



A. Aho, M. Lam, R. Sethi et J. Ullman

Compilateurs

Principes, techniques et outils

2^e édition

Avec plus de 200 exercices



PEARSON
Education

Table des matières

Préface	xxi
Le mot des traducteurs	xxv
1 Introduction	1
1.1 Processeurs de langages	1
Exercices	3
1.2 Structure d'un compilateur	4
1.2.1 Analyse lexicale	5
1.2.2 Analyse syntaxique	6
1.2.3 Analyse sémantique	8
1.2.4 Production de code intermédiaire	9
1.2.5 Optimisation de code	9
1.2.6 Production de code	10
1.2.7 Gestion de la table de symboles	10
1.2.8 Regroupement des phases en passes	11
1.2.9 Outils de construction de compilateurs	11
1.3 Évolution des langages de programmation	12
1.3.1 Vers des langages de plus haut niveau	12
1.3.2 Conséquences pour les compilateurs	13
Exercices	14
1.4 De la science en compilation	14
1.4.1 Modèles pour la conception et la réalisation de compilateurs	14
1.4.2 De la science en optimisation de code	15
1.5 Applications de la technologie des compilateurs	16
1.5.1 Implémentation de langages de programmation de haut niveau	16
1.5.2 Optimisations dépendant de l'architecture	18
1.5.3 Conception de nouvelles architectures	19
1.5.4 Traduction de programme	20
1.5.5 Outils de productivité	22
1.6 Notions de base des langages de programmation	23
1.6.1 Différence entre statique et dynamique	23
1.6.2 Environnements et états	24
1.6.3 Portée statique et structure de blocs	26
1.6.4 Contrôle d'accès explicite	28
1.6.5 Portée dynamique	29

1.6.6	Mécanismes de passage de paramètres	31
1.6.7	Synonymie	32
	Exercices	32
	Résumé	33
	Bibliographie	35
2	Un traducteur simple en une passe	37
2.1	Introduction	37
2.2	Définition de la syntaxe	40
2.2.1	Définition des grammaires	41
2.2.2	Dérivations	42
2.2.3	Arbres d'analyse	43
2.2.4	Ambiguïté	45
2.2.5	Associativité des opérateurs	45
2.2.6	Priorité des opérateurs	46
	Exercices	49
2.3	Traduction dirigée par la syntaxe	50
2.3.1	Notation postfixée	50
2.3.2	Attributs synthétisés	51
2.3.3	Définitions simples dirigées par la syntaxe	53
2.3.4	Parcours d'arbres	54
2.3.5	Schémas de traduction	55
	Exercices	57
2.4	Analyse syntaxique	57
2.4.1	Analyse syntaxique descendante	58
2.4.2	Analyse syntaxique prédictive	60
2.4.3	Quand utiliser les ε -productions ?	62
2.4.4	Développer un analyseur syntaxique prédictif	63
2.4.5	Récursivité à gauche	63
	Exercices	65
2.5	Un traducteur pour expressions simples	65
2.5.1	Syntaxe abstraite et syntaxe concrète	66
2.5.2	Adapter le schéma de traduction	66
2.5.3	Procédures pour les non-terminaux	68
2.5.4	Simplifier le traducteur	69
2.5.5	Le programme complet	69
2.6	Analyse lexicale	72
2.6.1	Élimination des blancs et des commentaires	73
2.6.2	Pré-vision	73
2.6.3	Constantes	74
2.6.4	Reconnaître les mots clés et les identificateurs	74
2.6.5	Un analyseur lexical	76
	Exercices	79
2.7	Tables de symboles	80
2.7.1	Une table de symboles par portée	81
2.7.2	Utilisation des tables de symboles	84
2.8	Production de code intermédiaire	86
2.8.1	Deux types de représentations intermédiaires	86

2.8.2	Construction des arbres abstraits	87
2.8.3	Vérification statique	91
2.8.4	Code à trois adresses	93
	Exercices	98
	Résumé	99
3	Analyse lexicale	101
3.1	Le rôle de l'analyseur lexical	101
3.1.1	Analyse lexicale et analyse syntaxique	103
3.1.2	Unités lexicales, motifs et lexèmes	103
3.1.3	Attributs d'une unité lexicale	104
3.1.4	Erreurs lexicales	105
	Exercices	106
3.2	Mise en mémoire tampon de l'entrée	106
3.2.1	Couples de tampons	107
3.2.2	Sentinelles	108
3.3	Spécification des unités lexicales	109
3.3.1	Chaînes et langages	110
3.3.2	Opérations sur les langages	111
3.3.3	Expressions régulières	111
3.3.4	Définitions régulières	113
3.3.5	Extensions des expressions régulières	114
	Exercices	115
3.4	Reconnaissance des unités lexicales	118
3.4.1	Diagrammes de transition	119
3.4.2	Reconnaissance des mots réservés et des identificateurs	121
3.4.3	Finalisation du traitement de l'exemple récurrent	122
3.4.4	Architecture d'un analyseur lexical reposant sur des diagrammes de transition	123
	Exercices	125
3.5	Le constructeur d'analyseurs lexicaux Lex	129
3.5.1	Utilisation de Lex	129
3.5.2	Structure d'un programme Lex	130
3.5.3	Résolution de conflits dans Lex	132
3.5.4	L'opérateur de pré-vision	133
	Exercices	134
3.6	Automates finis	135
3.6.1	Automates finis non déterministes	135
3.6.2	Tables de transition	136
3.6.3	Acceptation de chaînes d'entrée par des automates	137
3.6.4	Automates finis déterministes	137
	Exercices	139
3.7	Des expressions régulières aux automates	140
3.7.1	Conversion d'un AFN en AFD	140
3.7.2	Simulation d'un AFN	143
3.7.3	Efficacité de la simulation d'un AFN	144
3.7.4	Construction d'un AFN à partir d'une expression régulière	146
3.7.5	Efficacité des algorithmes de traitement des chaînes	149

	Exercices	152
3.8	Développement d'un constructeur d'analyseurs lexicaux	153
3.8.1	Structure de l'analyseur produit	153
3.8.2	Reconnaissance de motifs à l'aide d'AFN	154
3.8.3	AFD pour les analyseurs lexicaux	156
3.8.4	Implémenter l'opérateur de pré-vision	157
	Exercices	158
3.9	Optimisation des reconnaisseurs de motifs à base d'AFD	158
3.9.1	États importants d'un AFN	159
3.9.2	Fonctions calculées à partir de l'arbre abstrait	160
3.9.3	Calcul de <i>annulable</i> , <i>premierePos</i> et <i>dernierePos</i>	161
3.9.4	Calcul de <i>posSuivante</i>	162
3.9.5	Conversion directe d'une expression régulière en un AFD	164
3.9.6	Minimisation du nombre d'états d'un AFD	165
3.9.7	Minimisation du nombre d'états dans les analyseurs lexicaux	168
3.9.8	Compromis temps–espace pour la simulation des AFD	169
	Exercices	170
	Résumé	170
	Bibliographie	172
4	Analyse syntaxique	175
4.1	Introduction	176
4.1.1	Rôle de l'analyseur syntaxique	176
4.1.2	Grammaires de référence	178
4.1.3	Gestion des erreurs de syntaxe	178
4.1.4	Stratégies de rattrapage d'erreurs	180
4.2	Grammaires non contextuelles	181
4.2.1	Définition formelle des grammaires non contextuelles	181
4.2.2	Conventions de notation	182
4.2.3	Dérivations	183
4.2.4	Arbres d'analyse et dérivations	185
4.2.5	Ambiguïté	187
4.2.6	Contrôler le langage défini par une grammaire	188
4.2.7	Grammaires non contextuelles et expressions régulières	189
	Exercices	190
4.3	Écriture d'une grammaire	192
4.3.1	Analyse lexicale et analyse syntaxique	192
4.3.2	Éliminer l'ambiguïté	193
4.3.3	Élimination de la récursivité à gauche	194
4.3.4	Factorisation à gauche	197
4.3.5	Constructions qui ne sont pas non contextuelles	198
	Exercices	198
4.4	Analyse syntaxique descendante	199
4.4.1	Analyse syntaxique par descente récursive	201
4.4.2	PREMIER et SUIVANT	202
4.4.3	Grammaires LL(1)	204
4.4.4	Analyse syntaxique prédictive non récursive	208
4.4.5	Rattrapage d'erreur en analyse syntaxique prédictive	210

	Exercices	212
4.5	Analyse syntaxique ascendante	214
	4.5.1 Réductions	215
	4.5.2 Élagage des manches	215
	4.5.3 Analyse syntaxique par décalage-réduction	217
	4.5.4 Conflits pendant l'analyse syntaxique par décalage-réduction	219
	Exercices	221
4.6	Introduction à l'analyse syntaxique LR : LR simple	221
	4.6.1 Pourquoi les analyseurs syntaxiques LR ?	221
	4.6.2 Les items et l'automate LR(0)	223
	4.6.3 L'algorithme LR d'analyse syntaxique	227
	4.6.4 Construction des tables d'analyse syntaxique SLR	231
	4.6.5 Préfixes viables	234
	Exercices	236
4.7	Des analyseurs syntaxiques LR plus puissants	238
	4.7.1 Items canoniques LR(1)	238
	4.7.2 Construction des ensembles d'items LR(1)	239
	4.7.3 Tables d'analyse syntaxique LR(1) canonique	243
	4.7.4 Construction des tables d'analyse syntaxique LALR	244
	4.7.5 Construction efficace des tables d'analyse syntaxique LALR	248
	4.7.6 Compression des tables d'analyse syntaxique LR	252
	Exercices	254
4.8	Utiliser des grammaires ambiguës	255
	4.8.1 Priorité et associativité pour la résolution des conflits	255
	4.8.2 L'ambiguïté du « sinon en suspens »	257
	4.8.3 Rattrapage d'erreur en analyse syntaxique LR	259
	Exercices	261
4.9	Constructeurs d'analyseurs syntaxiques	262
	4.9.1 Le constructeur d'analyseurs syntaxiques Yacc	262
	4.9.2 Utiliser Yacc avec des grammaires ambiguës	266
	4.9.3 Création d'analyseurs lexicaux Yacc à l'aide de Lex	269
	4.9.4 Rattrapage d'erreurs en Yacc	270
	Exercices	271
	Résumé	272
	Bibliographie	274
5	Traduction dirigée par la syntaxe	277
5.1	Définitions dirigées par la syntaxe	278
	5.1.1 Attributs hérités et synthétisés	278
	5.1.2 Évaluation d'une DDS aux nœuds d'un arbre d'analyse	280
	Exercices	283
5.2	Ordres d'évaluation pour les DDS	283
	5.2.1 Graphes de dépendances	284
	5.2.2 Ordre d'évaluation des attributs	285
	5.2.3 Définitions S-attribuées	286
	5.2.4 Définitions L-attribuées	287
	5.2.5 Règles sémantiques avec effets de bord contrôlés	288
	Exercices	290

5.3	Applications de la traduction dirigée par la syntaxe	290
5.3.1	Construction des arbres abstraits	291
5.3.2	Structure d'un type	294
	Exercices	295
5.4	Schémas de traduction dirigés par la syntaxe	296
5.4.1	Schémas de traduction postfixes	296
5.4.2	Implémentation des STDS postfixes par la pile d'analyse	297
5.4.3	STDS avec des actions à l'intérieur des productions	299
5.4.4	Élimination des récursivités à gauche dans les DDS	300
5.4.5	STDS pour définitions L-attribuées	302
	Exercices	307
5.5	Implémentation de DDS L-attribuées	308
5.5.1	Traduction au cours d'une analyse par descente récursive	309
5.5.2	Production de code à la volée	311
5.5.3	DDS L-attribuées et analyse LL	313
5.5.4	Analyse ascendante de DDS L-attribuées	318
	Exercices	321
	Résumé	322
	Bibliographie	323
6	Production de code intermédiaire	325
6.1	Variantes des arbres abstraits	326
6.1.1	Graphes orientés acycliques pour les expressions	326
6.1.2	Méthode des nombres de valeur pour la construction des DAG	328
	Exercices	330
6.2	Code à trois adresses	331
6.2.1	Adresses et instructions	331
6.2.2	Quadruplets	333
6.2.3	Triplets	334
6.2.4	Forme à affectation statique unique	336
	Exercices	337
6.3	Types et déclarations	337
6.3.1	Expressions de types	338
6.3.2	Équivalence de types	339
6.3.3	Déclarations	340
6.3.4	Organisation de la mémoire pour les noms locaux	340
6.3.5	Suites de déclarations	342
6.3.6	Champs dans les structures et les classes	343
	Exercices	344
6.4	Traduction des expressions	345
6.4.1	Opérations dans les expressions	345
6.4.2	Traduction incrémentale	346
6.4.3	Adressage des éléments de tableau	347
6.4.4	Traduction des références aux tableaux	349
	Exercices	351
6.5	Contrôle de type	352
6.5.1	Règles pour le contrôle de type	352
6.5.2	Conversions de types	353

6.5.3	Surcharge des fonctions et des opérateurs	355
6.5.4	Inférence de type et fonctions polymorphes	356
6.5.5	Un algorithme d'unification	359
	Exercices	362
6.6	Flot de contrôle	363
6.6.1	Expressions booléennes	363
6.6.2	Code « court-circuit »	364
6.6.3	Instructions de flot de contrôle	364
6.6.4	Traduction des expressions booléennes en contrôle de flot	367
6.6.5	Prévention des branchements redondants	369
6.6.6	Valeurs booléennes et code de branchement	370
	Exercices	371
6.7	Reprise arrière	372
6.7.1	Génération de code en une passe avec reprise arrière	373
6.7.2	Reprise arrière pour les expressions booléennes	373
6.7.3	Instructions de flot de contrôle	376
6.7.4	Instructions rompre, continuer et aller à	378
	Exercices	379
6.8	Instructions d'aiguillage	380
6.8.1	Traduction des instructions d'aiguillage	380
6.8.2	Traduction dirigée par la syntaxe des instructions d'aiguillage	381
	Exercices	382
6.9	Code intermédiaire pour les procédures	383
	Résumé	384
	Bibliographie	385
7	Environnements d'exécution	387
7.1	Organisation de l'espace mémoire	387
7.1.1	Allocations de mémoire statiques et dynamiques	390
7.2	Allocation en pile	390
7.2.1	Arbre d'activation	390
7.2.2	Blocs d'activation	393
7.2.3	Séquences d'appel	395
7.2.4	Données de taille variable dans la pile	398
	Exercices	399
7.3	Accès en pile à des données non locales	400
7.3.1	Accès aux données sans procédures imbriquées	400
7.3.2	Problèmes dus à l'imbrication de procédures	401
7.3.3	Un langage avec imbrication des déclarations de procédures	401
7.3.4	Profondeur d'imbrication	403
7.3.5	Liens d'accès	404
7.3.6	Gestion des liens d'accès	404
7.3.7	Liens d'accès pour les procédures en paramètres	406
7.3.8	Adresseurs	407
	Exercices	409
7.4	Gestion du tas	410
7.4.1	Gestionnaire de mémoire	410
7.4.2	Hiérarchie de la mémoire d'un ordinateur	411

7.4.3	Localité des programmes	413
7.4.4	Réduction de la fragmentation	415
7.4.5	Demandes explicites de libération	417
	Exercices	420
7.5	Introduction au ramassage de miettes (collecte du rebut)	420
7.5.1	Objectifs des ramasse-miettes	420
7.5.2	Accessibilité	422
7.5.3	Ramasse-miettes à comptage de références	424
	Exercices	425
7.6	Introduction au ramassage de miettes à marquage	426
7.6.1	Un ramasse-miettes basique à marquage et balayage	427
7.6.2	Abstraction de base	428
7.6.3	Optimisation du marquage et balayage	430
7.6.4	Ramassage de miettes par marquage et tassage	431
7.6.5	Ramasse-miettes copiants	434
7.6.6	Comparaison des coûts	436
	Exercices	436
7.7	Ramasse-miettes à interruption brève	437
7.7.1	Ramassage de miettes incrémental	437
7.7.2	Analyse incrémentale de l'accessibilité	438
7.7.3	Principes de la collecte partielle	440
7.7.4	Ramassage de miettes à générations	441
7.7.5	L'algorithme du train	442
	Exercices	446
7.8	Sujets avancés en ramassage de miettes	447
7.8.1	Ramassage de miettes parallèle et concurrent	447
7.8.2	Relogement partiel	449
7.8.3	Collecte prudente pour langage faiblement typé	450
7.8.4	Références faibles	450
	Exercices	451
	Résumé	451
	Bibliographie	453
8	Production de code	455
8.1	Sujets importants pour la conception d'un générateur de code	457
8.1.1	Données d'entrée du générateur de code	457
8.1.2	Le programme cible	457
8.1.3	Sélection des instructions	458
8.1.4	Allocation de registres	460
8.1.5	Ordre d'évaluation	461
8.2	Langage cible	461
8.2.1	Un modèle simple de machine cible	462
8.2.2	Coût d'un programme et coût des instructions	464
	Exercices	465
8.3	Adresses dans le code cible	467
8.3.1	Allocation statique	467
8.3.2	Allocation en pile	469
8.3.3	Adresses des variables à l'exécution	472

	Exercices	472
8.4	Blocs de base et graphes de flot de contrôle	473
	8.4.1 Blocs de base	474
	8.4.2 Information sur l'utilisation ultérieure	476
	8.4.3 Graphes de flot de contrôle	476
	8.4.4 Représentation des graphes de flot de contrôle	478
	8.4.5 Boucles	478
	Exercices	479
8.5	Optimisation des blocs de base	479
	8.5.1 Représentation des blocs de base par des DAG	480
	8.5.2 Recherche des sous-expressions communes locales	480
	8.5.3 Élimination du code inutile	482
	8.5.4 Utilisation d'identités algébriques	482
	8.5.5 Représentation des accès aux tableaux	484
	8.5.6 Affectations de pointeurs et appels de procédures	485
	8.5.7 Reconstruction des blocs de base à partir des DAG	486
	Exercices	487
8.6	Un générateur de code simple	488
	8.6.1 Descripteurs de registres et d'adresses	489
	8.6.2 L'algorithme de production de code	490
	8.6.3 Conception de la fonction <i>choiReg</i>	493
	Exercices	494
8.7	Optimisation à lucarne	495
	8.7.1 Élimination des chargements et des rangements superflus	495
	8.7.2 Élimination du code inaccessible	495
	8.7.3 Optimisations du flot de contrôle	496
	8.7.4 Simplification algébrique et réduction de force	497
	8.7.5 Utilisation des idiomes de la machine	497
	Exercices	497
8.8	Allocation et affectation de registres	498
	8.8.1 Allocation de registres globale	498
	8.8.2 Décomptes d'utilisation	499
	8.8.3 Affectation de registres pour les boucles externes	500
	8.8.4 Allocation de registres par coloriage de graphe	500
	Exercices	502
8.9	Sélection des instructions par réécriture d'arbre	502
	8.9.1 Schémas de traduction d'arbres	502
	8.9.2 Génération de code par pavage d'arbre	504
	8.9.3 Filtrage par analyse syntaxique	507
	8.9.4 Routines de contrôle sémantique	509
	8.9.5 Filtrage d'arbre général	509
	Exercices	510
8.10	Production d'un code optimal pour les expressions	511
	8.10.1 Nombres d'Ershov	511
	8.10.2 Production du code à partir des arbres d'expressions étiquetés	512
	8.10.3 Évaluation d'expressions avec trop peu de registres	513
	Exercices	515
8.11	Production de code par programmation dynamique	516

8.11.1	Évaluation contiguë	516
8.11.2	L'algorithme par programmation dynamique	517
	Exercices	519
	Résumé	520
	Bibliographie	521
9	Optimisations indépendantes de la machine	523
9.1	Les sources principales d'optimisation	524
9.1.1	Causes de redondance	524
9.1.2	Un exemple à suivre : le tri rapide <i>Quicksort</i>	525
9.1.3	Transformations préservant la sémantique	527
9.1.4	Sous-expressions communes globales	527
9.1.5	Propagation de copie	530
9.1.6	Élimination de code inutile	531
9.1.7	Déplacement de code	532
9.1.8	Variables d'induction et réduction de force	532
	Exercices	535
9.2	Introduction à l'analyse de flot de données	536
9.2.1	Modélisation du flot de données	536
9.2.2	Le schéma d'analyse de flot de données	538
9.2.3	Schémas de flot de données sur les blocs de base	539
9.2.4	Définitions visibles	540
9.2.5	Analyse des variables actives	547
9.2.6	Expressions disponibles	549
9.2.7	Résumé	552
	Exercices	553
9.3	Fondements de l'analyse de flot de données	555
9.3.1	Demi-treillis	556
9.3.2	Fonctions de transfert	560
9.3.3	L'algorithme itératif pour les canevas généraux	562
9.3.4	Signification d'une solution de flot de données	564
	Exercices	566
9.4	Propagation de constantes	567
9.4.1	Valeurs de flot de données pour la propagation de constante	567
9.4.2	La jointure pour le canevas de propagation de constante	568
9.4.3	Fonctions de transfert pour le canevas de propagation de constante	568
9.4.4	Monotonie du canevas de propagation de constante	569
9.4.5	Non distributivité du canevas de propagation de constante	569
9.4.6	Interprétation des résultats	571
	Exercices	572
9.5	Élimination de redondance partielle	573
9.5.1	Les sources de redondance	573
9.5.2	Peut-on éliminer toutes les redondances ?	576
9.5.3	Problème du déplacement de code paresseux	577
9.5.4	Anticipation des expressions	578
9.5.5	L'algorithme de déplacement de code paresseux	579
	Exercices	587
9.6	Boucles dans les graphes de flot	588

9.6.1	Dominateurs	588
9.6.2	Ordonnancement en profondeur	591
9.6.3	Arcs dans un arbre de recouvrement en profondeur	593
9.6.4	Arcs de retour et réductibilité	595
9.6.5	Profondeur d'un graphe de flot	596
9.6.6	Boucles naturelles	596
9.6.7	Vitesse de convergence des algorithmes itératifs de flot de données	598
	Exercices	600
9.7	Analyse par régions	602
9.7.1	Régions	602
9.7.2	Hierarchies de régions pour les graphes de flot réductibles	603
9.7.3	Description rapide d'une analyse par régions	606
9.7.4	Hypothèses nécessaires sur les fonctions de transfert	607
9.7.5	Un algorithme pour l'analyse par régions	609
9.7.6	Traitement des graphes de flot non réductibles	613
	Exercices	614
9.8	Analyse symbolique	615
9.8.1	Expressions affines des variables de référence	616
9.8.2	Formulation d'un problème de flot de données	619
9.8.3	Analyse symbolique par régions	622
	Exercices	627
	Résumé	627
	Bibliographie	630
10	Parallélisme entre instructions	633
10.1	Architectures des processeurs	635
10.1.1	Pipelines d'instructions, branchements retardés	635
10.1.2	Exécution pipelinée	636
10.1.3	Lancement multiple	636
10.2	Contraintes de l'ordonnancement du code	637
10.2.1	Dépendances de données	637
10.2.2	Recherche des dépendances entre accès à la mémoire	638
10.2.3	Compromis entre utilisation des registres et parallélisme	640
10.2.4	Ordre des phases d'allocation de registres et d'ordonnancement de code	642
10.2.5	Dépendances de contrôle	642
10.2.6	Support matériel pour une exécution spéculative	643
10.2.7	Un modèle simple de machine	645
	Exercices	646
10.3	Ordonnancement d'un bloc de base	647
10.3.1	Graphes de dépendances de données	647
10.3.2	Ordonnancement de liste des blocs de base	649
10.3.3	Tris topologiques avec priorité	650
	Exercices	651
10.4	Ordonnancement global	652
10.4.1	Principes du déplacement de code	653
10.4.2	Déplacement de code vers l'amont	655
10.4.3	Déplacement de code vers l'aval	655

10.4.4	Mise à jour des dépendances de données	657
10.4.5	Algorithmes d'ordonnancement global	657
10.4.6	Techniques avancées de déplacement de code	660
10.4.7	Interaction avec un ordonnanceur dynamique	661
	Exercices	662
10.5	Le pipeline logiciel	662
10.5.1	Introduction	662
10.5.2	Pipeline logiciel des boucles	664
10.5.3	Allocation de registres et génération de code	666
10.5.4	Boucles avec dépendances entre les itérations	668
10.5.5	Buts et contraintes du pipeline logiciel	669
10.5.6	Un algorithme de pipeline logiciel	672
10.5.7	Ordonnancement de graphes de dépendances sans circuit	672
10.5.8	Ordonnancement de graphes de dépendances avec circuits	674
10.5.9	Améliorations possibles des algorithmes de pipeline logiciel	680
10.5.10	Expansion périodique de variable	681
10.5.11	Instructions conditionnelles	683
10.5.12	Support matériel pour le pipeline logiciel	684
	Exercices	684
	Résumé	686
	Bibliographie	688
11	Parallélisme et localité des données	689
11.1	Concepts de base	692
11.1.1	Multiprocesseurs	692
11.1.2	Parallélisme dans les applications	694
11.1.3	Parallélisme de boucle	695
11.1.4	Localité des données	697
11.1.5	Introduction à la théorie des transformations affines	699
11.2	Multiplication de matrices	702
11.2.1	L'algorithme de multiplication de matrices	702
11.2.2	Optimisations	704
11.2.3	Interférence de cache	707
	Exercice	707
11.3	Espace des itérations	707
11.3.1	Construction de l'espace d'itérations d'un nid de boucles	708
11.3.2	Ordre d'exécution des nids de boucles	710
11.3.3	Formulation matricielle des inégalités	711
11.3.4	Utilisation de constantes symboliques	712
11.3.5	Contrôler l'ordre d'exécution	712
11.3.6	Changement d'axes	716
	Exercices	717
11.4	Accès affines aux tableaux	719
11.4.1	Accès affines	719
11.4.2	Fonctions d'accès affines et non affines	720
	Exercice	721
11.5	Réutilisation de données	721
11.5.1	Types de réutilisation	723

11.5.2	Réutilisation interne	723
11.5.3	Réutilisation spatiale interne	727
11.5.4	Réutilisation externe	728
	Exercices	730
11.6	Analyse de dépendances pour les tableaux	731
11.6.1	Définition de la dépendance de données entre accès à un tableau	732
11.6.2	Programmation linéaire en nombres entiers	733
11.6.3	Le test du PGCD	734
11.6.4	Heuristiques de résolution des systèmes linéaires dans les entiers	736
11.6.5	Résolution des systèmes linéaires en nombres entiers : cas général	739
11.6.6	Résumé	740
	Exercices	741
11.7	Recherche d'exécution parallèle sans synchronisation	742
11.7.1	Un exemple en introduction	743
11.7.2	Partition spatiale affine	745
11.7.3	Contraintes de partitionnement spatial	746
11.7.4	Résolution des contraintes de partitionnement spatial	749
11.7.5	Un algorithme simple de génération de code	752
11.7.6	Élimination des itérations vides	754
11.7.7	Éliminer les tests des boucles internes	757
11.7.8	Transformations du code source	758
	Exercices	763
11.8	Synchronisation entre boucles parallèles	765
11.8.1	Un nombre constant de synchronisations	765
11.8.2	Graphe de dépendances d'un programme	766
11.8.3	Temps hiérarchisé	768
11.8.4	Algorithme de parallélisation	770
	Exercices	771
11.9	Exécution pipelinée	772
11.9.1	Qu'est-ce qu'une exécution pipelinée ?	772
11.9.2	La sur-relaxation (SOR) : un exemple	774
11.9.3	Boucles complètement permutable	774
11.9.4	Pipeliner des boucles complètement permutable	775
11.9.5	Théorie générale	777
11.9.6	Contraintes de partitionnement temporel	778
11.9.7	Résolution des contraintes de partitionnement temporel à l'aide du lemme de Farkas	781
11.9.8	Transformation du code	784
11.9.9	Parallélisation avec un minimum de synchronisations	789
	Exercices	791
11.10	Optimiser la localité des données	793
11.10.1	Localité temporelle des données calculées	793
11.10.2	Contraction de tableau	793
11.10.3	Entrelacement de parties indépendantes	796
11.10.4	Utilisation de toutes les transformations	798
	Exercices	799
11.11	Transformations affines : autres applications	800
11.11.1	Machines à mémoire distribuée	801

11.11.2	Processeurs à lancement multiple	802
11.11.3	Instructions vectorielles et SIMD	802
11.11.4	Préchargement	803
	Résumé	804
	Bibliographie	806
12	Analyses interprocédurales	809
12.1	Concepts de base	810
12.1.1	Graphe des appels	810
12.1.2	Sensibilité au contexte	812
12.1.3	Chaînes d'appels	814
12.1.4	Analyse sensible au contexte par clonage	816
12.1.5	Analyse sensible au contexte par résumés	817
	Exercices	819
12.2	Pourquoi une analyse interprocédurale ?	821
12.2.1	Invocation de méthodes virtuelles	821
12.2.2	Analyse de synonymie des pointeurs	821
12.2.3	Parallélisation	822
12.2.4	Détection des erreurs logicielles et des trous de sécurité	822
12.2.5	Injection de commandes SQL	823
12.2.6	Débordement de tampon	824
12.3	Représentation du flot de données par programmation logique	825
12.3.1	Introduction à Datalog	826
12.3.2	Règles Datalog	827
12.3.3	Prédicats intensionnels et extensionnels	828
12.3.4	Exécution des programmes Datalog	831
12.3.5	Évaluation incrémentale des programmes Datalog	832
12.3.6	Les règles Datalog qui posent problème	833
	Exercices	835
12.4	Un algorithme simple d'analyse des pointeurs	836
12.4.1	Pourquoi l'analyse des pointeurs est difficile	836
12.4.2	Un modèle pour les pointeurs et les références	838
12.4.3	Insensibilité au flot	838
12.4.4	Formulation en Datalog	839
12.4.5	Utilisation de l'information de type	840
	Exercices	842
12.5	Analyse interprocédurale insensible au contexte	843
12.5.1	Effets d'un appel de méthode	843
12.5.2	Calcul du graphe des appels en Datalog	845
12.5.3	Chargement dynamique : introspection dans les classes Java	846
	Exercices	846
12.6	Analyse de pointeurs sensible au contexte	847
12.6.1	Contextes et chaînes d'appels	848
12.6.2	Ajout du contexte aux règles Datalog	850
12.6.3	Observations complémentaires	850
	Exercices	851
12.7	Implémentation de Datalog par les DDB	852
12.7.1	Diagrammes de décisions binaires (DDB)	852

12.7.2	Transformations sur les DDB	853
12.7.3	Représentation de relations par des DDB	854
12.7.4	Opérations sur les relations et les DDB	855
12.7.5	Utilisation des DDB pour l'analyse de pointeurs	857
	Exercices	858
	Résumé	858
	Bibliographie	860
A	Une partie frontale complète	863
A.1	Le langage source	863
A.2	Programme principal	864
A.3	Analyseur lexical	865
A.4	Tables de symboles et types	868
A.5	Code intermédiaire pour les expressions	869
A.6	Code de branchement	872
A.7	Code intermédiaire pour les instructions	875
A.8	Analyseur syntaxique	879
A.9	Création de la partie frontale	883
B	Recherche de solutions linéairement indépendantes	885