TABLE DES MATIERES

Avant-propos.....x

Remerciements.....xi

Introduction.....1

CHAPITRE 1 - Les outils pour formaliser.....5

1. Ensembles et Eléments de logique.....5

1.1. Ensemble.....6
1.1.1 Présentation intuitive.....6
1.1.2 Définition.....6
1.1.3 Création d'un ensemble ISETL.....7
1.1.4 Opérations.....8
1.1.4.1 Union.....8
1.1.4.2 Intersection.....9
1.1.4.3 Différence, Complémentaire.....9
1.1.5 Théorèmes.....10
1.1.6 Inclusion.....16
1.1.7 Parties d'ensemble.....18
1.1.8 Produit cartésien.....19
1.1.8.1 Projecteurs.....19
1.1.9 Les Opérateurs en langage ISETL.....20
1.1.10 Quelques exemples ISETL.....23

1.2. Logique des propositions.....27
1.2.1 Introduction.....27
1.2.1.1 Implication, Equivalence en ISETL.....32
1.2.2 Tables de vérité.....33
1.2.2.1 Ecriture des tables de vérité en ISETL.....34
1.2.3 Propriétés des connecteurs.....35
1.2.4 Une axiomatique.....36
1.2.4.1 Axiomes.....36
1.2.5 Tautologies.....38
1.2.6 Exercices.....40
1.2.6.1 Exercice n°1.....40

1.2.6.2 Exercice n°2 41
1.2.6.3 Exercice n°3 42

1.3. Logique des prédicats 42
1.3.1 Définition 42
1.3.2 Quantificateurs 42
1.3.3 Règles 44
1.3.4 Exemples 50
1.3.5 Exercices corrigés 52

CHAPITRE 2 - Complément sur les ensembles 61

I. Correspondances - Fonctions - Relations 61

1.1. Correspondances 61
1.1.1 Représentation d'une correspondance 62
1.1.1.1 Successeur et prédécesseur 63
1.1.2 Domaine, Image d'une correspondance 64
1.1.3 Les correspondances en ISETL: Les maps 64
1.1.3.1 Création 65
1.1.3.2 Les opérations 65
1.1.3.3 Exemple 66
1.1.4 Transposée d'une correspondance 67
1.1.5 Image d'un élément d'un sous-ensemble 68
1.1.6 Relation, Fonction, Application 69
1.1.6.1 Relation 69
1.1.6.2 Composition 70
1.1.6.3 Fonction, Application 73
1.1.6.4 Théorème 79
1.1.6.5 Exemples 81
1.1.7 Exercices corrigés 81

1.2. Relations binaires 83
1.2.1 Définition d'une relation binaire 83
1.2.1.1 Ensemble produit 83
1.2.1.2 Relation 83
1.2.2 Représentation d'une relation binaire 84
1.2.3 Composition de deux relations binaire 85
1.2.4 Propriétés des relations 87
1.2.4.1 Relation identité 87
1.2.4.2 Relation transposée 88
1.2.4.3 Relation d'ordre 89
1.2.4.3.1 Définition 89
1.2.4.3.2 Relation d'ordre totale 90
1.2.4.3.3 Relation d'équivalence 91
1.2.5 Relations binaires en ISETL 91
1.2.5.1 Réflexive 91

1.2.5.2 Antiréflexive..... 91
1.2.5.3 Non réflexive..... 92
1.2.5.4 Symétrique 92
1.2.5.5 Antisymétrique au sens large..... 92
1.2.5.6 Antisymétrique au sens stricte..... 93
1.2.5.7 Transitive..... 93
1.2.5.8 Un exemple en ISETL..... 93
1.2.6 Ensemble stable intérieurement maximal (ESIMS)..... 97
1.2.6.1 Définition..... 97
1.2.6.2 Algorithme de recherche des ESIMS..... 99
1.2.6.3 Application - Nombre chromatique d'un graphe..... 101

CHAPITRE 3 - Les Types Abstraits..... 105

1. Quelques types abstraits et les suites..... 105

1.1. Le type abstrait booléen..... 106

1.2. Le type abstrait entier 106

1.3. Le type abstrait ensemble 107

1.4. Les Listes 107
1.4.1 Création..... 108
1.4.2 Les opérations 109
1.4.3 Suites et couples..... 110
1.4.4 Exemples..... 111

1.5. Le type abstrait suite 111
1.5.1 Une première définition : l'ensemble S(A)..... 111
1.5.1.1 Structure..... 111
1.5.1.2 Opérations de base..... 112
1.5.2 Une deuxième définition : l'ensemble K(A)..... 113
1.5.2.1 Structure..... 113
1.5.2.2 Opérations de base..... 114
1.5.3 Equivalence des deux versions..... 115
1.5.4 Fonctions ajoutées aux primitives..... 117
1.5.4.1 Longueur d'une suite..... 119
1.5.4.2 Extraction d'un élément..... 120
1.5.4.3 Concaténation de deux suites..... 122
1.5.4.4 Identité de deux suites..... 123
1.5.4.5 La fonction map..... 124
1.5.4.6 La fonction filter..... 125
1.5.4.7 La fonction member..... 125
1.5.4.8 La fonction foldl (pliage gauche)..... 126
1.5.4.9 La fonction foldr (pliage droit)..... 126

1.6. Les suites en ISETL 127
1.6.1 La structure 127
1.6.2 De l'algorithme à la programmation 127
1.6.3 Les opérations 134
1.6.4 Machine à états pour les suites 138
1.6.4.1 La fonction factorielle 139
1.6.4.1.1 Mae pour la fonction factorielle 139
1.6.4.1.2 Algorithme 140
1.6.4.2 Mae pour la fonction length 140
1.6.4.2.1 Définition de la mae 140
1.6.4.2.2 Mae réduite 142
1.6.5 Deux exercices détaillé 144
1.6.5.1 Plus grand élément d'une suite d'entiers 144
1.6.5.2 Première occurrence d'un élément dans une suite 147
1.6.6 Equations remarquables 152
1.6.7 Un pliage particulier: map 153

1.7. La currification 154
1.7.1 Définition 154
1.7.2 Exemple 154
1.7.3 Réalisation du type abstrait suite en ISETL 155
1.7.4 Exemples d'utilisation du module suite avec fonctions currifiées 156

CHAPITRE 4 - Programmer en langage Clean 159

I. Programmer avec des fonctions 160

1.1. L'expression Start en Clean 160
1.1.1 Un premier exemple .....
1.1.2 Un second exemple 161
1.1.3 Un troisième exemple 162

1.2. Les types de bases en Clean 162
1.2.1 Le type fonction (ou flèche) .....

1.3. Définir des fonctions simples 163
1.3.1 La fonction carré 163
1.3.2 La fonction double 164
1.3.3 La fonction est_pair 164

1.4. La currification 164
1.4.1 Définition 164
1.4.2 Exemple 164

1.5. Quelques fonctions standards sur les types simples 166
1.5.1 Pour le type Int 166

1.5.2 Pour le type Real..... 166
1.5.3 Pour le type Bool..... 167
1.5.4 Pour le type Char..... 167

1.6. Définir des fonctions..... 167
1.6.1 Par combinaison de fonctions existantes..... 167
1.6.2 Par cas..... 168
1.6.3 Par induction ou récursion..... 169

1.7. Types abstraits..... 170
1.7.1 Les opérations de base disponibles en Clean pour les suites 171

1.8. Codage des fonctions ajoutées aux primitives du type abstrait suite..... 172
1.8.1 Longueur d'une suite..... 172
1.8.2 Extraction d'un élément..... 173
1.8.3 Identité de deux suites..... 173
1.8.4 Concaténation de deux suites..... 174
1.8.5 La fonction map..... 174
1.8.6 La fonction filter..... 175
1.8.7 La fonction fold_l(pliage gauche)..... 175
1.8.7.1 Pliage gauche généralisé..... 176
1.8.8 La fonction fold_r(pliage droit) 176
1.8.8.1 Pliage droit généralisé..... 176

Intégrez vos propres modules*..... *177

2.1. Création des fichiers de définition et d'implémentation..... 177
2.1.1 Le fichier d'implémentation..... 177
2.1.2 Le fichier de définition..... 179

2.2. Classes et instances..... 180
2.2.1 Déclaration d'une classe et des instances 180
2.2.2 Instanciation des opérateurs matriciels... 182
2.2.2.1 Instance sur des opérateurs existants..... 183
2.2.2.1.1 Le Produit matriciel 183
2.2.2.1.2 Somme matricielle 184
2.2.3 Classes appliquées aux matrices..... 184
2.2.3.1 Classe "CalculScalaire" et instance..... 184
2.2.3.2 Classe "ProduitScalaire" et instance..... 185

2.3. Application..... 185

Un exercice sur les graphes*..... *185

Un exercice - Le concept de module - une réalisation des ensembles par les suites*.. *186

Un exemple de structure - le type suite*..... *189

6. Types enregistrements 189

6.1. Déclaration d'objets de type enregistrement 190

6.2. Déclaration de fonctions à paramètres de type enregistrement 191

CHAPITRE 5 - Programmer en langage Haskell 193

I. Programmer avec l'interpréteur (Hugs) 193

1.1. Définition de fonctions 196
1.1.1. Un premier exemple 196
1.1.2. Un second exemple 198
1.1.3. Un troisième exemple 200

1.2. L'expression Main en Hugs 201
1.2.1 Un premier exemple 202
1.2.2 Un second exemple 202
1.2.3 Un troisième exemple 202

1.3. Les types simples 203
1.3.1 Les types de bases 203
1.3.2 Le type fonction (ou flèche) 203
1.3.3 Structures de données prédéfinies 204

1.4. Définir des fonctions simples 204
1.4.1 La fonction carré 204
1.4.2 La fonction double 205
1.4.3 La fonction est_pair 205

1.5. Quelques fonctions standards sur les simples 205
1.5.1 Pour le type Int 205
1.5.2 Pour le type Real 205
1.5.3 Pour le type Bool 206
1.5.4 Pour le type Char 206

1.6. Composition de fonctions 206
1.6.1 Afficher une expression la classe Show 207

1.7. Le type abstrait suite 208
1.7.1 Les opérations de base disponibles en Hugs pour les suites 208
1.7.2 Une réalisation du type abstrait suites 209

1.8. Codage des fonctions ajoutées aux primitives du type abstrait suite 211
1.8.1 Longueur d'une suite 211
1.8.2 Extraction d'un élément 212
1.8.3 Identité de deux suites 212

1.8.4 Concaténation de deux suites..... 213
1.8.5 La fonction map..... 213
1.8.6 La fonction filter (filtrage des éléments d'une suite)..... 214
1.8.7 La fonction fold_l (pliage gauche)..... 214
1.8.8 La fonction fold_r (pliage droit)..... 215

1.9. La currification..... 216
1.9.1 Définition..... 216
1.9.2 Exemple..... 216

HAPITRE 6 - Compléments et Applications..... 219

Plier*..... *219

1.1. Pliage gauche..... 219
1.1.1 Algorithme..... 220
1.1.2 Exemple..... 220
1.1.2.1 Pliage gauche généralisé..... 220

1.2. Pliage droit..... 221
1.2.1 Algorithme..... 222
1.2.2 Exemple..... 222
1.2.2.1 Pliage droit généralisé..... 222

Les suites en ZF notation*..... *223

2.1. La ZF notation en langage Clean..... 223
2.1.1 Pour les suites : [f x \\ x <- s | P x]..... 223
2.1.2 Pour les tableaux : { f x \\ x <-: t | P x}..... 223
2.1.3 Transformation : suite tableau, tableau suite..... 224
2.1.4 Complément sur les tableaux..... 225
2.1.5 La fonction filter (filtrage des éléments d'une suite) en ZF notation..... 226
2.1.6 La fonction map en ZF notation..... 226

2.2. La ZF notation en langage Haskell (Hugs)..... 227
2.2.1 les suites : [f x | x <- s , P x]..... 227
2.2.2 La fonction filter (filtrage des éléments d'une suite) en ZF notation..... 227
2.2.3 La fonction map en ZF notation..... 228

Les tableaux en langage Haskell (Hugs)*..... *229

1.1. Le type tableau: Array..... 229

1.2. Création d'un tableau..... 229

1.3. Les fonctions de base..... 229

3.4. Quelques autres fonctions 230

4. Trier 236

4.1. Définition 236

4.2. Tri par insertion 236
4.2.1 insertion d'un élément x dans une suite triée 236
4.2.2 tri par insertion 238

4.3. Sélection du plus grand élément (spg) 239
4.3.1 sélection et extraction du plus grand élément (spg) 240

4.4. Tri par sélection 242

4.5. Tri par partition 244
4.5.1 Principe du partitionnement de deux suites 244
4.5.1.1 Sans utiliser la ZF notation 244
4.5.1.2 En utilisant la ZF notation 245
4.5.2 Tri par partition 246

4.6. Tri par fusion 249

5. Relations binaires 249

5.1. Les relations binaires en langage Clean 249
5.1.1 Une autre réalisation du type abstrait ensemble 259
5.1.1.1 Exemples d'utilisations 261
5.1.1.2 Une autre réalisation du type abstrait graphe 264

5.2. Les relations binaires en langage Haskell (Hugs) 268

Quelques Applications 274

6.1. Inversion d'une image noir et blanc 274

6.2. Concaténation des suites composants une suite 278

6.3. Tri par sélection du plus petit élément d'une suite 278

6.4. Partition d'une suite de chaînes de caractères contenant ou non la lettre "d" 281

6.5. Addition des triples de tous les éléments multiple de 5 d'une suite 281

6.6. Equivalent d'un pliage gauche pour le type tableau 283

6.7. Elément membre d'une suite 286

6.8. Somme des éléments d'une suite..... 286

6.9. Eléments pairs d'une suite..... 286

6.10. Recherche du plus grand élément d'une suite.....287

6.11. Ajout d'un élément en fin de suite..... 287

6.12. Indice de la première occurrence d'un élément dans une suite.....288

6.13. Traduction d'une suite en chaîne de caractères..... 289

6.14. Dernière occurrence d'un élément x dans une suite.....290

6.15. Automates finis déterministes, à l'aide du type tableau.....290
6.15.1 Une approche en Clean - Le type tableau en Clean 290
6.15.2 Etude d'une fonction associant à l'ensemble des caractères une valeur.... 291
6.15.3 Ecrire un automate détectant les nombres entiers et décimaux (réels)..... 293
6.15.4 Ecrire un automate détectant une chaine commençant par a ou b contenant les lettres u et p collées et se terminant par z suivi d'un espace..... 297

IAPITRE 7 - Etude de cas et programmation Clean - Gestion des Inscriptions des Auditeurs du conservatoire National des Arts et Métiers..... 301

Une première approche 301

1.1. L'enregistrement des inscriptions..... 301
1.1.1 La fiche-auditeur..... 301
1.1.2 Exemples de fiches d'inscription..... 302

1.2. Les outils d'extractions..... 304
1.2.1 La fonction nth..... 304
1.2.2 Extraire le numéro du $i^{ème}$ élément du fichier..... 305
1.2.3 Extraire le nom du $i^{ème}$ élément du fichier..... 305
1.2.4 Extraire l'âge du $i^{ème}$ élément du fichier..... 305
1.2.5 Extraire les couples (UV,note) du $i^{ème}$ élément du fichier..... 306

1.3. Les fonctions du cahier des charges..... 306
1.3.1 Combien d'auditeurs sont inscrits ? 306
1.3.2 Combien d'auditeurs sont inscrits à au moins n UV ? 307
1.3.3 Ensemble des couples (UV,notes) 307
1.3.4 Nombres d'UV que suit un auditeur..... 308
1.3.5 Inscriptions rejetés (auditeurs inscrits à aucune UV). 308
1.3.6 Quelles sont les UV ouvrables. 308
1.3.7 Liste des UV, sans doublon. 309
1.3.8 La fonction nb_uv, nombre d'inscrit à telle UV..... 310

1.3.9 Lisibilité. 311
1.3.10 La moyenne d'âge des auditeurs..... 311
1.3.11 La moyenne des notes d'une UV donnée. 311
1.3.12 Eliminer les fiches des auditeurs inscrits à aucune UV. 312

1.4. Quelques autres fonctions..... 312
1.4.1 Donner la Fiche d'un inscrit à partir de son indice..... 312
1.4.2 La fonction uv_String..... 313
1.4.3 Donner la fiche de l'élève ayant le numéro n..... 313
1.4.4 Donner la fiche de l'élève X..... 314
1.4.5 Donner la liste des personnes inscrites à une UV donnée..... 315

2. Une seconde approche 316

2.1. Modélisation..... 316
2.1.1 Les auditeurs..... 316

2.2. Extraction des enregistrements..... 318

2.3. Création de fonctions pour le traitement des données..... 319
2.3.1 Extraction de la liste complète des auditeurs : la fonction nomaudit..... 319
2.3.2 Extraction de la liste complète des numéros d'inscription des auditeurs : la fonction numaudit..... 322
2.3.3 Recherche du nom de l'auditeur à partir du numéro d'inscription et réciproquement. 323
2.3.4 Extraction de la liste complète des unités de valeurs des auditeurs..... 325
2.3.5 Inscription des auditeurs à au moins une unité de valeur..... 327
2.3.6 Rejet des auditeurs non inscrits à au moins une unité de valeur..... 328
2.3.7 Le traitement des notes..... 331
2.3.8 Calcul de la moyenne obtenue par un auditeur. 337
2.3.9 Le traitement de l'âge. 341

CHAPITRE 8 - Etude de cas et programmation Haskell - Réalisation d'un logiciel de gestion clientèle..... 345

1. Présentation du projet..... 345

1.1. Enoncé du projet..... 345
1.1.1 Présentation du module ensembliste "set.hs"..... 345

1.2. Cahier des charges..... 348

1.3. Objectifs du module "gestion.hs"..... 349

1.4. Convention de notation..... 349

Structuration des données.....*350*

2.1. L'ensemble "clients".....350
2.1.1 Analyse du problème.....350
2.1.2 Structure de l'ensemble.....350
2.1.3 Codification en Hugs.....351

2.2. L'ensemble "Produits".....351
2.2.1 Analyse du problème.....351
2.2.2 Structure de l'ensemble.....351
2.2.3 Codification en Hugs.....352

2.3. L'ensemble "Commandes".....352
2.3.1 Analyse du problème.....352
2.3.2 Structure de l'ensemble.....352
2.3.3 Codification en Hugs.....353

Les fonctions "tri" et "ins".....*353*

3.1. La fonction "tri".....353
3.1.1 Analyse du problème.....353
3.1.2 Signature et Algorithme.....354
3.1.3 Codification en Hugs.....354

3.2. La fonction "ins".....354
3.2.1 Signature.....355
3.2.2 Spécification.....355
3.2.3 Problème de la hiérarchisation des trois clés (c1, c2, c3).....355
3.2.4 Traduction en algorithme.....356
3.2.5 Codification en Hugs.....356

Les fonctions manipulant l'ensemble "clients".....*357*

4.1. "trinp": Tri des clients par Nom et Prénom.....357
4.1.1 Signature et Algorithme.....357
4.1.2 Codification en Hugs.....357

4.2. "trivn": Tri des clients par Ville et par Nom.....357
4.2.1 Signature et Algorithme.....357
4.2.2 Codification en Hugs.....358

4.3. "clivil": Ensemble des clients d'une ville donnée.....358
4.3.1 Signature et Algorithme.....358
4.3.2 Codification en Hugs.....359

Les fonctions manipulant l'ensemble "produit".....*359*

5.1. "totpro": Nombre total des produits fabriqués par l'entreprise.....359

5.1.1 Signature et Algorithme
5.1.2 Codification en Hugs

5.2. "triref": Tri des produits par référence
5.2.1 Signature et Algorithme
5.2.2 Codification en Hugs

5.3. "trides": Tri des produits par désignation
5.3.1 Signature et Algorithme
5.3.2 Codification en Hugs

5.4. "triprix" : Tri des produits par prix
5.4.1 Signature et Algorithme
5.4.2 Codification en Hugs

5.5. "prixttc": Prix ttc d'un produit suivant sa référence
5.5.1 Signature et Algorithme
5.5.2 Explication de l'algorithme
5.5.3 Codification en Hugs

6. *Les fonctions manipulant l'ensemble "commandes"*

6.1. "tricom": Tri alphabétique des clients avec leurs commandes
6.1.1 Signature et Algorithme
6.1.2 Codification en Hugs

6.2. "procli": Liste des produits commandés par un client
6.2.1 Signature et Algorithme
6.2.2 Explication de l' algorithme
6.2.3 Codification en Hugs

6.3. "ttccli": Total TTC pour la commande d'un client
6.3.1 Analyse du problème
6.3.2 Signature et Algorithme
6.3.3 Explication de l'algorithme
6.3.4 Codification en Hugs

6.4. "chifvent" : Chiffre de vente de l'entreprise
6.4.1 Signature et Algorithme
6.4.2 Explication de l'algorithme
6.4.3 Codification en Hugs

6.5. "facture" : Facture pour un client
6.5.1 Analyse du problème
6.5.2 Définition de la structure des données de type "facture"
6.5.3 Signature
6.5.4 Algorithme
6.5.5 Explication de l'algorithme

6.5.6 Codification en Hugs.....370
6.5.7 Explication détaillée de l'algorithme.....371

6.6. "reappro" : Liste des produits à réapprovisionner(ref,quantité) 371
6.6.1 Analyse du problème.....371
6.6.2 Signature et algorithme.....372
6.6.3 Codification en Hugs.....372
6.6.4 Explication détaillée de l'algorithme.....373

Les fonctions d'initialisation des ensembles. *374*

7.1. "init_cli": Initialiser l'ensemble"clients".....374

7.2. "init_pro": Initialiser l'ensemble "produits".....374

7.3. "init_com": Initialiser l'ensemble "commandes".....374

Les fonctions d'affichage (Hugs) *375*

8.1. "showsC": Affichage de l'ensemble "clients".....375

8.2. "showsP": Affichage de l'ensemble "produits".....375

8.3. "showsCO": Affichage de l'ensemble "commandes".....375

8.4. "showsLC": Affichage d'une liste de clients.....376

8.5. "showsLP": Affichage d'une liste de produits.....376

8.6. "showsLCO": Affichage d'une liste de commandes.....376

8.7. "showsLR": Affichage d'une liste de réapprovisionnement. 376

8.8. "showsCC": Affichage de la commande d'un client 377

8.9. "showsF": Affichage d'une facture. 377

APITRE 9 - Programmer en langage Poly/ML..... 381

La structure PolyML. *381*

1.1. Démarrage d'une session Poly/ML.....381

1.2. Quitter une session Poly/ML.....382

1.3. Inclure des fichiers sources.....382
1.3.1 Un premier exemple : la fonction iterer.....383

2. *Liaisons et environnements* 38
3. *Expressions - Evaluation des expressions.* 38
3.1. Expressions Numériques 38
3.2. Expressions Booléennes 38
3.3. Extension de la notion d'expression 386
3.4. Exercices 387
4. *Identificateurs locaux et globaux.* 387
4.1. Liaisons globales 387
4.2. Déclarations globales simultanées 387
4.3. Liaisons locales 388
4.4. Exercice 389
4.5. Déclarations locales simultanées 389
4.6. Déclarations locales emboitées 390
4.7. Exercices 390
5. *Expressions conditionnelles* 391
6. *Poly/ML et la synthèse de type* 391
6.1. Les types de base de Poly/ML 392
6.1.1 Opérateurs infixés 392
6.1.2 Booléens et fonctions de comparaison 393
6.1.3 Entiers et Réels 393
6.1.3.1 Conversion de type 393
6.1.3.2 Fonctions arithmétiques 394
6.1.3.3 Relations d'ordre 395
6.1.3.4 Fonctions prédéfinies sur le type List 395
6.1.3.5 Fonctions sur le type string 395
6.1.3.6 Opérations sur le type ref 397
6.1.3.7 La fonction o 397
6.1.3.8 Quelques exercices 398
7. *Les fonctions en PolyML.* 399
7.1. Fonctions récursives 400

7.1.1 Terminaison d'un traitement récursif..... 401

7.2. Fonctions à paramètres fonctionnels..... 401

7.3. Fonctions à plusieurs paramètres, récursivité et fonctions locales.....402

7.4. Filtrage par motifs "Pattern matching".....405

7.5. Fonctions polymorphes.....405

7.6. Notion de fermeture..... 406

7.7. Gestion des préconditions - exceptions..... 406
7.7.1 Fonctions totales et fonctions partielles..... 406
7.7.2 Utilisation des exceptions..... 407

7.8. Effets de bord et objets mutables.....408
7.8.1 Effets de bord.....408
7.8.2 Objets mutables..... 409
7.8.3 Risques liés aux effets de bord.....411

7.9 Quelques exercices.....412

Le type suite en PolyML. 419

8.1. Fonctions de manipulation de suites.....420
8.1.1 La fonction length.....420
8.1.2 Nième élément d'une suite..... 421
8.1.3 Appartenance d'un élément à une suite. 422
8.1.4 Application d'une fonction sur chaque élément d'une suite. 423
8.1.5 Plier à gauche.....424
8.1.6 Plier à droite.....426
8.1.7 Pliage gauche généralisé.....426
8.1.8 Pliage droit génerlisé.....427
8.1.9 filtrer..... 428

8.2. Utilité des fonctions auxiliaires.....429

8.3. Exercices.....430

Trier en PolyML. 434

9.1. Insertion d'un élément x dans une suite triée.....434

9.2. Tri par partition.....435

9.3. Tri par sélection.....437

9.4. Tri par fusion 43
9.4.1 définition des fonctions take et drop. 43
9.4.2 Tri par fusion 43

10. Le type ensemble en Poly/ML. 43

11. Automates finis en PolyML. 443

11.1. Tableaux et Vecteurs 443
11.1.1 Les Vecteurs. 443
11.1.2 Les Tableaux. 444
11.1.3 Exemples. 445

11.2. Automate détectant les nombres entiers et réels 448

11.3. Automate détectant une chaîne commençant par "a" ou "b" contenant les lettres "u" et "p" collées et se terminant par "z" suivi d'un espace. 449
11.3.1 Sous forme de suites 451

12. Types construits 452

12.1. Types produits. 452
12.1.1 Généralisation : produit cartésien de deux types. 452
12.1.2 Introduction de types produits dans les fonctions. 453
12.1.3 Exercices 454

12.2. Types enregistrements. 455
12.2.1 Déclaration d'objets de type enregistrement 455
12.2.2 Déclaration de fonctions dont le paramètre est de type enregistrement 456

13. Créez vos structures. 456

14. Petite étude de cas - Gestion de comptes bancaires. 462

14.1. Définition du domaine d'étude 463

14.2. Transcription en langage Poly/ML 464

15. Processus fonctionnels en Poly/ML 470

15.1. La structure process 470
15.1.1 Exemples de déclaration 471
15.1.2 Exemple applicatif de deux processus non concurrents (indépendants) 471
15.1.3 Processus concurrents 473
15.1.4 Synchronisation entre processus 477
15.1.5 Quelques exemples applicatifs 478
15.1.6 Deux processus effectuant un produit matriciel 484

IAPITRE 10 - Les Arbres en fonctionnel..... 487

Structure d'arbre.....489

1.1. Arbres de Type 1..... 489
1.1.1 Opérations sur cette structure..... 490

1.2. Arbres de Type 2..... 491
1.2.1 Opérations sur le type 2..... 492
1.2.2 Une réalisation en Haskell..... 492

1.3. Arbres de type 3..... 494

1.4. Arbres de recherche.....495
1.4.1 Insertion d'une clé dans un arbre binaire de recherche..... 495
1.4.2 Hauteur d'un arbre.....497

1.5. Les 2-3 arbres..... 498
1.5.1 Opérations sur les 2-3 arbres.....499
1.5.2 Transformation d'une suite en arbre.....504
1.5.3 Transformation d'un 2-3 arbre en suite..... 505
1.5.4 Une réalisation en Clean..... 505

1.6. Les AVL (arbres binaires de recherche quasi équilibrés)..... 508
1.6.1 Hauteur d'un AVL.....509
1.6.2 Pente d'un AVL..... 509
1.6.3 Insertion d'un élément dans un AVL.....509
1.6.4 Rééquilibrage d'un AVL..... 509
1.6.5 Transformation d'une suite en AVL.....512
1.6.6 Une réalisation en Clean..... 514
1.6.7 Exemple d'arbres binaires en Poly/ML.....517

bliographie..... 523

NEXES.....

nnexe n° 1 : *Un interpreteur PROLOG en langage ISETL*..... 527

nnexe n° 2 : *Programmation en langage CLEAN - module matriciel implémentation et définition* 533

nnexe n° 3 : *Programmation en langage CLEAN - GESTION DES AUDITEURS DU CNAM (module implémentation)* 539

nnexe n° 4 : *Programmation en langage CLEAN - GESTION DES AUDITEURS DU CNAM (module définition)* 543

- Annexe n° 5 : *Programmation en langage CLEAN - GESTION DES AUDITEURS DU CNAM (module d'exécution)* 545

- Annexe n° 6 : *Programmation en langage CLEAN - Une seconde approche du Programme de GESTION DES AUDITEURS DU CNAM*..... 549

- Annexe n° 7 : *Le module Set en Haskell*..... 559

- Annexe n° 8 : *Programmation en langage Haskell - LOGICIEL DE GESTION CLIENTELE*.....563