en extension sur lesquels sont reportés les arcs de rôle liant des instances entre elles.

Nous présenterons l'extraction de la représentation simpliciale des concepts en extension dans la section VIII.2.4.1, page 201. Considérons tout de même la configuration de la figure VIII.9, par exemple.

Pour extraire automatiquement de telles représentations, on peut proposer d'explorer le simplexe S_C représentant le concept C et de suivre les liens r qui en partent, voici $A_{X:\doteq:\forall}$:

> $A_{X:\doteq:\forall}$:
> `Pour tout sommet` s `dans` S_C
>   `Si il y a une face étiquetée` r `qui en part,`
>   `Inclure le premier sommet le long de ce chemin dans la`
>   `liste` Sol
> `La représentation simpliciale de` $\forall r : C$ `est alors` $\langle Sol \rangle$

Le simplexe solution $\langle Sol \rangle$ est le simplexe formé de tous les sommets qui sont à une distance de 1 de S_C si l'on considère uniquement S_r (arcs r) dans S_C.

Remarque : Dans ce cas, nous nous appuyons encore une fois sur la représentation simpliciale des concepts en extension. Mais nous avons également besoin de la représentation des rôles sur ces extensions. Nous verrons comment extraire de telles représentations dans la section VIII.2.4.2, page 202.

VIII.2.3.2.b Les restrictions de nombre $A_{X:\doteq:n}$ Soit, par exemple, le concept introduit par l'équation suivante :

$$A \doteq B \wedge (\geq 3.r)$$

qui dénote l'ensemble des individus x tels qu'ils appartiennent au concept B et entretiennent au moins 3 liens de rôle r :

$$I(A) = \{x | x \in I(B) \wedge ||y|(x,y) \in I(r)|| \geq 3\}$$

Cette fois, la représentation simpliciale S_A du concept A est définie par le nombre de liens de rôles étiquetés r qu'elle possède. Encore une fois, cette opération s'effectue sur la représentation simpliciale des concepts en extension (dont la construction est abordée ci-dessous). L'algorithme de représentation des restrictions de rôle $A_{X:\doteq:n}$, est le suivant :

```
A_{X:≐:n} :
Pour chaque sommet σ_i de la représentation en extension
   Vérifier si des 1-simplexes α_j partent de ce sommet
      Si aucun arc, terminer.
      Sinon, tant qu'il y a un arc α_j non examiné,
         Vérifier s'il est étiqueté par un rôle r
         Si pas d'étiquette de rôle, terminer.
         Sinon Passer à l'arc α_{j+1} suivant
   Mettre à jour les compteurs c_r^i de role de chaque r pour ce sommet σ_i
   Former la liste list des indices i des compteur c_r^i égaux à n
     Le simplexe solution s'appuie sur les sommets σ_i avec i ∈ list
```

VIII.2.4 Extraction des complexes d'extension par A_{role} et $A_{\in}$

VIII.2.4.1 Les complexes d'instanciation de concepts

Les complexes d'instanciation sont construits à partir des assertions de la Abox. En effet, ils correspondent à donner une interprétation simpliciale de formules de LD, c'est-à-dire à représenter les concepts décrits par ces formules de LD en extension.

Comme nous l'avons vu au chapitre VII, pour la représentation des prédicats du premier ordre, à partir de la modélisation des individus, nous pouvons dessiner les domaines d'extension pour un prédicat unaire. Ainsi, si les individus sont des sommets, un concept (qui est un prédicat unaire), est alors représenté comme un simplexe qui s'appuie sur les sommets qui représentent ses instances.

L'algorithme de construction des complexes d'extension, $A_{\in}$, est donc l'application de l'algorithme de base A_{cs}, au produit des ensembles : (instances $\times$ concepts effectifs), déclarés dans la Abox Ab_r ; et en prenant pour la relation binaire λ, la relation d'instanciation $\in$:

```
A∈ :
Soit A l'ensemble des instances de Ab_r
Soit B l'ensemble des concepts effectifs de Ab_r
Pour chaque instance a ∈ A,
   Représenter un sommet σ_a distinct,
Pour chaque concept b de B,
   Pour chaque instance a_b du concept b
      étiqueter le sommet σ_{a_b} par b
Former les simplexes S_b de chaque concept b de B,
avec tous les sommets étiquetés de leur nom.
```

Remarque : Avec les complexes simpliciaux, nous disposons de simplexes de dimension arbitraire, pour représenter des domaines aussi grands que désiré. Par exemple, le domaine d'un prédicat $C(x)$ tel que $D_C = \{a, b, h, c\}$ sera modélisé, par un tétraèdre, alors qu'il ne pourrait être modélisé, avec des ensembles classiques, que comme un plus gros diagramme de Venn contenant quatre croix.

VIII.2.4.2 Les complexes de rôles

Ce que nous pouvons représenter à propos d'une rôle donné est plus limité que ce que nous pouvons représenter à propos d'un concept :

1. Avec le langage simple de LD, AL, que nous utilisons, il n'y a pas de constructeur de rôles. Ainsi, nous pouvons introduire des rôles en écrivant des relations *d'inclusion* à leur sujet. dans ce cas, nous les représentons par un complexe à part, en utilisant l'algorithme déjà illustré $A_{X:\leq}$ (voir page 193).
2. Les autres informations que nous pouvons extraire à propos des rôles, sont plus *indirectes*. Elles concernent les liens entre des concepts qui sont fournis dans les définitions de concepts. Par exemple, dans une restriction de valeur, du type de l'exemple précédent :

$$D \doteq \forall r : C$$

Nous avons vu dans le chapitre VII que dans ce second cas, nous pouvions représenter le concept D à partir des représentations en extension du concept C. La représentation simpliciale de concepts en extension fait intervenir les assertions de la Abox. On peut alors étiqueter certains arcs par le nom du rôle r si elle représentent le lien de rôle qui existe entre deux instances de concepts liées par définition.

Par exemple, si nous avons dans la Abox les assertions suivantes :

$$(dodo \in D), (coco \in C)$$

nous savons par la définition de D ci-dessus qu'un lien de type r les lie selon :

$$(dodo, coco) \in r$$

ainsi, si l'on représente *dodo* et *coco* par des sommets, on peut tendre un arc (simplexe de dimension 1) entre ces deux sommets et l'étiqueter r. La représentation simpliciale du rôle r sur ce complexe d'extension sera l'union des faces étiquetées r lorsque l'on a représenté toutes les assertions de la Abox :

$$S_r = \langle \langle dodo, coco \rangle, ... \rangle$$

L'algorithme d'extraction des complexes de rôles, A_{role} est alors :

A_{role} :
Construire la représentation en extension $S_{ext}(Ab_r)$ de Ab_r avec $A_\in$
Pour chacun des concepts effectifs C de Ab_r,
Si (C est défini dans Tb_r à l'aide d'une restriction de nombre
sur le rôle r :
Et
l'instance c_o liée par ce role r est nommée dans Ab_r)
Alors mettre à jour un arc Arc_r dans le complexe d'extension $S_{ext}(Ab_r)$.
Mettre à jour la représentation S_r du rôle r.

VIII.2.5 Conclusion sur les extractions de complexes

Nous venons de revoir toutes les représentations simpliciales que l'on peut extraire d'une Kbase simple écrite dans la langage de LD AL, selon le chapitre VII. Ces représentations se divisent en deux sortes :

- Celles qui représentent des liens entre les concepts : par exemple de subsomption, avec le complexe d'inclusion ;
- Celles qui en représentent les extensions : en d'autres termes, une interprétation des formules de LD de la Kbase.

Comment pouvons nous à présent exploiter ces représentations simpliciales pour le traitement des requêtes hétérogènes entre deux Kbases écrites en LD ? Pour répondre à cette interrogation, nous allons commencer par nous intéresser au traitement d'une requête non hétérogène. Nous verrons quels sont les algorithmes qui permettent de produire une substitution valide pour chaque type de pattern. Nous verrons également que l'on peut utiliser les représentations simpliciales même pour le traitement des requêtes non-hétérogènes, pour "lire" ces substitutions, au lieu de procéder à un matching sur les expressions en LD.

Ce n'est que dans la section suivante, que nous augmenterons ces étapes de traitement pour inclure finalement l'algorithme d'adaptation, utile au traitement des requêtes contenant des termes hétérogènes.

VIII.3 Gestion d'une requête non hétérogène

Nous allons nous intéresser dans cette section, au traitement d'une requête non hétérogène, étape par étape, à l'aide des :

TAB. VIII.1 – Sémantique intuitive de toutes les requêtes qu'il est possible de poser sur les concepts d'une Tbox. ??, avec A désignant un concept constant.

Requête en LD	Sens intuitif
$?X(A \leq X)$	Quel concept X subsume A ?
$?X(X \leq A)$	Quel concept X est subsumé par A ?
$?X, ?Y(X \leq Y)$	Quels sont tous les concepts en relation d'inclusion ?
$?X(A \doteq X)$	Quelle est la définition de A ?
$?X(X \doteq A)$	Quel concept X est défini par A ?
$?X, ?Y(X \doteq Y)$	Quels sont tous les concepts définis par équation ?

- Expressions de LD, déjà présentes dans la Kbase ;
- Représentations simpliciales, extraites à l'aide des algorithmes $A_{\leq}$, $A_{\doteq}$, $A_{\in}$ et A_{role} , et présentées dans la section précédente.

VIII.3.1 Toutes les requêtes qu'il est possible de poser à une Kbase en LD

Nous commençons par analyser toutes les requêtes qu'il est possible de poser à une Kbase écrite en LD. Pour cela, nous rappelons la formation syntaxique des requêtes, nous en donnons un sens intuitif, puis nous cherchons à quelles opérations simpliciales sur les représentations simpliciales extraites les requêtes correspondent.

VIII.3.1.1 Rappels de syntaxe

Dans le domaine des bases de connaissances en logique des descriptions, les requêtes sont des expressions syntaxiquement correctes de LD où l'on remplace certains atomes par des patterns typés. Nous avons déjà vu au chapitre III, que ces patterns peuvent porter sur les types suivants (ceux autorisés par la syntaxe de LD) :

- concept, noté $?X$;
- rôle, noté $?r$;
- individu, noté $?x$.

Pour une syntaxe *complète* des requêtes autorisées par le langage simple de LD AL, se reporter au chapitre III.

VIII.3.1.2 Sémantique intuitive des requêtes

A chacune des formules de requête possibles, il correspond une question qui en représente la sémantique intuitive. La transposition en français de ces questions est donnée par la table VIII.1 pour les questions faisant intervenir des connaissances contenues dans une Tbox, et par la table VIII.2 pour les questions sur les extensions et les rôles, faisant intervenir des connaissances contenues dans une Abox.

TAB. VIII.2 – Sémantique intuitive de toutes les requêtes qu'il est possible de poser à propos des individus et des rôles d'une Abox, avec a désignant un individu constant, C un concept constant et r_0 un rôle constant.

Requête en LD	Sens intuitif
$?x\ C(x)$	Quels individus x sont des instances de C ?
$?X\ X(a)$	De quel concept X, a est-elle une instance ?
$?x\ r_0(a,x)$	Quel individu x est lié à a par un rôle r_0 ?
$?x, ?y\ r_0(x,y)$	Quels individus x et y sont liés par le rôle r_0 ?
$?r(a,b)$	Quel rôle r relie a et b ?
$?r, ?x\ ?r(a,x)$	A quelle instance x, a est-elle reliée, et par quel rôle ?

VIII.3.1.3 Opérations simpliciales correspondantes

Nous venons de voir qu'à chacune des requêtes syntaxiquement correcte, correspond un sens intuitif. Répondre à chacune d'entre elles, revient ensuite à effectuer une opération sur Kb_r. Cette opération est une opération de *recherche* de l'information demandée dans la Kbase.

On peut simplement effectuer cette recherche par un *matching* syntaxique sur les formules de LD de Kb_r : dans ce cas, le traitement d'un requête se décompose en deux étapes :

1. Recherche de l'information correspondant au pattern par matching sur les expressions de LD de la Kbase ;
2. Utilisation des représentations simpliciales uniquement pour traiter les termes hétérogènes (le cas échéant) pour leur gestion par adaptation.

Mais ne peut-on pas utiliser les représentations simpliciales dès l'étape 1 ? Cela correspond à rechercher l'information correspondant aux patterns contenus dans les requêtes sur les représentations simpliciales ? A quelle *opération* sur les représentations simpliciales correspond alors chacune des requêtes qu'il est possible de poser ? C'est ce que nous allons étudier maintenant.

Pour cela, nous divisons les requêtes présentées dans la section VIII.3.1 en trois cas de figure :

- **a)** Les requêtes portant sur des concepts et faisant intervenir la relation d'inclusion : cela correspond aux trois premières lignes de la table VIII.1 ;
- **b)** Les requêtes portant sur les connaissances en extensiosn : cela correspond aux requêtes de la table VIII.2 ;
- **c)** Les requêtes portant sur des concepts et faisant intervenir la relation d'équation $\doteq$: cela correspond aux trois dernières lignes de la table VIII.1.

VIII.3.1.3.a Requêtes sur des concepts en relation d'inclusion Considérons, par exemple, la question :

"Quel concept X subsume A ?"

dont la requête en LD s'écrit :

$$?X(A \leq X)$$

Pour répondre en utilisant directement un matching sur les expressions en LD de Kb_r, il faut rechercher une formule de LD commençant par :

TAB. VIII.3 – Opérations simpliciales correspondant aux requêtes qu'il est possible de poser à propos des concepts d'une Tbox et faisant intervenir la relation d'inclusion de concepts $\leq$. Avec S_A désignant la représentation simpliciale *d'inclusion* du concept constant A (c'est-à-dire que toutes ces recherches s'effectuent sur le complexe extrait à partir des relations d'inclusion de la Tbox), et $\subseteq$ désignant la relation topologiqe d'inclusion entre simplexes. L'inclusion de simplexes est testée par l'algorithme $A_{\subseteq}$.

Requête en LD	Opération simpliciale correspondante
$?X(A \leq X)$	Rechercher S_X tel que $S_X \subseteq S_A$.
$?X(X \leq A)$	Rechercher S_X tel que $S_A \subseteq S_X$.
$?X, ?Y(X \leq Y)$	Lister les inclusions de simplexes.

$$A \leq$$

et donner comme substitution valide pour $?X$ la partie droite de la description sélectionnée.

Pour répondre en utilisant les représentations simpliciales, nous reprenons ce que nous avons dit à la section VIII.2.2.1.b, page 191, à propos de la relation d'inclusion et de sa correspondance en termes simpliciaux. Ainsi, si l'on utilise la représentation simpliciale d'inclusion, rechercher un concept qui *subsume* A revient à chercher un simplexe *contenu dans* le simplexe de A, c'est-à-dire :

Rechercher un simplexe S_X dont tous les sommets *et toutes les faces* soient dans S_A.

De cette manière, nous pouvons aussi trouver *tous* les simplexes qui subsument A en listant tous les simplexes contenus dans S_A, alors que sur la représentation en LD, il y a certaines inclusions qui sont implicites et découlent d'une déduction à partir des autres définitions (cas que nous illustrons dans les exemples de la section suivante).

L'algorithme $A_{\subseteq}$ qui permet de trouver un simplexe S_X inclus dans un autre simplexe S_A, il faut tester l'inclusion de toutes les faces de S_X dans S_A :

$A_{\subseteq}$:
Soit $\{S_X^+\}$ l'ensemble des faces de S_X
Soit $\{S_A^+\}$ l'ensemble des faces de S_A
Tant qu'il reste une face f_X de $\{S_X^+\}$ non examinée
  Si $f_X \in \{S_A^+\}$, passer à la suivante
  Sinon, terminer
Si toutes les faces sont testées avec succès, alors $S_X \subseteq S_A$

En généralisant cette analyse, nous donnons maintenant toutes les opérations simpliciales correspondant aux requêtes qu'il est possible de poser à propos des concepts dans une Tbox et faisant intervenir la relation d'inclusion dans la table VIII.3.

VIII.3.1.3.b Les requêtes sur les extensions En ce qui concerne les requêtes à propos des individus et des rôles, y répondre revient soit à utiliser les expressions contenues dans Ab_r, soit à effectuer des opérations simpliciales sur les représentations en extension (construites avec $A_{\in}$).

Rechercher un individu revient à chercher un sommet, et rechercher un rôle revient à chercher une étiquette d'arc (car c'est une relation binaire).

TAB. VIII.4 – Opérations simpliciales correspondant aux requêtes qu'il est possible de poser à propos des individus et des rôles d'une Abox. Ces recherches s'effectuent sur les représentations en extension des concepts.

Requête en LD	Opération simpliciale correspondante
$?x\ C(x)$	Rechercher les sommets x de S_C.
$?X\ X(a)$	Rechercher les simplexes S_X, qui ont a pour sommet.
$?x\ r_0(a, x)$	Rechercher les sommets x reliés à a par un arc r_0 (algo A_{arc}).
$?x, ?y\ r_0(x, y)$	Rechercher les couples de sommets x et y liés par un arc r_0 (algo A_{arc}).
$?r(a, b)$	Rechercher les arcs r qui relient les sommets a et b (algo A_{arc}).
$?r, ?x\ ?r(a, x)$	Rechercher les sommets x auxquels a est relié, et préciser par quel arc r (algo A_{arc}).

Par exemple, lorsque l'on pose la question :

"Quels individus x sont des instances de C ?"

on doit rechercher les instances du concept *effectif* C, cela correspond à la requête LD :

$$?x\ C(x)$$

Ce qui en représentation simpliciale se traduit par rechercher l'ensemble des sommets du complexe d'extension. Nous donnons les opérations simpliciales correspondant aux requêtes qu'il est possible de poser à une Abox sur la table VIII.4. L'algorithme qui donne l'ensemble des sommets d'un simplexe est trivial. En ce qui concerne la recherche sur les arcs, par A_{arc}, il faut procéder à une recherche de tous les 1-simplexes. Pour cela, il faut développer l'écriture des simplexes qui est sous forme maximale (voir chapitres 5 et 6), comme pour l'algorithme d'inclusions. Ainsi, A_{arc} recherche les arcs étiquetés d'un simplexe :

A_{arc} :
Soit $\{S_X^+\}$ l'ensemble des faces de S_X (son développement).
Tant qu'il reste une face f_X de $\{S_X^+\}$ non examinée
  Si ($dim(f_X) = 1$ Et qu'il existe $r =$ étiquette (f_X))
    ajouter (f_X, r) à la liste $list_{arcs}^X$ des arcs étiquetés de S_X.

VIII.3.1.3.c Les requêtes sur les équations Considérons maintenant la requête :

"Quelle est la définition de A ?"

dont la requête en LD s'écrit :

$$?X(A \doteq X)$$

Répondre à cette question, revient à chercher le concept X (simple ou composé) qui est à droite d'une équation commençant par :

$$A \doteq$$

Pour répondre à cette question en utilisant les représentations simpliciales, il faut utiliser les complexes d'équation dont nous avons parlé dans la section VIII.2.3, page 198 ce qui correspond à faire les recherches sur les extensions comme pour le cas précédent.

VIII.3.2 Traitement des requêtes

VIII.3.2.1 Deux cas de figures

Replaçons-nous dans le cadre de l'interaction entre deux Kbases. Les requêtes, provenant de KB_e, et posant des questions sur un concept à KB_r, peuvent ou non contenir des termes hétérogènes pour KB_r ; et nous savons que selon qu'elles contiennent des termes hétérogènes, KB_r aura besoin ou non d'utiliser une adaptation.

Mais même si les requêtes ne contiennent pas de termes hétérogènes, il se peut que l'information nécessaire pour répondre, c'est-à-dire être capable de produire une substitution valide du pattern, ne soit pas exprimée *explicitement* dans la Kbase. L'agent récepteur est alors obligé d'opérer un *raisonnement* simple pour déduire l'information qui lui manque mais qui est présente *implicitement*. Ainsi, lors de la réception d'une requête non hétérogène, KB_r peut être confrontée à deux cas de figure :

1. L'information nécessaire au matching est explicite dans la Kbase KB_r : il suffit alors d'effectuer un matching *direct* du pattern de la requête avec la Kbase ;
2. La connaissance nécessaire est implicite, il faut l'expliciter par un raisonnement : il est alors nécessaire d'effectuer un matching avec *déduction* ;

VIII.3.2.2 Résolution par matching direct

VIII.3.2.2.a Requête sur un concept Nous commençons par nous intéresser au cas le plus simple possible. Il s'agit d'une requête simple, portant sur un concept (pattern $?X$), faisant intervenir une relation d'inclusion, et dont l'information nécessaire pour répondre est présente explicitement dans KB_r : il suffit donc de procéder à un matching direct pour fournir une substitution valide de X.

Reprenons l'exemple de la page 190. Soit la requête de concept (pattern $?X$), appelée $Q^a_{e\to r}$, d'indice a, envoyée par KB_e à KB_r :

$$Q^a_{e\to r} = ?X(X \leq C)$$

Rappelons que c'est Tb_e qui *envoie* la requête et que l'expression de LD de cette requête est donc *consistante* avec cette Tbox. Ensuite, c'est Tb_r qui reçoit la requête et qui va la *gérer localement*, uniquement avec les connaissances contenues dans sa Tbox (voir hypothèses du chapitre III). Pour construire une réponse, Tb_r doit alors tenter de trouver une *interprétation* pour cette requête, puis effectuer une *substitution* du pattern $?X$ qu'elle contient avec un concept de sa base (car ce pattern est du type “concept”).

Remarque : La réponse résultant de cette substitution peut être unique ou pas. Une seule devra être sélectionnée, et en cas d'échec de l'interaction, les autres peuvent être stockées dans une pile et dépilées pour être utilisées dans la suite de l'interaction.

Dans notre exemple, l'expression $X \leq C$ de la requête $Q^a_{e\to r}$ peut être mise en correspondance par KB_r avec les deux descriptions de Tb_r suivantes :

1. $A \leq C$, une substitution valide pour répondre à $Q^a_{e\to r}$ sera alors $R^a_{r\to e} = (X \to A)$. Si Tb_e reçoit ensuite cette substitution, il sera “d'accord” avec Tb_r puisqu'il n'y a pas de contradiction avec sa propre description $A \leq C$.

TAB. VIII.5 – Aboxes Ab_e et Ab_r associées arbitrairement (pour les besoins de l'exemple) aux deux Tboxes Tb_e et Tb_r des agents KB_e et KB_r, de l'exemple page 190.

Ab_e	$\{c_o \in C$	Ab_r	$\{f_o \in F$
	$a_o \in A$		$g_o \in C$
	$d_o \in D$		$c_o \in C$
	$(d_o, c_o) \in r\}$		$(f_o, c_o) \in s\}$

2. $G \leq C$, une substitution valide pour répondre à $Q^a_{e \to r}$ sera alors $R^a_{r \to e} = (X \to G)$. Si Tb_e reçoit cette substitution, il pourrait en déduire que Tb_r "appelle" G le concept qui chez lui s'appelle A.

D'après la table VIII.1, cette requête correspond à l'opération simpliciale suivante, utilisant $A_{\subseteq}$:

"quel concept X est tel que sa représentation simpliciale S_X

est incluse dans celle de C, S_C $(S_X \subseteq S_C)$?"

c'est-à-dire que $A_{\subseteq}$doit être utilisé pour rechercher un S_X qui *vérifie* l'inclusion, cela suppose une production préalable de candidats. Parmi les simplexes extraits par $A_{\leq}$, ceux qui vérifient cette relation sont bien S_A et S_G (par construction).

VIII.3.2.2.b Requête sur un individu Reprenons les Tbox définies dans l'exemple VIII.2.2, et définissons les Abox Ab_e et Ab_r associées (voir table VIII.5).

Soit la nouvelle requête, d'indice b, posée par KB_e à KB_r à propos d'un individu :

$$Q^b_{e \to r} = ?x(x \in C)$$

D'après ce que nous avons dit dans la section précédente, cette requête correspond à la question :

"Quel individu est instance du concept C ?"

Pour construire une réponse, Tb_r doit alors tenter de trouver une *interprétation* pour cette requête, puis effectuer une *substitution* du pattern $?x$ qu'elle contient avec un individu de sa Kbase KB_r, et donc de son Abox Ab_r, puisque le pattern est du type "individu".

Dans notre exemple, l'expression $x \in C$ de la requête peut être mise en correspondance par KB_r avec :

$$c_o \in C, g_o \in C$$

Par un matching direct, une réponse possible consiste en la substitution valide suivante :

$$R^b_{r \to e} = (x \to c_o)$$

La Kbase KB_e connaissant l'individu c_o et la considérant elle aussi comme une instance du concept C, il sera "d'accord" avec la réponse.

Par contre, si la réponse porte plutôt sur g_o, il se peut que KB_e rejette la réponse auquel cas KB_r pourra choisir son autre candidat dans un deuxième temps, ou bien plus simplement que KB_e considère que KB_r "appelle" l'individu c_o, g_o.

VIII.3.2.2.c Comment lire une substitution sur une représentation simpliciale ?

Nous examinons dans ce paragraphe comment l'on peut utiliser les opérations simpliciales obtenues sur les tables VIII.3 et VIII.4 afin de lire directement la substitution valide, sans effectuer un matching syntaxique sur les expressions de LD. En d'autres termes, nous voulons savoir s'il est possible d'utiliser les représentations simpliciales dès le cas simple de requête non hétérogène gérable par matching direct.

Reprenons la requête $Q^a_{e \to r}$ de la page 208 qui correspond à la question :

"quel concept est inclus dans C ?"

Si nous traduisons cette question en formalisme simplicial à l'aide de la table VIII.3, cela revient à tester à l'aide de $A_{\subseteq}$:

"quel simplexe contient le simplexe de C ?"

Plus précisément, dans le cas de notre exemple, et compte tenu de la façon dont nous avons construit automatiquement ces simplexes par $A_{X:\leq:traits}$, cela revient à :

"chercher tous les simplexes qui ont un de leurs sommets égal à f_C."

En effet, f_C est le *seul* trait associé au concept C, dans cet exemple. et tous les concepts inclus dans C hériteront de ce trait et l'auront donc comme sous-face dans la représentation simpliciale associée.

Traiter la requête $Q^a_{e \to r} = ?X(X \leq C)$ en utilisant la représentation simpliciale revient alors à sélectionner tous les simplexes s'appuyant sur f_C.

Notons que cette recherche est plus facile si on l'effectue sur la représentation simpliciale *duale* (voir chapitre V). En effet, en posant la question en termes duaux, cela revient à :

"Chercher tous les sommets de la représentation simpliciale du trait f_C".

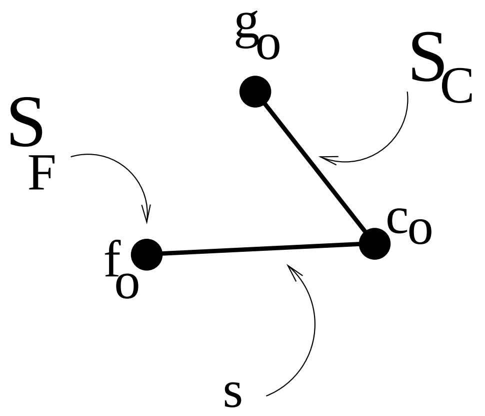
Rappelons que pour obtenir la représentation duale, nous devons considérer la relation binaire entre les concepts représentés par des sommets et les traits représentés comme des simplexes s'appuyant sur ces sommets, lorsque l'on construit la représentation simpliciale avec A_{cs}.

C'est-à-dire que l'on applique A_{cs} (voir page 194) à $\{concepts\} \times \{$traits associés par $A_{X:\leq:traits}\}$. Ensuite, on sélectionne sur cette représentation simpliciale le simplexe S_{f_C} , on obtient pour notre exemple :

$$S_{f_C} = \langle A, G \rangle$$

Donc les substitutions valides pour le pattern $?X$ de $Q^a_{e \to r}$ sont bien $(X \to A)$ et $(X \to G)$, comme le résultat obtenu par matching syntaxique sur les expressions de LD.

FIG. VIII.10 – Représentation simpliciale d'extension de la Abox Ab_r (voir table VIII.5) de l'agent récépteur KB_r. Le concept C a pour représentation le 1-simplexe S_C, le concept F, a pour représentation le 0-simplexe S_F, et le rôle s a pour représentation l'unique arc (s_0, c_0).



Reprenons maintenant la deuxième requête $Q^b_{r\to e}$ du paragraphe VIII.3.2.2.b. En formalisme simplicial, elle se traduit par (voir table VIII.3) :

"Quel sommet appartient au complexe d'extension du concept C ?"

Traiter la requête $Q^b_{e\to r}$ localement par KB_r, en utilisant directement la représentation simpliciale, revient alors à *sélectionner* les sommets appartenant au simplexe S_C dans le complexe d'extension contruit à l'aide de $A_\in$. La représentation simpliciale d'extension de Ab_r est donnée par la figure VIII.10.

VIII.3.2.3 Cas de matching avec déduction

Nous venons de voir que la gestion d'une requête non hétérogène passe par le *matching* du pattern de la requête avec les expressions LD de la Kbase Kb_r. Mais nous pouvons également avoir à traiter des requêtes dont le pattern ne matche pas *directement* avec une expression de Kb_r, mais dont on peut fournir une réponse (à l'aide d'une substitution valide) en passant par un *raisonnement* simple sur les connaissances de contenues dans Kb_r.

VIII.3.2.4 Requête sur un concept

Reprenons le même exemple que ci-dessus pour Tb_e et Tb_r, et posons la nouvelle requête d'indice c :

$$Q^c_{e\to r} = ?X(E \leq X)$$

qui se traduit par "à quel concept appartient E ?". Une réponse possible serait $R^c_{r\to e} = (X \to C)$, par le raisonnement en 4 étapes suivant :

1. On a par définition de Tb_r : $E \leq G \wedge F$.
2. On a par définition de Tbr : $G \leq C$.
3. De la description en 1, et par *définition* du constructeur $\wedge$, nous pouvons en déduire la description 3 : $E \leq G$.

4. Par *transitivité* de la relation $\leq$, et de la description déduite en 3, nous déduisons la description 4 : $E \leq C$.

Ainsi, Tb_2 peut maintenant matcher $?X(E \leq X)$ avec la description déduite en 4 : $E \leq C$, et trouver la substitution valide $R^c_{r \to e}= (X \to C)$.

Cette substitution a été trouvée après 4 étapes de raisonnement avant le matching, et a fait intervenir de la connaissance au sujet de :

– la définition du constructeur $\wedge$,
– la transitivité du lien $\leq$.

Ces étapes de raisonnement sont couramment appliquées dans la recherche de substitutions lors d'interrogations de bases de données en LD avec des patterns [13]. Cependant, il apparaît plus direct de lire cette inclusion directement sur la représentation simpliciale duale, tel qu'expliqué dans le paragraphe VIII.3.2.2.c.

VIII.4 Traitement des requêtes hétérogènes

Nous venons de voir comment procéder au matching direct et déduit d'une requête sur le Kbase Kb_r afin de fournir une substitution valide pour les patterns contenus dans les requêtes non hétérogènes. Nous avons vu également comment trouver ces même substitutions en lisant les représentations simpliciales extraites de Kb_r. Illustrons à présent la gestion d'une requête hétérogène, c'est-à-dire, contenant en plus du pattern, au moins un terme hétérogène pour Kb_r. La gestion d'une telle requête nécessite d'utiliser une *adaptation* pour fournir une substitution valide.

Nous illustrons l'utilisation de l'algorithme topologique d'adaptation proposé au chapitre VI (système ESQIMO) à l'aide d'un exemple de requête hétérogène portant sur un concept. Nous verrons en fin de section un exemple portant sur un individu et un exemple portant sur un rôle.

VIII.4.1 Préliminaires

Reprenons l'exemple de la page 190, et intéressons-nous uniquement aux relations d'inclusion notées $Incl_e$ et $Incl_r$ des Tbox des agents respectivement émetteur et récepteur. Ajoutons maintenant à la Kbase envoyeur, la description suivante :

$$A \leq H$$

L'interaction a maintenant lieu entre les Tboxes de la table VIII.6. Considérons dans ce cadre, la nouvelle requête, d'indice d, envoyée par la nouvelle base de KB_e à KB_r :

$$Q^d_{e \to r} = ?X(X \leq H)$$

TAB. VIII.6 – Nouvelle définition de Tb_e où l'on introduit le terme H, qui sera totalement hétérogène pour Tb_r.

$Tb_e=$		$Tb_r=$	
	$\{A \doteq E \wedge C$		$\{F \doteq \forall s.C$
	$D \doteq \forall r.C$		$G \leq C$
	$B \leq E$		$E \leq G \wedge F$
	$A \leq C$		$A \doteq B \wedge C$
	$A \leq H\}$		$A \leq C\}$

VIII.4.2 Un problème de compréhension

L'agent recevant la requête va chercher une substitution valide pour le pattern $?X$ de $Q^d_{e\to r}$ en tenant compte du fait que : c'est un concept, qu'il est relation d'inclusion avec un autre concept, et que ce deuxième concept est H. Pour donner une substitution valide de $?X$, il va tenter un matching syntaxique :

- Soit sur les données explicites de Tb_r dont il dispose, comme nous l'avons illustré dans le paragraphe VIII.3.2.2.a : le matching syntaxique échoue puisqu'il n'y a aucune description dans $Incl_r$ qui contienne le terme H ;
- Soit sur les données *déduites* à partir de Tb_r comme nous venons de le voir dans le paragraphe VIII.3.2.4 : ce cas échoue également car il n'y a aucun information à propos de H qui peut être déduite dans $Incl_r$ des autres équations, pour la simple raison que H n'est introduit nulle part dans cette Tbox.

Nous nous retrouvons dans un cas de Not-understanding (cf. chapitre III), où Kb_r est incapable de trouver une interprétation pour $Q^c_{e\to r}$ et donc de fournir une substitution valide de $?X$. Afin de gérer cette requête tout de même, Kb_r va tenter d'adapter le *terme hétérogène* de $Q^c_{e\to r}$ (c'est-à-dire ici H), en un *terme connu* pour lui (c'est-à-dire ayant une description explicite ou déduite dans $Incl_r$) par *adaptation*.

VIII.4.3 L'algorithme d'adaptation d'ESQIMO

Afin de procéder à cette adaptation, nous utilisons l'algorithme d'ESQIMO proposé au chapitre VI. Les grandes lignes de cet algorithme sont les suivantes :

1. **Représentation des données** du problème : le cas source, le cas cible et l'espace des figures possibles sont représentés à l'aide de complexes simpliciaux.
2. **Appariement structurel** : les représentations simpliciales des cas source et cible sont appariées en conservant leur structures.
3. **Transformation** : de l'appariement découle une transformation que l'on applique à la solution du cas source afin d'obtenir la solution du cas cilbe.

L'algorithme d'adaptation que nous allons proposer pour gérer les termes hétérogènes, noté $A_{\div}$, s'inspire de l'algorithme d'ESQIMO, moyennant quelques hypothèses que nous allons étudier. La représentation des données se fera avec les algorithmes déjà étudiés dans la section VIII.2. L'appariement structurel et l'application de la transformation sont réalisés par l'algorithme $A_{\div}$ sur la représentation simpliciale des données.

VIII.4.3.1 Représenter toutes les données

Pour utiliser l'algorithme d'ESQIMO dans le contexte de la gestion d'une requête hétérogène par une Kbase en LD, il nous faut donc commencer par contruire une représentation simpliciale de ce nouveau problème. Les données de ce nouveau problème sont :

- **Cas source** : rappelons qu'en analogie, le cas source est le cas connu dont on s'inspire pour trouver une solution au cas cible. Dans le problème de l'hétérogénéité sémantique, un cas est un terme, et un cas source est un terme *connu* de Kb_r.
- **Univers** des figures possibles : dans le problème des tests de QI, les figures que connaît ESQIMO, c'est-à-dire ce qu'on appelle l'univers des figures possibles, peuvent être candidates pour constituer le cas source. Dans le problème des termes hétérogènes, tous les termes contenus dans Kb_r peuvent être candidats pour être des cas source.
- **Cas cible** : rappelons qu'en analogie, le cas cible est le cas que l'on cherche à résoudre. C'est ce cas qui sera comparé aux candidats sources. Le cas cible dans le problème des termes hétérogènes est le terme hétérogène lui-même.

Représenter l'univers des possibles, revient donc à représenter simplicialement tous les termes définis dans Kb_r. Nous savons construire cette représentation simpliciale, à l'aide des algorithmes présentés dans la section VIII.2. Le cas source sera donc un simplexe appartenant à la représentation simpliciale de Kb_r. Notons que la représentation simpliciale de Kb_r se divise en deux types de complexes : les complexes d'inclusion et les complexes d'extension. Il faudra donc tenir compte de cette division lorsque l'on appliquera l'algorithme ESQIMO.

Représenter le cas cible revient alors à représenter le terme hétérogène. Mais comment représenter le terme hétérogène ? Si il est hétérogène, il est atomique et nous n'avons aucun connaissance à son sujet. Nous devons alors conserver toute information à son sujet qui vient du contexte dans lequel il est employé dans la requête d'où il provient. C'est pourquoi il s'avère préférable de représenter toute la requête à l'aide de complexes simpliciaux afin de disposer du maximum d'informations avant l'appariement structurel. Nous n'avons pas illustré jusqu'à présent la construction d'une représentation simpliciale d'une requête.

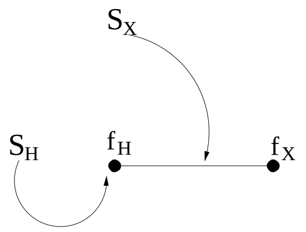
VIII.4.3.2 Appariement structurel

Apparier le cas cible avec les candidats sources revient alors à tenter d'apparier la représentation simpliciale de la requête avec la représentation simpliciale de Kb_r. Comme nous l'avons dit, il faudra décider sur laquelle des représentations simpliciales de Kb_r (inclusion ou extension) a lieu cet appariement. Nous verrons dans la suite que cela dépend du type du terme hétérogène qu'il faut adapter. Cette étape constitue le coeur de l'agorithme d'ESQIMO, elle est réalisée pour le problème des termes hétérogènes par l'algorithme appelé A_{app}. nous verrons que cet algorithme se décompose en plusieurs sous-fonctions.

VIII.4.3.3 Transformer le cas source

Transformer le cas source afin de trouver un sens pour le cas cible, revient alors très simplement à identifier sur la représentation de Kb_r utilisée le nom du concept ou du role ou de l'individu qui se trouve "à la place" du terme hétérogène une fois l'appariement accompli. Pour cela, il

FIG. VIII.11 – Représentation simpliciale de la requête $Q^c_{e\to r}$ posée par Kb_e à Kb_r dont les Tbox sont définies par la table VIII.6. Cette requête contient un terme hétérogène, H, et l'on construit sa représentation simpliciale afin de *l'apparier* structurellement avec des représentations simpliciales des termes connus dans $Incl_r$ qui doit gérer le terme hétérogène *localement*.



est donc important de disposer du *type* du terme hétérogène, de la même manière que nous nous servons du type des patterns pour les matcher.

Dans ESQIMO, appliquer la transformation découlant de l'appariement consistait en une déformation le long d'un chemin polygonal. Dans le cas des termes hétérogènes, la transformation est triviale puisqu'elle revient à lire le nom du sommet qui se situe à la place du terme hétérogène. C'est pour cela que nous avons souligné plus haut combien l'étape d'appariement est l'étape cruciale dans l'adaptation des termes hétérogènes.

VIII.4.4 Représentation simpliciale de la requête

Avant de procéder à l'appariement de structures entre le cas cible et les cas sources, nous devons construire la représentation simpliciale du cas cible. Nous avons dit qu'il est préférable de représenter le cas cible (terme hétérogène) avec son contexte (toute la requête) afin de disposer d'un maximum d'informations et donc de contraintes lors de l'appariement.

Une requête reçue par Kb_r est une expression de LD simple et nous savons donc en fournir une représentation simpliciale, d'après les correspondances établies au chapitre VII. Dans le cas de $Q^d_{e\to r} = ?X(X \leq H)$, l'expression de LD consiste uniquement en une inclusion de concepts. D'après la section VIII.2.2.1.b, page 191 de ce chapitre, cela correspond à un simplexe de dimension 1 dont un des traits est l'unique trait caractérisant le concept le plus grand, et le deuxième trait est celui qui est associé au plus petit concept, introduit pour le spécifier par rapport au plus grand, dans lequel il est inclus. Ici le plus grand concept est H, on lui associe le trait f_H, par l'algorithme $A_{X:\leq:traits}$. Le concept X à trouver est alors différencié de H par le trait f_X et nous obtenons les listes suivantes :

$$S_H = \langle f_H \rangle, S_X = \langle f_H, f_X \rangle$$

Ces listes de traits nous permettent alors de construire la représentation simpliciale de $Q^c_{e\to r}$ (voir figure VIII.11), en passant appliquant A_{cs} à la matrice d'incidence de la relation $\leq$ entre les ensembles $\{H, X\}$ et $\{f_H, f_X\}$ qui est triviale (non détaillé).

VIII.4.5 Algorithme d'adaptation $A_{\div}$ de la requête avec le complexe d'inclusion extrait de Kb_r

Nous disposons maintenant de la représentation simpliciale de la requête $Q^c_{e \to r}$, que nous notons S^c_X, et du complexe d'inclusion de Tb_r, que nous notons $C_{Tb_r:\leq}$. Nous pouvons donc procéder à leur *appariement* structurel.

VIII.4.5.1 Appariements dans ESQIMO ESQ_{app}

Rappelons que la procédure générale d'appariement structurel, que nous notons ici ESQ_{app}, entre deux simplexes S_A et S_B dans ESQIMO est la suivante (voir chapitre VI) :

- Soit som_A, l'ensemble des sommets de S_A, et som_B l'ensemble des sommets de S_B.
- Si $Card(som_A) = Card(som_B)$, les appariements possibles sont tous ceux qui permettent un recouvrement total de S_B. Par exemple, nous pouvons tout simplement apparier un sommet de S_A avec un et un seul sommet de S_B . Toutes les permutations de cet appariement seront aussi acceptables. Mais nous pouvons aussi projeter toute partie de S_A sur un sommet de S_B, du moment que tous les sommets de S_B sont appariés. Des exemples d'appariements autorisés sont donnés dans le chapitre VI.
- Si $Card(som_A) > Card(som_B)$, il faut que certains sommets de S_A *disparaissent* ou bien que plusieurs sommets de S_A se *transforment* en un seul sommet de S_B.
- Enfin, si $Card(som_A) < Card(som_B)$, il faut qu'un sommet de S_A se *transforme* en plusieurs sommets de S_B, ou bien qu'il y ait *création* de sommets, ou que plusieurs sommets de S_A se *transforment* en encore plus de sommets dans S_B. Toutes ces heuristiques d'appariement ont été envisagées et programmées.

Pour chaque appariement établi, ESQ_{app} détermine une déformation élémentaire associée pour passer de chacun des éléments de la paire appariée à l'autre. Ce sont ces déformations élémentaires qui constituent ensuite les transformations.

VIII.4.5.2 Adaptation des termes hétérogènes $A_{\div}$

L'algorithme $A_{\div}$ d'appariement structurel pour les termes hétérogènes doit donc utiliser l'algorithme d'appariement d'ESQIMO, ESQ_{app}. Pour l'utiliser, il faut :

1. Tester tous les simplexes $S_{candidat}$ de $C_{Tb_r:\leq}$ pour vérifier s'ils s'apparient avec S^c_X : cette fonction de test est réalisée par l'algorithme A_{test}.
2. Appliquer alors à chaque essai l'algorithme d'appariement ESQ_{app} ci-dessus : cette fonction est appelée à chaque fois par A_{test}.
3. Identifier dans chaque candidat $S_{candidat}$ les sommets qui ont la même place que la représentation du terme hétérogène de $Q^c_{e \to r}$: cette fonction est réalisée par l'algorithme A_{id} qui renvoie les candidats à l'adaptation du terme inconnu.

Ainsi, la fonction $A_{\div}$ se décompose en :

$$A_{\div} = A_{id \circ ESQ_{app} \circ A_{test}}$$

VIII.4.5.3 Fonction de test A_{test}

La fonction de test A_{test} doit parcourir le complexe approprié extrait de Tb_r afin de préparer les appariements possibles avec S_X^c. Ce parcours s'effectue selon les étapes :

- Soit C_{Tb_r} la représentation simpliciale de Tb_r à exploiter : par exemple C_{Tb_r} =$C_{Tb_r:\leq}$ si la requête porte sur des inclusions de concepts ;
- Soit $C^+_{Tb_r}$ le développement de C_{Tb_r} : l'ensemble des simplexes de C_{Tb_r} ;
- Soit $C^+_{Tb_r/d_X}$ la sélection dans $C^+_{Tb_r}$ tous les simplexes de même dimension que S_X^c ;
- Soit $C^+_{Tb_r/candidats}$ la sélection dans $C^+_{Tb_r/d_X}$ des simplexes correspondant à un concept nommé dans Tb_r : ce sont les simplexes qui ont une existence pour Kb_r ;
- Pour chaque élément candidat dans $C^+_{Tb_r/candidats}$, appeler la fonction d'appariement ESQ_{app}.

Ces étapes se traduisent par l'algorithme de sélection de candidats, qui s'applique directement sur la représentation simpliciale des données S_X pour la requête, C pour la Tbox Tb_r, et au terme hétérogène noté H :

A_{test} :
`Soit` t `le type du terme hétérogène` H
`Sélectionner la représentation` C `extraite de` Tb_r `en fonction de` t
`Soit` C^+ `le développement de` C
`Soit la` d `dimension de` S_X
`Soit` $list$ `les simplexes de dimension` d `dans` C^+
`Pour chaque simplexe` $\sigma \in list$
  `Si` σ `porte un nom`
    `appeler` $ESQ_{app}(\sigma, S_X)$
    `stocker l'appariement dans` $list_{app}$
  `Sinon, l'effacer de` $list$

Remarque : Nous éliminons de $list$ les simplexes qui ne portent pas de nom pour Tb_r. En effet, même si les simplexes ne correspondant pas à des concepts de Tb_r peuvent s'apparier avec S_X dans le sens où ce sont des simplexes de même dimension que lui, ils ne correspondent pas à la *définition* d'un quelconque concept et ne pourront donc pas nous fournir un candidat pour l'interprétation du terme hétérogène en un terme connu, puisqu'ils ne sont pas connus eux-mêmes.

VIII.4.5.4 Fonction d'identification des candidats à l'adaptation A_{id}

La fonction précédente termine avec une liste $list$ ne contenant que les simplexes candidats et la liste $list_{app}$ de leurs appariements avec S_X par ESQ_{app}. Pour chacun de ces appariements, il faudra identifier le nom du simplexe "à la place" de H. Un appariement de la liste $list_{app}$ est un ensemble de paires de correspondances (voir algorithme ESQIMO, chapitre VI) avec à gauche un nom de sommet de S_X et à droite un nom de sommet de C. Voici l'algorithme simple qui découle pour A_{id} :

A_{id} :

```
Pour chaque élément a de list_app
   Pour chaque paire p de a
   Si (l'élément de gauche est H
       Et
       l'élément de droite est un nom de type t existant)
      ajouter l'élément de droite dans list
   Sinon, passer au suivant
```

VIII.4.5.5 Application à l'exemple

Dans notre exemple, les données du problème sont :

- S_X est le simplexe que nous venons de construire pour la requête $Q^c_{e\to r}$, à savoir S^c_X ;
- Le complexe extrait de Tb_r à utiliser est donc celui d'inclusion, il a déjà été étudié dans la section VIII.2.2.1.b. Rappelons que nous le notons $C_{Incl_r:\leq}$;
- Le terme hétérogène de la requête est H, c'est un concept (type t).

Le résultat de l'appariement du simplexe S_X avec $C_{Tb_2:\leq}$ par $A_\div$ s'obtient par les étapes suivantes :

$$A_\div = A_{id\circ ESQ_{app}\circ A_{test}}$$

Appliquons donc A_{test} en premier A_{test} (H, Tb_r, S^c_X) :

$t = concept$;

$C = C_{Incl_r:\leq}$;

$d = 1$;

$list = [S_A; S_G; \langle f_C, f_G\rangle; \langle f_F, f_G\rangle; \langle f_G, f_E\rangle; \langle f_F, f_E\rangle]$;

$\langle f_C, f_G\rangle, \langle f_F, f_G\rangle, \langle f_G, f_E\rangle, \langle f_F, f_E\rangle$ ne portent pas de nom pour Tb_r ;

$list_{app} = [S_A; S_G]$;

Où nous notons les listes entre [;]. Ainsi, A_{test} (H, Tb_r, S^c_X) donne en sortie seulement deux candidats : S_A et S_G.

L'algorithme d'appariement ESQ_{app} apparie sans difficulté ces simplexes de dimension 1. Cela donne les paires :

$$ESQ_{app}(S_X, S_A) = [(f_H, f_C); (f_X, f_A)]ou[(f_H, f_A); (f_X, f_C)]$$

$$ESQ_{app}(S_X, S_G) = [(f_H, f_C); (f_X, f_G)]ou[(f_H, f_G); (f_X, f_C)]$$

Et l'algorithme d'identification A_{id} appliqué à ces listes deux possibilités d'adaptation pour H à chaque fois :

$$A_{id}(ESQ_{app}(S_X, S_A), H) = [C; A]$$

$$A_{id}(ESQ_{app}(S_X, S_A), H) = [C; G]$$

Les deux appariements donnent un même candidat C pour adapter H : $(H \rightharpoondown C)$. Il est donc tentant de reternir ce candidat en priorité. Dans ce cas, $Q^c_{e\to r}$ est ramenée à la requête simple $Q^a_{e\to r} = ?X(X \leq C)$ de la section VIII.3.2.2.a traitant des matchings directs sur des requêtes non hétérogènes.

Remarque : Quel que soit le candidat finalement retenu, le terme hétérogène est adapté en un terme connu, et la requête peut maintenant être traitée comme une requête non hétérogène. Bien sûr, le choix du candidat va influer sur les résultats du matching pour le pattern de la requête. Mais Kb_r n'a pas d'autre moyen de savoir quelle adaptation de H est plus ou moins proche des connaissances de Tb_e auxquelles il n'a pas accès. C'est seulement au niveau d'un *cycle* d'interaction que les différentes solutions pourront être testées, et que les candidats pourront être éliminés.

VIII.4.6 Production de substitutions Valides

A partir des candidats pour le terme hétérogène, nous pouvons proposer des adaptations qui nous mèneront à fournir des substitutions valides pour le pattern de la requête.

Dans notre exemple, le seul candidat retenu pour adapter le terme H est C. Nous démarrons la recherche d'une susbstitution du pattern, avec l'adaptation suivante :

$$(H \rightharpoondown C)$$

Ainsi, la requête $Q^c_{e\to r}$ devient :

$$Q^c_{e\to r} = ?X(X \leq C)$$

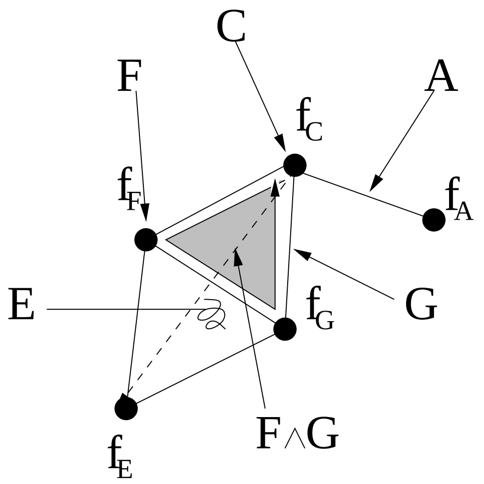
qui est ramenée à une requête soit avec matching direct, soit avec déduction, comme nous l'avons déjà étudié. Dans le cas précis de cet exemple, la requête est même ramenée à une requête que nous avons déjà traitée et qui est $Q^a_{e\to r}$ (voir section VIII.3.2.2.a, page 208).

Finalement, la réponse complète que la Kbase réceptrice Kb_r fournira à Kb_e est l'une des deux suivantes :

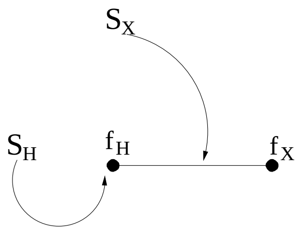
$$R^c_{r\to e} = (H \rightharpoondown C, X \to A)$$

$$R^c_{r\to e} = (H \rightharpoondown C, X \to G)$$

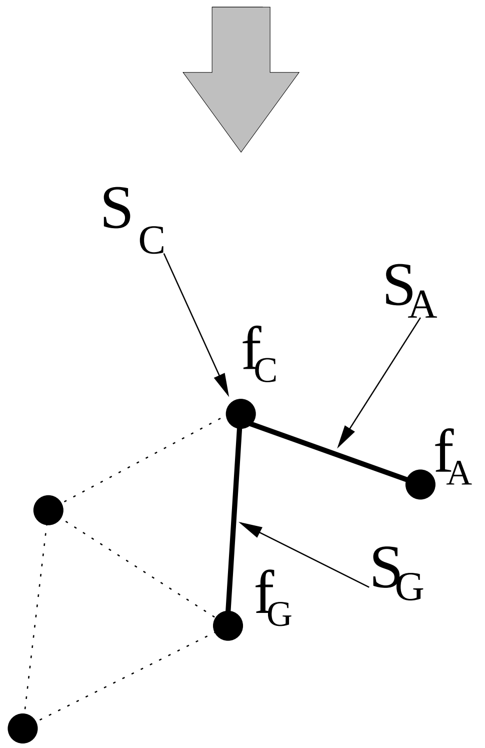
Fig. VIII.12 – Illustration de l'appariement structurel de la représentation simpliciale du pattern d'une requête avec le complexe extrait de la Kbase réceptrice. Ici la requête utilise un pattern de concept, le terme hétérogène à adapter est aussi un concept, le complexe extrait que l'on utilise est un concept d'inclusion. a) Le complexe d'inclusion est celui qui est donné par la figure VIII.7, avec les notations allégées. b) Le simplexe de pattern est construit dans la section précédente. c) Les deux simplexes candidats appariables avec S_X sont alors S_A et S_G, comme le montre cette sous-figure. Elle représente les simplexes de la sous-figure a) qui "restent" après appariement avec le simplexe de la sous-figure b).



(a) Complexe d'inclusion de la Tbox réceptrice



(b) Simplexe représentant le pattern de la requete.



(c) Simplexes candidats, résultant de l'appariement structurel du simplexe du pattern de la requete dans le complexe d'inclusion de la Tbox réceptrice.

VIII.4.7 Autres exemples

VIII.4.7.1 Requête hétérogène sur un individu

Illustrons à présent une gestion de requête qui nécessite d'utiliser une *adaptation* pour fournir une substitution valide.

Considérons dans ce cadre, la requête suivante :

$$Q^e_{e \to r} = ?x(x \in H)$$

envoyée par la Kbase de KB_e à KB_r.

L'agent recevant la requête va chercher une substitution valide pour le pattern $?x$ en tenant compte du fait que c'est un individu et qu'il est relation d'inclusion avec le concept H. Mais il ne connait toujours pas le concept H !

KB_r aura alors recours à une adaptation du concept H, exactement selon la procédure illustrée en section VIII.4. Cette adaptation donne pour H, les deux candidats A ou C. Les réponses que peut formuler KB_r sont alors :

$$R^e_{r \to e} = (H \rightharpoondown A, x \to \emptyset)$$

$$R^e_{r \to e} = (H \rightharpoondown C, x \to C)$$

Dans le premier cas, KB_r ne connaît aucune instance de A dans Ab_r, dans le deuxième cas, $Q^e_{e \to r}$ est ramenée à $Q^d_{e \to r}$, et la réponse est la même, à l'adaptation près.

VIII.4.7.2 Requête hétérogène sur un rôle

Après avoir étudié comment répondre à une requête portant sur des concepts, puis sur des individus, intéressons-nous au cas des requêtes nécessitant une adaptation sur les rôles.

Remarque : Il n'est pas très intéressant d'étudier les requêtes portant sur un rôle puisque nous pouvons exprimer peu d'informations à propos des rôles eux-mêmes dans AL. Les rôles sont plutôt utilisés dans des définitions de concepts, et interviennent donc aussi indirectement dans les requêtes.

Soit la requête posée par KB_e à KB_r, et déjà abordée au chapitre III :

$$Q^f_{e \to r} = ?X(X \doteq \forall r.C)$$

Nous avions vu que la fonction de gestion de la requête essaie d'abord de matcher le pattern de la requête avec le meilleur candidat.

Mais pour déterminer ce candidat, KB_r devrait connaître tous les termes de $Q^f_{e \to r}$, ce qui n'est pas le cas. En effet, le rôle r lui est inconnu. Nous sommes donc en présence d'un terme hétérogène, mais cette fois il est du type “rôle”.

Nous savons maintenant que pour adapter ce terme hétérogène en un terme connu, nous devons d'abord procéder à un appariement *structurel* entre la représentation simpliciale de la requête et celle de la Kbase KB_r.

La requête est une expression de LD contenant un constructeur $\forall$ qui restreint le type. Comme nous l'avons vu dans la section VIII.2.3.2.a, la représentation simpliciale d'un tel pattern est le complexe formé de tous les arcs étiquetés r sur le complexe d'extension (voir figure VIII.9).

Cette représentation simpliciale de la requête sera appariée sur la représentation en extension de Kb_r qui est donnée par la figure VIII.10.

Ainsi, les candidats pour adapter le terme hétérogène r sont réduits au seul rôle s. Il en découle un candidat pour le pattern $?X$, réduit au concept :

$$F \doteq \forall s.C$$

comme nous l'avions déjà vu au chapitre III.

ce matching a réussi, avec la substitution $X \rightarrow F$, grâce à l'adaptation préalable : $r \rightharpoondown s$.

Finalement, la substitution valide donnée par KB_r comme réponse sera :

$$R^d_{r \rightarrow e} = (X \rightarrow F, r \rightharpoondown s)$$

comme prévu au chapitre III.

VIII.5 Conclusions sur l'intégration des outils

Nous avons repris la procédure générale de gestion d'une requête hétérogène afin de positionner les algorithmes que nous proposons dans ce chapitre par rapport à tout un cycle de traitement. Nous avons vu que le traitement d'une requête dépend du type de pattern et peut ou pas présenter des termes hétérogènes.

Dans tous les cas, le traitement commence par une phase préalable d'extraction des représentations simpliciales à partir de l'écriture en LD de la Kbase qui doit traiter la requête. Cette phase d'extraction est détaillée et illustrée dans le cas de chaque type de connaissance et chaque type de relation, et ce uniquement pour le langage de LD AL. Nous aboutissons à deux familles de représentations simpliciales : un complexe d'inclusion et un complexe d'extension.

Nous nous sommes ensuite intéréssés aux requêtes qu'il est possible de poser à une Kbase et qu'elle aura à gérer. Nous avons fourni pour chacune d'entre elles une sémantique intuitive mais surtout une correspondance en termes d'opérations simpliciales à effectuer sur les représentations simpliciales extraites de la Kbase, pour y répondre.

Nous avons pu ensuite nous intéresser au traitement même de ces requêtes. Nous avons commencé par nous intéresser aux requêtes ne contenant pas de termes hétérogènes et qui nécessitaient soient un matching direct soit un raisonnement simple pour y répondre. Nous avons pu ainsi traiter le cas des requêtes hétérogènes en dernier.

Pour cela, il nous a fallu appliquer l'algorithme d'adaptation développé au chapitre VI aux données du nouveau problème. Nous avons donc détaillé le nouvel algorithme résultant avant de l'illustrer sur des concepts, des individus et des requêtes. Nous avons terminé sur l'exemple du chapitre III, enfin résolu.

Conclusion de la thèse

Nous avons présenté le projet InterViews qui est proposé pour aborder les nouveaux domaines de recherche découlant de l'évolution du Web sur Internet. Dans ce cadre, nous nous sommes intéressés au problème de l'hétérogénéité sémantique entre agents dialogiques. Nous avons défini un agent dialogique comme un composant logiciel qui est capable d'interagir à un haut niveau sémantique avec d'autres agents (logiciels ou humains).

Notre travail a commencé par l'identification d'un nouveau problème entre agents interagissant par dialogisme : l'hétérogénéité sémantique. Ce problème, reconnu comme fondamental aujourd'hui par la communauté des systèmes multi-agents, doit être distingué du problème de l'intégration de bases de données hétérogènes. Notons que le problème de l'Hétérogénéité Sémantique est à différencier de l'hétérogénéité logicielle et matérielle (problèmes qui ont déjà été abordés et pour lesquels des solutions efficaces sont déjà mises en oeuvre), et de l'intégration de bases de données hétérogènes (domaine où l'on vise à homogénéiser deux sources présentant une hétérogénéité sémantique). Il a donc fallu, dans cette thèse, délimiter, définir et formaliser ce problème nouveau. Pour cela nous avons utilisé la logique des descriptions.

Nous avons posé les hypothèses de la *gestion* de l'hétérogénéité sémantique entre agents dialogiques modélisés en logique de descriptions pour InterViews, par opposition à sa *résolution* qui mènerait à l'homogénéisation des Kbases. Compte tenu de ces contraintes, nous avons enfin défini *formellement* l'hétérogénéité sémantique entre agents.

Nous avons ensuite proposé, pour ce problème, une solution *nouvelle* et *originale* qui est d'utiliser un raisonnement à partir de cas afin de trouver une interprétation aux termes hétérogènes dans une requête. Cela nous a conduit à étudier plus spécifiquement l'étape d'adaptation dans le raisonnement par analogie, pour laquelle nous avons fourni un modèle fondé sur des outils de la topologie algébrique.

En effet, la métaphore spatiale nous a menés à l'utilisation de notions provenant de la topologie algébrique que l'on a présentées en tant que *formalisme* de *représentation* des *connaissances*. Nous avons également fourni un processus de construction d'une telle représentation, par catégorisation, pour un agent disposant de détecteurs assimilés à des prédicats unaires.

Ce modèle a mené à l'implémentation d'un système de résolution automatique d'analogies simples. Avec ce formalisme de représentation des connaissances basé sur la topologie algébrique, nous avons proposé un algorithme *d'adaptation* de *structures* pour l'analogie : ESQIMO. Nous avons présenté en détail son algorithme général et l'avons illustré avec des exemples simples. Le système ESQIMO a fait l'objet de plusieurs publications [92], [89], [91], [90], [93]. Notons que ESQIMO a été prototypé en Mathematica puis programmé en Caml.

Nous avons ensuite montré pourquoi et comment il est possible d'intégrer tous ces outils dans un système complet et cohérent pour le traitement de l'hétérogénéité sémantique dans une interaction dialogique. Cette correspondance se traduit soit par une *extraction* de complexes à partir directement des relations *d'inclusion* de la Tbox, ou alors par une interprétation des formules de LD en termes de complexes simpliciaux (qui sont alors les *extensions*) [95].

Nous avons, pour finir, illustré dans le dernier chapitre, l'intégration des outils proposés aux chapitres III, V et VI, en nous appuyant sur les correspondances établies au chapitre VII, chapitre qui a également fait l'objet de publication. Pour cela, nous avons détaillé toutes les étapes du traitement d'une requête par un agent récepteur : depuis l'extraction d'une représentation simpliciale jusqu'à la production d'une substitution valide.

Ce travail, qui se veut précurseur, mérite d'être poursuivi afin de le mener vers des étapes plus implémentatoires ; en particulier, nous souhaiterions : implémenter les différents algorithmes présentés dans la langage développé pour la plate-forme InterViews, puis les tester lors de l'interaction dialogiques d'agents.

En effet, la subtilité de la gestion du problème de la HS dans InterViews est de gérer l'hétérogénéité lors d'une interaction pour rendre la communication possible, tout en préservant les différences et en évitant le plus possible une homogénéisation, qui est toujours difficile à mettre en oeuvre en environnement contrôlé, et impossible à atteindre dans les cas généraux des systèmes ouverts de l'Internet. Pour ces raisons, il est délicat et important de bien délimiter et bien définir le problème de la HS entre agents avec un formalisme rigoureux et ayant fait ses preuves en représentation des connaissances.

Nous aimerions par ailleurs approfondir l'approche formelle de comparaison entre la logique de descriptions et le formalisme CAT proposé dans cette thèse. Nous aimerions notamment affiner les mécanismes d'extraction de complexes pour chaque type de relations d'une Kbase en LD, et étendre ces mécanismes à d'autres langages de LD que $\mathcal{AL}$.

Enfin, notre proposition devra être plongée dans un cycle complet d'interaction, afin d'envisager le traitement de problèmes tels que le MUP (défini au chapitre III), où il est nécessaire de prendre en compte plusieurs pas d'interaction.

Références

[1] Peter F. Patel-Schneider Alex Borgida. A semantics and complete algortihm for subsomption in the classic description logic. *Journal of Artificial Intelligence Research*, 1 :277–308, 1994.

[2] Aristotle. *The Organon*, chapter Prior Analytics. Harvard University Press, Cambridge, MA, 1838.

[3] Alessandro Artale, Enrico Franconi, Nicola Guarino, and Luca Pazzi. *Data and Knowledge Engineering*, chapter Part-Whole Relation in Object-centered Systems : an Overview. Elsevier, 1996.

[4] R. H. Atkin. *Combinatorial Connectivities in Social Systems*. Verlag, 1977.

[5] F. Baader. *Artificial Intelligence Today, Recent Trends and Developments*, chapter Logic-Based Knowledge Representation, pages 13–41. Number 1600 in Lecture Notes in Computer Science. Springer Verlag, m.j. wooldridge and m. veloso edition, 1999.

[6] F. Baader and al. Terminological knowledge representation : A proposal for a terminological logic. Technical Report TM-90-04, DFKI, Deutsches Forschungszentrum für Künstliche Intelligenz, 1990.

[7] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 96–101. Morgan Kaufmann, 1999.

[8] Jon Barwise and John Etchemendy. Heterogenous logic. In *Diagrammatic Reasoning*, chapter 7, pages 211–234. AAAI Press, 1995.

[9] J. B. Berthelin. Contestation dans le dialogue artificiel. *Cahiers de linguistique sociale*, 1992.

[10] A. Borgida, R. Brachman, D. McGuinness, and L. Resnick L. Classic : a structural data model for objects. *SIGMOD-89*, pages 58–67, 1989.

[11] Alexander Borgida. Description logics in data manageent. Portion of the paper based on invited talk at IFIP'92, available on the web at Borgida's homepage, 1995.

[12] Alexander Borgida. On the relative expressiveness of description logics and predicate logics. Technical Report 9600004-5, Rutgers University, New Brunswick, USA, January 1996.

[13] Alexander Borgida and Deborah L. McGuiness. Asking queries about frames. In *KR'96*, 1996.

[14] R. J. Brachman. *A structural paradigm for representing knowledge*. PhD thesis, Division of Engineering, Harvard University, 1977.

[15] R. J. Brachman. What's in a concept : structural foundations for semantic networks. *Int. Journal of Man-Machine Studies*, 9 :127–152, 1977.

[16] Ronald J. Brachman. "i lied about the trees" : Or, defaults and definitions in knowledge representation. *The AI Magazine*, 6 :80–93, 1985.

[17] Levesque H. Brachman R., Pigman V. An essential hybrid reasoning system : Knowledge and symbol level accounts of krypton. In *IJCAI-85*, pages 532–539, 1985.

[18] Peter Burmeister. Formal concept analysis with comimp : Introduction to the basic features. Found on the ComImp Web Site.

[19] Alistair E. Campbell, Hans Chalupsky, and Stuart C. Shapiro. Ontological mediation : An analysis. DRAFT, OM web site, 1995.

[20] Alistair E. Campbell and Stuart C. Shapiro. Algorithms for ontological mediation. Technical Report 98-02, University of Buffalo, 1998.

[21] J. L. Casti. *Complexification*. Harper Perennial, 1 edition, 1994.

[22] Alon Y. Levy Catriel Beeri and Marie-Christine Rousset. Rewriting queries using views in description logics. In *Proceedings of the 16th ACM Symposium on Principles of Database Systems*. PODS-97, 1997.

[23] D.J. Chalmers, R.M. French, and D.R. Hofstadter. High-level perception, representation and analogy. *Journal of Experimental and Theoretical Artificial Intelligence*, 4(3) :185–211, July-Sept 1992.

[24] B. Chandrasekaran, Janice Glasgow, and N. Hari Narayan. Introduction. In Janice Glasgow, editor, *Diagrammatic Reasoning*. AAAI Press/MIT Press, 1995.

[25] Guillaume Chicoisne, Pierre-Michel Ricordel, Sylvie Pesty, and Yves Demazeau. Outils et pistes pour la pratique du dialogisme entre agents. In *Systèmes Multi-Agents, de l'interaction à la sociabilité*. JFIADSMA'98, Hermès, 1998.

[26] Catherine A. Clement and Dedre Gentner. Systematicity as a selection constraint in analogical mapping. *Cognitive Science*, 15(1), January-March 1991.

[27] Allen Collins and Ross Quillian. Retrieval time from semantic memory. *J. of verbal learning and verbal behavior*, 9 :240–247, 1969.

[28] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. *Principles of Knowledge Representation*, chapter Reasoning in Description Logics, pages 193–238. Studies in Logic. CSLI Publications, 1996.

[29] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. Technical Report RR-95-07, Deutsches Forschungszentrum für Künstlische Intelligenz, Kaiserlautern, 1995.

[30] Thomas G. Evans. A program for the solution of a class of geometric analogy intelligence-test questions. In *Semantic Information Processing*, chapter 5, pages 271–353. The MIT Press, 1968.

[31] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. The structure-mapping engine. *Artificial Intelligence*, 41(1) :1–63, November 1989.

[32] D. Fensel and S. Decker ad M. Erdmann andR. Studer. Ontobroker : The very high idea. In *Proceedings of the 11th International Flairs Conference*. FLAIRS-98, AAAI, 1998.

[33] Jacques Ferber. *Les Systèmes Multi-Agents*. InterEditions, 1995.

[34] Tim finin and Richard Fritzson. Kqml as an agent communication language. In *CIKM'94*. ACM Press, November 1994.

[35] Béatrice Fuchs, Jean Lieber, Alain Mille, and Amedeo Napoli. Vers une théorie unifiée de l'adaptation en raisonnement à partir de cas. In *RAPC99*, pages 77–85, Palaiseau, Juin 1999. AFIA.

[36] Ron fulbright and Larry M. Stephens. Classification of multiagent systems. Technical Report ECE 06-94-02, University of South Carolina, USA, June 1994.

[37] Dedre Gentner. Studies in metaphor and complex analogies. In *Symposium on Metaphor as Process*. American Psychology Society, 1980.

[38] Dedre Gentner. Structure-mapping a theoretical framework for analogy. *Cognitive Science*, 7(2) :155–170, April-June 1983.

[39] Giuseppe De Giacomo and Maurizio Lenzerini. What's in an aggregate : foundations for description logics with tuples and sets. In *IJCAI95*, pages 801–807. Morgan Kaufmann, 1995.

[40] Jean-Louis Giavitto and Erika Valencia. Combinatorial algebraic topology for dagrammatic reasoning. Technical Report 1165, LRI, ura 410, april 1998.

[41] M.L. Gick and K.J. Holyoak. Schema induction and analogical transfert. *Cognitive Psychology*, 1(15) :1–38, 1983.

[42] M.-D. Gineste and B. Indurkhya. Modèles mentaux, analogie et cognition. In *Les modèles mentaux, approche cognitive des représentations*, chapter 6, pages 143–170. masson, Paris, 1992.

[43] Nicola Guarino. Formal ontology, conceptual analysis and kowledge representation. *IJHCS*, 43(5/6), 1995.

[44] Christopher Habel and Carola Eschenbach. Abstract structures in spatial cognition. In Christian Freska, Matthias Jantzen, and RÜdiger Valk, editors, *Foundations of Computer Science*, volume 1337 of *LNCS*. Springer-Verlag, 1997.

[45] Rogers P. Hall. Computational approaches to analogical reasoning : A comparative analysis. *Artificial Intelligence*, 39(1) :39–120, may 1989.

[46] J. P. Haton and al. *Le raisonnement en intelligence artificielle*. InterEditions, 1991.

[47] Garry Hendrix. *Associative Networks : representation and use of knowledge by computers*, chapter Encoding knowledge in partitionned networks, pages 51–92. Findler Academic Press, 1979.

[48] M. Hesse. *Models and Analogies in Science*. Indiana : University of Notre Dame Press, 1966.

[49] Karl Hewitt. The challenge of open systems. *BYTE*, pages 233–242, April 1985.

[50] Graeme Hirst, Susan McRoy, Peter Heeman, Philip Edmonds, and Diane Horton. Repairing conversational misunderstandings and non-understandings. *Speech Communication*, 15, 1994.

[51] John H. Holland, Keith J. Holyoak, Richard E. Nisbett, and Paul R. Thagard. *Induction – Processes of Inference, learning, and Discovery*. The MIT Press, 1986.

[52] K. J. Holyoak and P. Thagard. Analogical mapping by constraint satisfaction. *Cognitive Science*, 13(3) :295–355, July-september 1989.

[53] K.J. Holyoak. *The Psychology of Learning and Motivation*, chapter The Pragmatics of Analogical Transfer. Academic Press, g.h. bower edition, 1985.

[54] Richard Hull. Managing semantic heterogeneity in databases : A theoretical perspective. In *PODS 97*. ACM Symposium on Principles of Database Systems, 1997.

[55] Bipin Indurkhya. Modes of analogy. In K.P.Jantke, editor, *Analogical and Inductive Inference*, volume 397 of *LNAI*, pages 217–230. Springer-Verlag, 1989.

[56] N. Johnson-Laird. La théorie des modèles mentaux. In *Les modèles mentaux, approche cognitive des représentations*, chapter Introduction, pages 1–21. masson, Paris, 1992.

[57] Daniel Kayser. *La représentation des connaissances*. Hermès, 1997.

[58] R. Küsters and A. Borgida. What's not in a name ? initial explorations of a structural approach to integrating large concept knowledge-bases. Technical Report DCS-TR-391, Department of Computer Science, Rutgers University, 1999.

[59] Mario Lenz, Brigitte Bartsch-Sporl, Hans-Dieter Burkhard, and Stefan Wess, editors. *Case-Based Reasoning, From Foundations to Applications*. Number 1400 in LNAI. Springer, 1998.

[60] Luger. *Artificial Intelligence*. Addison-Wesley, 1998.

[61] R. MacGregor and D. Brill. Recognition algorithms for the loom classifier. In *AAAI-92*, pages 774–779. MIT Press, 1992.

[62] Erica Melis and Manuela Veloso. *Handbook of Practical Logic*, chapter Analogy in Problem Solving. Springer-Verlag, 1996.

[63] J.S. Mill. *Collected Works of John Stuart Mill*, chapter System of Logic. University of Toronto Press, Toronto, Canada, 1873.

[64] Marvin Minsky. *Mind Design*, chapter A Framework for Representing Knowledge. MIT Press, 1981.

[65] Marvin Minsky. *Extraterrestrials : Science and Alien Intelligence*, chapter Communication with Alien Intelligence. Cambridge University Press, 1985.

[66] Melanie Mitchell. *analogy Making as Perception*. MIT Press, 1993.

[67] Tom M. Mitchell, Richard M. Keller, and Smadar T. Kedar-Cabelli. Explanation-based generalization : A unifying view. *Machine Learning*, 1(1) :253–263, 1986.

[68] J.P. Müllet and J. Quinqueton, editors. *IA distribuée et systèmes multi-agents*, chapter La théorie des actes de langage en intelligence artificielle distribuée : utilisations et limites, pages 229–249. Hermès, 1996.

[69] Ivan Moore. A simple and efficient algorithm for onferring inheritance hierarchies. In *TOOLS 96*. TOOLS Europe, Prentice-Hall, 1996.

[70] Clayton T. Morrison and Eric Dietrich. Structure-mapping engine vs. high-level perception : The mistaken fight over the explanation of analogy. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, pages 678–682. CCS, 1995.

[71] Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*. Number 422 in LNAI. Springer-Verlag, New York, 2nd edition, 1995.

[72] M. Ross Quilian. Word concepts : A theory and simulation of some basic capabilities. *Behavioral Science*, 12 :410–430, 1967.

[73] W. Quine. *From a Philosophical Point of View.* Harvard University Press, 3rd edition, 1980.

[74] A Ribeiro. *Un Modèle d'Interaction Dynamique pour les Systèmes Multi-Agents.* PhD thesis, Université Joseph Fourier, INP, Grenoble, February 2000.

[75] Gérard Sabah. *L'intelligence Artificielle et le langage, la représentation des connaissances.* Hermès, Paris, 1990.

[76] Andrea Schaerf. Reasoning with individuals in concept languages. Action ESPRIT nř6810 (COMPULOG2), 1993.

[77] Andrea Schaerf. *Query Answering in Concept-Based Knowledge Representation Systems : Algorithms, Complexity, and Semantic Issues.* PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1994.

[78] Andrea Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2) :141–176, 1994.

[79] Mark W. Schmolze J. The nikl experience. *Computational Intelligence*, 7(1) :48–69, 1991.

[80] J.R. Searle. *Speech Acts.* Cambridge University Press, 1969.

[81] J. F. Sowa. *Conceptual structures.* Addison-Wesley, 1984.

[82] Luc Steels. *The Talking Heads Experiment.* LABORATORIUM, Antwerpen, 1999.

[83] F. Flores T. Winograd. *Understanding Computers and Cognition : A New Foundation for Design.* Ablex, Norwood, 1986.

[84] A.I. Ujomow. *Studien zur Logik der wissenschaftlichen Erkenntnis*, chapter Die Hauptformen und regeln der Analogieschlüsse. Akademie Verlag, 1967.

[85] Erika Valencia. Un modèle topologique pour le raisonnement diagrammatique. Rapport pour le DEA Sciences Cognitives, LIMSI. See also `http ://www.lri.fr/~erika/`, August 1997.

[86] Erika Valencia. Hitch hiker's guide to esqimo. RR 1173, LRI, ura 410 CNRS, Université Paris-Sud, 91405 Orsay, France. See also `http ://www.lri.fr/~erika/`, May 1998.

[87] Erika Valencia. Etude de cas en logique de description pour un modèle d'interaction entre bases de connaissances. Technical Report à paraître, LIMSI-CNRS, avril 2000.

[88] Erika Valencia. Représentation basée sur l'accès par le contenu pour l'interaction dialogique. Technical Report à paraître, LIMSI-CNRS, janvier 2000.

[89] Erika Valencia and Jean-Louis Giavitto. Algebraic topology for knowledge representation in analogy solving. In ECCAI, editor, *European Conference on Artificial Intelligence (ECAI98)*, pages 88–92, Brighton, UK, 23–28 August 1998. Christian Rauscher.

[90] Erika Valencia and Jean-Louis Giavitto. Modeling diagrammatic reasoning tasks with a topological framework. In *TwD98 Workshop*, dic 1998.

[91] Erika Valencia and Jean-Louis Giavitto. Topologie algébrique pour la construction de représentations diagrammatiques. In Paris 8, editor, *7ème Colloque de l'Association pour la Recherche Cognitive*, Paris, France, dic 1998. ARC.

[92] Erika Valencia, Jean-Louis Giavitto, and Jean-Paul Sansonnet. Esqimo : Modelling analogy with topology. In Franck Ritter and Richard Young, editors, *Second European Conference on Cognitive Modelling (ECCM2)*, pages 212–213, Nottingham, UK, 1–4 April 1998. Nottingham University Press.

[93] Erika Valencia and Jean-Paul Sansonnet. Représentation simpliciale des connaissances et application à l'analogie. In *RAPC99*, pages 47–53, Palaiseau, Juin 1999. AFIA.

[94] Erika Valencia and Jean-Paul Sansonnet. Modélisation de l'hétérogénéité sémantique entre agents dialogiques. In *8èmes Journées Francophones Intelligence Artificielle Distribuée Systèmes Multi-Agents*, Saint-Jean la Vetre, France, Octobre 2000.

[95] Erika Valencia and Jean-Paul Sansonnet. Simplicial representation for description logics databases. In Ulrike Sattler Franz Baader, editor, *2000 International Workshop on Description Logics*, Aachen, Germany, August 2000. RWTH.

[96] Tony Veale. *Metaphor, Memory and Meaning : Symbolic and Connectionist Issues in Metaphor Interpretation*. PhD thesis, School of Computer Applications, Dublin City University, Glasnevin, Dublin, Ireland, October 1995.

[97] Tony Veale, Diarmuid O Donoghue, and Mark Keane. Computability as a limiting cognitive constraint : Complexity concerns in metaphor comprehension about which cognitive linguists should be aware. On the Metaphor Web Sit, Dublin City University, to be published in Cognitive Linguistics, 1996.

[98] D. Vernant. Du dire au faire : les niveaux d'analyse des phénomènes communicationnels. *RIHM*, 1(1) :11–32, 2000.

[99] K. von Luck, B. Nebel, C. Peltason, and A.Schmiedel. The anatomy of the back-system. Technical report, Department of Computer Science, Technische Universität Berlin, Berlin, Germany, 1987.

[100] S. H. Weber and A. Stolcke. l_0 : A testbed for miniature language acquisition. *International Computer Science Institute*, 1990.

[101] Peter C. Weinstein and William P.Birmingham. Comparing concepts in differentiated ontologies. In *KAW'99*, 1999.

[102] J.S. Weitzenfeld. Valid reasoning by analogy. *Philosophy of Science*, 1984.

[103] Rudolf Wille. Conceptual graphs and formal concept analysis. In D. Lukose, H. Delugach, M. Keeler, L. Searle, and J. F. Sowa, editors, *Conceptual structures : Fulfilling Peirce's dream*, number 1257 in Lecture Notes in Artificial Intelligence, Berlin–Heidelberg–New York, 1997. Springer–Verlag.

[104] Stephen Wolfram. *Mathematica*. Addison-Wesley, Redwood City, CA, 1988.