De quelques opérateurs négligés

Louis Frécon

Résumé

Les algorithmes ont pour support des algèbres plus ou moins classiques, d'où la nécessité de ne pas trop limiter les opérateurs standards.

Cet article rappelle d'abord la nécessité de banaliser à tous les niveaux les opérateurs max et min. Au plan numérique, l'intérêt de ces opérateurs est sensible dans certains domaines applicatifs, puis pour linéariser la programmation en général, enfin pour accélérer le mode pipe-line des processeurs. Et ces remarques s'étendent rapidement hors du domaine numérique.

Il revisite également la « mise en parallèle » et la composition quadratique comme autres candidats sérieux au statut d'opérateurs standards.

Abstract

Algorithms are founded on algebraic structures. So, they may be poor and lengthy if standard operators forms a too short list. This paper is devoted to the reevaluation of max/min, parallel and quadratic compositions as serious candidates as generic and standard operators.

Sommaire

Introduction

Dans les années 70, Jean Kuntzmann professait à l'ENSIMAG que « L'informatique, c'est les mathématiques en action ».

Plus précisément, on peut postuler que, pour toute application, l'algorithme est d'autant plus efficace qu'il s'appuie sur l'algèbre adéquate.

Cette idée était à l'origine des langages extensibles à opérateurs, comme Algol 68 ou, à un moindre degré, Prolog, Ada ou C++.

C'est ainsi que deux étudiants de l'INSA Lyon établirent expérimentalement que paradoxalement, les applications Cobol de plus de 2.000 lignes gagnait à être refaites en Algol 68, au sens de 5 critères distincts : nombre de lignes, encombrement mémoire, vitesse d'exécution, durée de développement, durée de mise au point, Algol 68 permettant d'exploiter au plus près toute modélisation pertinente.

Cette approche pose un problème : les types construits doivent posséder non seulement un domaine et une représentation, mais doivent aussi être munis d'opérateurs judicieux, à la fois expressifs et riches en propriétés, afin d'éviter d'obscurcir les algorithmes par le bricolage des représentations, et d'optimiser le code.

On a par exemple longtemps subi en programmation l'effet désastreux d'une trop lente émergence du type chaîne, qui a du trop souvent se contenter d'exister sans disposer d'opérateurs adéquats.

Comme il n'est pas question pour les développeurs de passer leur vie à réinventer l'eau chaude, la généricité des opérateurs est un question importante pour éviter la duplication des efforts : c'est un vecteur du report d'expérience.

Le cas max/min

Définitions et premières propriétés

On pose en général :

$$max(x, y) \equiv si\ x< y\ alors\ y\ sinon\ x$$
$$min(x, y) \equiv si\ x<y\ alors\ x\ sinon\ y$$

De ce fait,

$$max(x, y) \equiv -min(-x, -y) \quad (dualité)$$
$$max(x, y) + min(x, y) \equiv x + y.$$

Ces opérateurs sont riches en propriétés. En effet, ils sont associatifs et commutatifs, ce qui justifie des transcriptions allégées comme

$$max(x, max(max(y, z), t)) \equiv max(x, y, z, t)$$

Ils sont idempotents, mutuellement distributifs, et dotés d'une propriété d'absorption. Ils sont aussi distributifs par rapport à l'addition, et distributifs à droite par rapport à la soustraction.

On peut enfin leur associer deux opérateurs unaires

$$abs(x) \equiv max(x, -x)$$
$$pos(x) \equiv max(x, 0).$$

ce qui permet d'établir des propriétés comme :

$$abs(x) + x = \max(x, -x)+x = \max(2x, 0) = pos(2x).$$

Champs applicatifs

Chemins optimaux

Les problèmes de chemins optimaux ayant donné naissances à des centaines d'algorithmes similaires, le domaine a été unifié par Kuntzmann puis Gondran par la théorie des dioïdes, structures algébriques à deux opérateurs : un *opérateur adjonctif* pour synthétiser deux chemins, un *opérateur séquentiel* pour allonger un chemin.

Recherche de chemin	Opérateur adjonctif	Opérateur séquentiel
de *longueur* minimale de *coût* minimal de *taille* minimale	min	+
de longueur maximale ou *chemin critique*	max	+
de *capacité* maximale	max	min

Interface Logique/Numérique

L'interface numérique/logique a été initialement assurée par le système Boole/Vallée

$$et(x,y) \equiv x * y$$

$$ou(x, y) \equiv x + y - x*y.$$

A partir de Lukasiewicz, on l'a remplacé avantageusement en posant dans $\mathbb{B} = \{0, 1\}$:

$$ou(x, y) \equiv max(x, y)$$

$$et(x, y) \equiv min(x, y)$$

relations qui s'étendent aux logiques multivaluées à la Lukasiewicz et aux logiques floues du type Zadeh.

Fonctions linéaires par morceaux

Elles sont souvent employées comme approximation de fonctions empiriques continues, par exemple pour cadrer une valeur ou pour représenter des distributions trapézoïdales, des fonctions en S, des courbes en baignoire…. Elles peuvent être définies à l'aide d'expressions conditionnelles ou, si elles sont continues, à l'aide de min et max, max correspond alors aux concavités et min aux points saillants.

Calcul symbolique

Les systèmes de calcul symbolique peuvent

- utiliser *max, min, abs, pos* pour le traitement de expressions conditionnelles ;
- tirent partie du fait que *max, min, abs, pos*, se ramènent finalement à l'usage de max (ou min) et du moins unaire ;
- exploiter les nombreuses propriétés de ces opérateurs.

Optimisation des programmes

Au sens des performances, l'optimisation des programmes repose largement sur l'optimisation des expressions, donc en fait sur des calculs symboliques effectués sur le programme vu comme une expression.

En ce sens, l'intégration d'un volet max/min peut se révéler très efficace : certains emboîtements d'instructions conditionnelles peuvent se ramener à des expressions conditionnelles, elles-mêmes réductibles à des expressions en min/max.

Ainsi, soit à cadrer une valeur x entre 0 et 200. Une fonction de cadrage comme :

cadrage (x, inf sup) = [local y ;
si x<inf alors y :=inf sinon
si x <sup alors y :=x sinon y := sup ;
retour y]

peut être remplacée par :

cadrage (x, inf sup) = [local y ;
y :=(si x<inf alors inf sinonsi x <sup alors x sinon sup) ;
retour y]

puis par :

cadrage (x, inf sup) = [local y ;
y := max(inf, min (x, sup)) ;
retour y]

et, enfin, en remarquant l'affectation unique et l'usage unique de y :

cadrage (x, inf, sup) = [retour max(inf, min (x, sup))]

qui évite le recours à l'allocation de mémoire, et peut même se ramener à une macro-instruction :

cadrage (x, inf, sup) = max(inf, min (x, sup))

facilitant de nouvelles réductions.

Linéarisation de l'exécution

Les transformations précédentes prennent tout leur sens quand on observe que min, max, abs, pos, peuvent être des opérateurs natifs pour les processeurs :

- *pos* transmet l'opérande si son signe est 0, et 0 sinon,
- *abs* transmet l'opérande si son signe est 0, et son complément sinon,
- *max/min* transmet l'un ou l'autre des opérandes selon le résultat d'une comparaison bit à bit des deux opérandes présumés de même type.

En effet,

- les entiers non signés se comparent simplement bit à bit, en partant du bit de plus fort poids : la comparaison est finie s'ils diffèrent, sinon la comparaison se ramène à celle des bits de poids inférieur ;
- les entiers signés se comparent de façon similaire, en comparant d'abord les signes : la comparaison est finie si les signes diffèrent, sinon, on se ramène à la comparaison d'entiers sans signe, directe si les deux entiers sont positifs, à inverser s'ils sont négatifs ;
- les nombres virgule flottante type IEEE se comparent de même, leur représentation prévoyant que deux nombres virgule flottante sont entre eux comme les entiers signés de même représentation binaire ; en particulier, la comparaison de nombre virgule flottante à précision multiple s'arrête le plus souvent à la comparaison des mots de tête, porteurs du signe, de l'exposant, et des bits les plus significatifs de la mantisse ; de ce fait, cette comparaison directe est généralement beaucoup plus rapide qu'une addition/soustraction.

Les transformations examinées antérieurement, en refoulant toujours plus bas des branchements explicites ou implicites, peuvent finalement les évacuer du flot d'instructions, lissant au mieux le fonctionnement pipe-line du processeur, donc son efficacité globale.

Généricité

Ensembles numériques

Dans les paragraphes précédents nous n'avons pas attaché d'importance excessive au domaine numériques : les raisonnements et propriétés sont sensiblement les mêmes que les opérandes soient entiers, rationnels ou réels. De fait la seule question est importante est celle de savoir si les nombres sont signés, ce qui change la borne inférieure du domaine, donc les éléments neutres et/ou absorbants.

Domaine	max		min	
	elt neutre	elt absorbant	elt neutre	elt absorbant
$\mathbb{N}$, $\mathbb{Q}^+$, $\mathbb{R}^+$	0	$+\infty$	$+\infty$	0
$\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$	$-\infty$	$+\infty$	$+\infty$	$-\infty$

Ensembles finis totalement ordonnés

Avec juste raison, N.Wirth introduisit en Pascal les *ensembles finis totalement ordonnés*, avec d'une part les ensembles énumérés, et, comme sous-types, les intervalles de ces ensembles comme les intervalles d'entiers ou de caractères.

Cependant, les opérations autorisées se limitaient

- aux opérations unaires *pred* et *succ,*
- aux comparaisons.

En effet, l'objectif semblait essentiellement de justifier :

- une boucle *for* généralisée (à pas unitaire),
- un assouplissement de l'indexation des tableaux,
- une instruction *case* permettant de clarifier certaines cascades de test, et de les accélérer en les ramenant aux niveaux inférieurs à l'exploitation d'un table de branchement.

Ces dispositions ont plus ou moins renouvelé la programmation selon les implémentations, certaines n'autorisant que des ensembles de 16 éléments au plus, laissant ces nouveautés lettre morte, d'autres en autorisant jusqu'à 4096, ce qui permettait de nouvelles analyses.

Cette (r)évolution était peut-être trop timide.

A l'évidence, les types wirthiens totalement ordonnés peuvent être étendus à l'aide de *max* et *min* opérant entre termes d'un même ensemble E, en gardant leurs propriétés intrinsèques : associatifs, commutatifs, idempotents, mutuellement distributifs.

Opérateur de E	Elément neutre	Elément absorbant
max	inf(E)	sup(E)
min	sup(E)	inf(E)

On peut de même admettre l'usage de *max* et *min* entre termes relevant de types différents E_1 et E_2 partageant un référentiel commun E : le résultat est alors dans $\hat{E}_{12}$, le plus petit intervalle de E contenant E_1 et E_2 ; et ce résultat est précalculable si E1 et E2 sont disjoints.

L'apparition de structures indexées par des clés, comme les *dictionnaires*, montre l'intérêt porté maintenant à l'indexation de structures de données par des chaînes ou symboles. On sait la différence entre l'introduction *a minima* d'un type chaîne, et l'introduction d'un type chaîne, de plein statut, avec des opérateurs fondant une algèbre des chaînes. Cette algèbre basée sur la concaténation, la réplication… pourrait aussi s'enrichir de max/min, avec cette fois une différence importante pour l'ordre total sous-jacent, ordre ne portant plus sur un domaine isomorphe à un intervalle de $\mathbb{N}$, mais ordre lexicographique sur un domaine maintenant isomorphe à [0 1].

Treillis

Dans les ensembles partiellement ordonnés, deux valeurs a et b ne sont pas toujours comparables, mais s'il s'agit de treillis, elles possèdent toujours un plus petit majorant, noté max(a, b), et un plus grand minorant, noté min(a, b).

Quand rencontre-t-on des treillis ?

De fait, si X est un ensemble totalement ordonné, son ordre induit sur X^n une structure de treillis si on pose que p surclasse q dans X^n si chacune des composantes de p surclasse la composante correspondante de q dans X.

Treillis de divisibilité

La relation « divise », notée « | », introduit un ordre partiel sur l'ensemble des entiers, qui se trouve ainsi doté d'une structure de treillis, ayant 1 comme élément minimal, et dans lequel chaque nombre premier a le statut d'atome.

Attachons à tout entier n sa décomposition en facteurs premiers, et notons expo(n) la suite de leurs exposants, avec par exemple :

$$2100 = 2^2 \cdot 3 \cdot 5^2 \cdot 7 \rightarrow \text{expo}(n) = (2, 1, 2, 1)$$

expo(n) peut être considéré comme donnant les coordonnées de n dans un treillis de divisibilité ayant une dimension par nombre premier.

On a alors les relations :

$$expo(m \cdot n) = expo(m) + expo(n)$$
$$expo(pgcd(m, n)) = min(expo(m), expo(n))$$
$$expo(ppcm(m, n) = max(expo(m), expo(n))$$

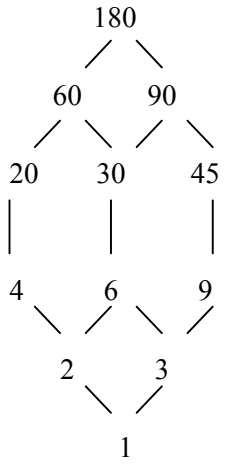
Ces propriétés peuvent être notamment mises au service de la représentation de tout treillis fini par un codage multiplicatif conservant l'ordre partiel (voir exemple ci-après).

Treillis logiques

Goguen a établi que la plupart des concepts des logiques floues pouvaient s'étendre à tout système logique fondé sur un ensemble de vérité ayant une structure de treillis.

On connaît par ailleurs diverses logiques multi-valuées ayant un ensemble de vérité isomorphe à $\mathbb{B}^n$.

Exemple de codage multiplicatif conservant une structure de treillis



La composition parallèle d'impédances

Présentation

Le problème général de la valeur d'un faisceau de chemins élémentaires reliant un point A et à un point B, admet un cas déviant important pour ses lois de composition : celui des *réseaux d'impédances.*

L'électrotechnique et l'électronique ont créé et vulgarisé les notions de résistance (définie à propos des circuits en courant continu) puis d'impédance (généralisation au cas des régimes alternatifs permanents).

Ces impédances sont souvent combinées dans des circuits, des filtres etc….

Yves Rocard a montré que ces concepts sont adaptables aux domaines admettant des modèles linéaires similaires, comme celui des vibrations en mécanique (domaine élastique), ou l'étude de l'acoustique ; la théorie des quadripôles fournit alors une approche unificatrice.

Ces concepts ont également été adaptés à la thermique. Ils peuvent l'être à l'hydraulique…

C'est dire leur importance en informatique technique.

Propriétés

Les combinaisons « en série » sont additives, et notées comme telles.

Une notation du genre R1//R2 pour les combinaisons parallèles n'est-elle qu'une sténographie regrettable, comme on l'a longtemps affirmé, ou a-t-elle un sens algébrique ?

Posons

$$R1 \,//\, R2 \equiv inv(inv(R1)+inv(R2)).$$

On vérifie facilement que « // » est associatif et commutatif, ce qui justifie des notations allégées comme R1//R2//R3//R4. « // » est d'élément neutre infini (liaison ouverte), et d'élément absorbant 0 (court-circuit).

De plus,

$$R \,//\, (-R) \equiv \infty.$$

Toute résistance ayant ainsi pour « // » une symétrique –R, « // » engendre une structure de groupe, et la possibilité de résoudre les équations correspondantes, avec :

$$A \,//\, R = B \rightarrow R = (-A)//B.$$

L'exigence usuelle de résistance positive est ici respectée si A > B.

$$47 \,//\, R = 33 \rightarrow R = (-47)//33 = 110$$

qui se lira par exemple : pour obtenir une résistance de 33Ω à partir d'une résistance de 47 Ω, il suffit de monter en parallèle sur celle-ci une résistance de 110 Ω.

Mais évidemment

$$33 // R = 47 \rightarrow R = (-33) // 47 = -110.$$

n'aura de sens que si la résistance inconnue peut être un dipôle amplificateur (ce qui n'est possible qu'avec certaines technologies).

Plus généralement, soit le système :

$$R//R1 = 20 ; R//R1//R2 = 15 ; R//R2 = 30$$

Nous en tirons successivement :

R2	= (-(R//R1))//15	= (-20)//15	= 60
R	= 30//(-R2)	= 30//(-60)	= 60
R1	= 20//(-R)	= 20//(-60)	= 30.

Le calcul est simple bien que surprenant ; ce qui conforte la thèse de Robert Laurini[1], que l'utilisateur devrait pouvoir se contenter d'écrire ses équations, à charge pour le système de programmation de les résoudre en fonction des invocations et des contraintes, ce que confirment les systèmes simples comme :

$$R1 // R2 = A,\ R1 + R2 = B$$

qui se ramène à un système du second degré, la positivité de R1 et R2 supposant $0 \leq 4A \leq B$.

Le système (//, +) reste cependant spécifique, car // n'est pas idempotente. De plus,en général

$$A//(B+C) \neq (A//B)+(A//C).$$

Robustesse

La formule donnée en définition de R1//R2 est la moins sensible à d'éventuelles fluctuations des opérandes. Utiliser une expression équivalente comme R1·R2/(R1+R2) est exact tant que R1 et R2 sont des valeurs exactes, ce qui n'est vrai que dans les exercices académiques.

En pratique, les valeurs R1 et R2 sont plutôt cadrées que définies (a) par une valeur nominale (b) par une précision certifiée. Les opérandes sont alors à valeur dans un intervalle connu plutôt que connus exactement.

En matière de calcul en intervalles, le théorème de Nickel précise que l'évaluation directe d'une expression n'est exacte que si chaque variable a une seule occurrence, et est indépendante des autres. En ce sens, l'évaluation de R1·R2/(R1+R2) est une évaluation laxiste (ou pessimiste) de R1//R2, pour qui la formule initiale demeure la référence.

Par exemple, soit R1 = 20 ohms (à 5%) et R2= 47 ohms (à 10%), qu'on pourra noter R1 = [19 21], R2=[42 52],

$$R1//R2 = inv([0{,}047\ 0{,}052]+[0{,}019\ 0{,}023]) = inv([0{,}067\ 0{,}076]) = [13\ 16]$$

mais R1·R2/(R1+R2)=[11 18]. *Proposer un opérateur dédié combat une telle imprécision qui peut, par propagation, se révéler lourde de conséquences.*

La composition quadratique []

Depuis Pythagore, on s'intéresse à l'opération

$$A\ []B \equiv rac2(A^2 + B^2)$$

qui lierait l'hypoténuse aux deux autres côtés du triangle rectangle.

[1] communication privée

En tant que telles, cette opération est associative, commutative, d'élément neutre 0 et d'élément absorbant infini.

On peut lui attacher une parité quelconque.

Utilisations

On retrouve la composition quadratique dans tous les calculs de distance euclidienne.

On la retrouve aussi dans la notion de « valeur efficace » : du point de vue énergétique, la valeur efficace d'une tension périodique est la composition quadratique des tensions sinusoïdales qui la composent.

On a toujours $a[]b \leq a + b$: on peut donc se servir de [] pour agréger des valeurs allant « plutôt dans le même sens », lorsque l'addition pure et simple semble exagérée, ou lorsqu'on craint un poids excessif des contributions marginales. En effet, quand b est petit devant a, $a [] b \sim a (1 + b^2/2a^2)$, et si b vaut 14% de a, $a[]b = 1{,}01 \cdot a$.

En docimologie, on peut l'appliquer en vue d'un classement donnant plutôt plus de poids aux forces qu'aux faiblesses, par exemple en vue d'un projet.

candidat	N1	N2	N3	+	[]	Commentaire
A	14	13	13	40	23,11	Correct
B	15	13	12	40	23,19	Intéressant
C	18	13	9	40	**23,91**	**Fort (mais une impasse)**
D	18	11	11	40	23,75	Pointu

Références

Bergman M., et Kanoui H., *SYCOPHANTE, système de calcul formel sur ordinateur*, rapport de fin de contrat DRET, Groupe Intelligence Artificielle, Faculté des Sciences de Luminy, Université Aix-Marseille II, 1975.

Frécon L., *Eléments de Mathématiques Discrètes,* PPUR, 2002

Frécon L., *Quelques principes du langage Ampère*, Groplan 1978.

Gondran et Minoux, *Graphes et algorithmes*, Eyrolles, 1985.

Kuntzmann J., *Algèbre de Boole,* Dunod, 1965

Kuntzmann J., *Théorie des Réseaux*, Dunod, 1968.

Rocard Y. *Dynamique générale des vibrations,* Masson, 1943-1997.

Wirth N., *Introduction à la programmation systématique*, trad. Lecarme O., Masson 1983,