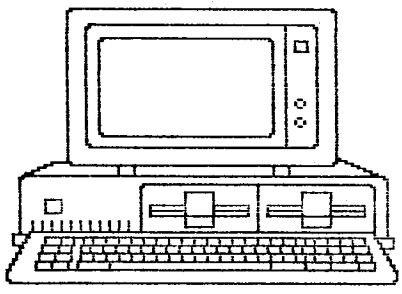
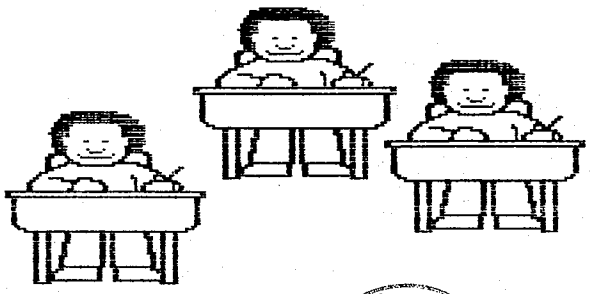
SIMPLIFIER... ce n'est pas tres simple !

De l'apport de l'informatique a ...

la comprehension
des difficultes
des eleves
dans le domaine du
CALCUL ALGEBRIQUE





Etude de quelques systemes de calcul formel

Presentation du projet BRM d'aide a l'apprentissage du calcul algebrique

SIMPLIFIER.....ce n'est pas très simple .....

ou

de l'apport de l'informatique à la compréhension des difficultés des élèves dans le domaine du calcul algébrique.

Dans le cadre du développement* d'un logiciel de calcul formel pour élèves de collège , nous avons été amenés à examiner de près en quoi consistent habituellement les transformations effectuées sur des expressions mathématiques. Nous nous sommes rapidement rendu compte que des mots du genre " simplifier , réduire , calculer , effectuer , développer , factoriser , ..." ne recouvraient pas des activités bien précises , bien descriptibles , bien cataloguées ; qu'elles intervenaient pourtant à tout moment en mathématique , qu'à partir d'un certain niveau elles ne soulevaient plus trop d'ambiguïtés , mais que dans la phase d'apprentissage au collège et au lycée, elles comportaient de nombreuses et sérieuses difficultés pour les élèves.

Les règles fondamentales de calcul qui gèrent l'ensemble des manipulations des expressions algébriques sont bien connues , mais ce sont des règles plus élaborées qui sont le plus souvent utilisées et qui font également l'objet d'un apprentissage. Ces règles plus élaborées sont extrêmement nombreuses , liées aux expressions à manipuler , dépendantes de l'objectif à atteindre. (d'ailleurs souvent mal ou peu précisé)

Dans l'idée de mettre à la disposition des élèves un nombre "restreint" de transformations licites sur les expressions , pouvant d'ailleurs dépendre du niveau des élèves , nous avons essayé de mettre un peu d'ordre dans cette pratique quotidienne du matheux , de regrouper ces transformations par grands thèmes , ... Il nous a fallu également préciser en quoi consiste une expression mathématique et dégager les notions essentielles permettant la compréhension et l'apprentissage corrects des règles de calcul (en particulier : la structure d'une expression mathématique et les conventions d'écriture liées à l'usage des parenthèses et aux règles de priorité sur les opérations).

Ce n'est pas seulement un système cohérent et complet que nous souhaitions, mais un ensemble de transformations le plus proche possible de la pratique habituelle. La démarche utilisée par un logiciel de calcul formel comme MuMath était donc à exclure dès le départ : en effet celui-ci transforme systématiquement soustraction en addition et division en multiplication , ce qui ne correspond pas toujours à la pratique du calcul.

* G. Pozon ; J. Brosse , E MEYER - 1987-88

Le travail mené dans cette direction nous à renforcé , si besoin était , dans la constatation de l'extr@me complexité qui existe dans ce domaine de l'activité mathématique et en particulier dans les difficultés liées à l'usage du signe "moins" (soit pour désigner un opposé , soit pour désigner une différence)

Prenons un exemple avant de préciser la structure des expressions mathématiques considérée et la classification choisie pour les transformations possibles sur elles.

Développer , réduire et ordonner **(2x-1)(3+x)-x**

Par combinaisons de règles élaborées :

$(2x-1)(3+x)-x = 2x^2 + 5x -3 -x = 2x^2 + 4x -3$

En utilisant la double distributivité et en "simplifiant" chacun des termes du développement :

$(2x-1)(3+x)-x = 6x + 2x^2 -3 -x -x = 2x^2 + 4x -3$

Par un développement plus explicite :

$$\begin{aligned}(2x-1)(3+x)-x &= (2x-1)*3 + (2x-1)*x -x\\ &2x.3 -3 + 2x.x -x -x\\ &6x - 3 + 2x^2 - x - x\\ &(6x -x -x) + 2x^2 - 3\\ &4x + 2x^2 - 3\\ &2x^2 + 4x -3\end{aligned}$$

En utilisant les axiomes classiques de la structure de corps (et encore ! on utilisera une forme de l'associativité et de la commutativité déjà élaborée ainsi qu'un usage "ambigu" du signe "moins") :

$$\begin{aligned}(2x-1)(3+x)-x &= (2x-1)*3 + (2x-1)*x -x\\ &(2x+(-1))*3 + (2x+(-1))*x + (-x)\\ &3(2x+(-1)) + x(2x+(-1)) + (-x)\\ &3.2x+3.(-1)+x.2x+ x.(-1) + (-x)\\ &(3.2)x+(-3)+2.x.x+(-x)+(-x)\\ &6x + (-3) + 2.x^2 + (-1).x +(-1).x\\ &2.x^2 + (6x + (-1)x + (-1)x) +(-3)\\ &2.x^2 + (6 + (-1) + (-1)).x + (-3)\\ &2.x^2 + 4.x + (-3)\\ &2.x^2 + 4.x -3\end{aligned}$$

Quelles règles "simples" , "universelles" , "explicitables" peut-on dégager de ces calculs?

STRUCTURE d'une EXPRESSION MATHEMATIQUE

Voici , sur un exemple , ce à quoi nous souhaitons parvenir et qui nous paraît fondamental pour une manipulation correcte des expressions mathématiques :

(2x-1)(3+x)-x : c'est la différence entre (2x-1)(3+x) et x ;
(2x-1)(3+x)...: c'est le produit de 2x-1 par 3+x ;
2x-1 .....: c'est la différence entre 2x et 1 ;
2x .....: c'est le produit de 2 par x ;
3+x .....: c'est la somme de 3 et de x .

Ceci dégage la structure de l'expression et peut être représenté sous forme d'un arbre :

(2x-1)(3+x)-x

```
     !
     !
     - ( différence )
     !
     !_____ (2x-1)(3+x)
     !
     !_____ x
```

(2x-1)(3+x)

```
     !
     !
     * ( produit )
     !
     !____ 2x-1
     !
     !____ 3+x
```

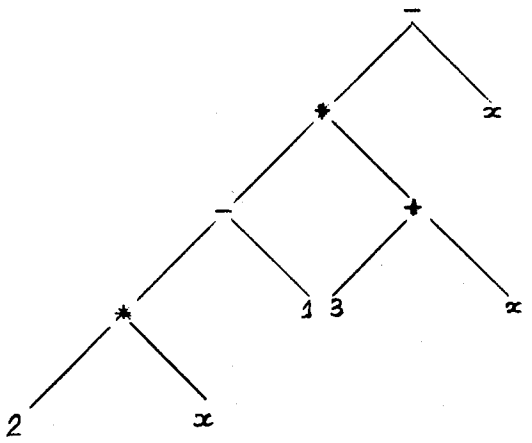
2x-1

```
     !
     !
     -
     !
     !___ 2x
     !
!___ 1
```

etc

Voici le dessin complet de l'arbre :

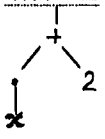
(2x-1)(3+x)-x



Quelques unes des difficultés pour choisir la représentation des expressions mathématiques et leurs structures :

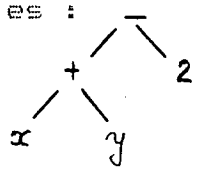
* Bien que (x)+2 soit égal à x+2 , nous souhaitons , comme l'écriture le fait , distinguer ces deux expressions :

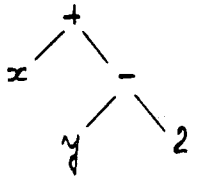
(x)+2



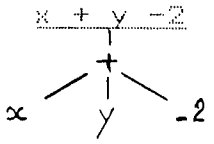
x+2

* Comment interpréter x + y - 2 ?
Voici deux solutions possibles :

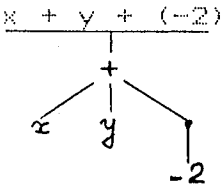




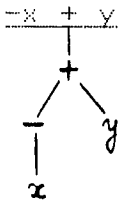
Comme il n'y a pas de raison " dans la pratique habituelle du calcul " de choisir entre ces deux solutions , nous avons finalement opté pour la solution suivante :

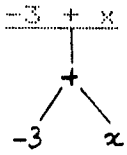


Ceci nous amène à considérer x + y + (-2) de la manière suivante :



* Que faire avec -x+y ? et -3 + x ? Voici les solutions adoptées :





Nous avons donc introduit la notion d'opposé ; par contre nous n'avons pas introduit la notion d'inverse qui n'apparaît pas dans l'écriture habituelle.

Le choix finalement adopté :

Nous présentons ce choix sous forme de "représentation informatique" , équivalente aux expressions " de surface " habituelles, en utilisant la notation préfixée.

Une expression est soit un ATOME , soit une LISTE.

Un ATOME est soit un ENTIER , soit un IDENTIFICATEUR.

(Nous excluons délibérément les nombres décimaux et à fortiori les réels non rationnels pour rester le plus près possible des transformations habituelles)

Les pratiques de calcul nous ont amené à faire des distinctions même parmi les entiers (entiers strictement négatifs ,-1 , 0 , 1 , entiers strictement positifs)

Un IDENTIFICATEUR est soit un identificateur de variable , soit un identificateur d'expression , soit un identificateur de sous-expression. (ces distinctions se justifient par la suite du travail) Un tel IDENTIFICATEUR est une lettre ou un mot commençant par une lettre ...

Une LISTE est l'un des objets suivants :

[EXP] .....: liste à 1 élément du type expression.
exemple : (2.x + 3)

[- EXP] .....: liste à 2 éléments
exemple : - $5.x^2$

[- EXP1 EXP2] ..: liste à 3 éléments pour une différence.
exemple : (2.x + 3) - 5.x

[/ EXP1 EXP2] ..: liste à 3 éléments pour un quotient.
exemple : (2.x + 3)/(x+1)

[^ EXP1 ENTP] ..: liste à 3 éléments pour une puissance.
ENTP est entier positif ou nul
exemple : $(x+1)^2$

[+ EXP1 EXP2 ... EXPn] :liste à n éléments (n entier > 1)
exemple : (2.x+3) + 5.x + 7

[* EXP1 EXP2 ... EXPn] :liste à n éléments (n entier > 1)
exemple : $5.x^2.y$

LES OUTILS à METTRE à la DISPOSITION des ELEVES :

L'élève pourra travailler avec plusieurs expressions (par ex, A = 2.x+1 ; B = x-3 ...) et définir une nouvelle expression à partir de celles déjà définies (par ex. C = A+B-x).
Il pourra à tout moment :

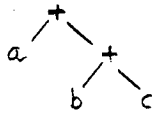
* obtenir la liste des expressions en cours
* modifier cette liste
* pour chaque expression obtenir sa structure au premier niveau (par ex. , pour A : somme de 2.x et de 1) , ainsi que la liste de ses sous-expressions (pour A : 2.x, 1, 2, x)
* substituer dans une expression une variable par sa valeur, ou un identificateur d'expression par sa valeur , ou une sous-expression par une autre expression.

Ces différents outils mis à la disposition des élèves soulèvent, entre autres problèmes, celui de la reconnaissance d'une sous-expression dans une expression donnée , ce qui , comme la reconnaissance de la structure de l'expression , nous paraît une notion fondamentale à acquérir. (remarque : si E = x+y+z, x+y , et à fortiori x+z , n'apparaîtra pas comme une sous-expression de E ; il faudra au préalable que l'élève provoque le regroupement nécessaire)
De nombreuses "erreurs" ne proviennent-elles pas d'une mauvaise reconnaissance des sous-expressions , d'une mauvaise reconnaissance des arguments d'un opérateur ? (dans x-y(x+y), x-y n'est pas une sous-expression)
L'étude informatique menée pour passer de l'écriture habituelle (2x-x(x+y)) à la structure.....
([- [* 2 x] [* x [+ x y]]]) a bien mis en relief les problèmes de reconnaissance de forme , les difficultés liées aux conventions d'écriture , à l'ordre de priorité des opérateurs et à l'usage des parenthèses.

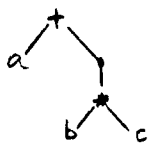
Quelques exemples simples :

a+b+c

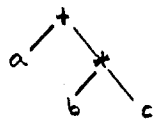
a+(b+c)



a+(bc)



a+bc



Les parenthèses ne traduisent pas " la profondeur " de l'expression qu'elles contiennent ; cette "profondeur" dépend encore de la nature de l'opération "chapeau" et de la structure de l'expression elle-même. Pour chaque sous-expression EI d'une expression E donnée, il est bon de porter son attention sur la nature de EI (EI est une somme , ou un produit , ou un quotient ...) et de se poser la question : de quelle opération EI est-elle un argument? On aboutit au "regard" suivant : dans 2(xy+5x) , xy est un produit , terme d'une somme. Ce regard, porté d'une part vers le "haut", d'autre vers le "bas", est également très important pour l'apprentissage correct des règles de calcul. (il suffit de penser par exemple aux simplifications à effectuer dans les fractions rationnelles :

Remarquons cependant qu'en attendant la généralisation de systèmes informatiques du type "mathcad" qui possède un éditeur permettant de façon simple l'écriture habituelle des expressions mathématiques et leurs visualisations habituelles , une difficulté supplémentaire sérieuse est introduite par l'obligation de l'écriture en ligne (2.X^2/(X+1)) : ce ne sont plus tout à fait les mêmes conventions qui sont utilisées (différence entre et A/B.C) et la lisibilité est bien différente d'un système à l'autre.

Dans l'une des versions du projet, nous avons cependant introduit une interface utilisateur qui écrit les expressions mathématiques sur plusieurs lignes (cas des exposants et des fractions).

LE SYSTEME muMATH DES TRANSFORMATIONS ALGEBRIQUES

TRANSFORMATIONS AUTOMATIQUES

Un certain nombre de transformations sont automatiquement et systématiquement appliquées.
·Les calculs sur les rationnels sont effectués.
·Les propriétés suivantes sont appliquées:

```
0 + A --> A     1 * A --> A     0 * A --> 0
A ^ 1 --> A     1 ^ A --> 1     A ^ 0 --> 1
```

·Dans les sommes et les produits , les parenthèses inutiles sont supprimées (utilisation de l'associativité) et les termes ou les facteurs sont rangés dans un ordre lié à leur nature (nombre, variable , fonction ...) et , pour les variables , à leur ordre d'introduction ...
·Termes semblables et facteurs semblables sont regroupés . (par exemples : 3 x + 2 x --> 5 x ; x^5 / x^2 --> x^3)

FONCTIONS DE TRANSFORMATION des expressions

Ces fonctions permettent à l'utilisateur de transformer des expressions pour leur donner une forme plus agréable , par l'application des règles de calcul habituelles (distributivité , développement , réduction au même dénominateur ...).Il n'est pas toujours possible à l'aide de ces fonctions d'obtenir la forme souhaitée ... ceci est un problème difficile ... Pour plus de souplesse il faudra utiliser les variables de contrôle (voir plus loin).

EXPAND (expr).....: numérateur et dénominateur sont développés ;
chaque terme du numérateur est divisé (et simplifié) par le dénominateur et <expr> est rendue comme la somme de ces fractions. Exemple :
EXPAND((x+1)^2/x) --> x+2+1/x

EXPD (expr).....: après réduction au même dénominateur , numérateur et dénominateur sont développés. Exemple :
EXPD(3+x/(x+1)^2) --> (3 x^2 + 7 x + 3)/(x^2 + 2 x + 1)

FCTR (expr).....: au numérateur comme au dénominateur , le plus grand commun diviseur est mis en facteur. Il ne s'agit pas contrairement à ce qui est suggéré par le nom choisi , d'une réelle factorisation ...! Exemple :
FCTR(6a*x+2a*x^2/b) --> 2 a x*(3 b + x)/b

DIVOUT(expr,sous-expr)..: simplifie des facteurs communs au numérateur et au dénominateur , contenant <sous-expr> , même si ceux-ci ne sont pas factorisés.
Par exemple : DIVOUT ((x^2 - y^2) / (x^2 + 2xy + y^2) , x) donnera (x-y)/(x+y).

PARFRAC (expr,sous-expr).: rend une décomposition en fractions rationnelles de <expr> (par rapport à <sous-expr>) , où les dénominateurs sont les facteurs reconnus du dénominateur de <expr>. Par exemple: PARFRAC (1/((x + 1)(x-1)),x); donnera
1/(-2 + 2 x) - 1/(2 + 2 x)

LES VARIABLES de CONTROLE ALGEBRIQUE

Exemple.: contrôle du développement de a(b+c)

Considérons la règle R : a(b+c) = ab + ac. Cette règle contient deux transformations :

T1 : a(b+c) --> ab + ac
T2 : ab + ac --> a(b+c)

C'est la valeur de la variable de contrôle **NUMNUM** qui va obliger le système à appliquer soit T1 , soit T2 , à chaque fois que le cas se présente.

·Premier principe :
Si NUMNUM = 0 , aucune transformation n'est appliquée.
Si NUMNUM > 0 , c'est T1 qui est appliquée.
Si NUMNUM < 0 , c'est T2 qui est appliquée.

·Deuxième principe :
L'application de R (T1 ou T2) peut se contrôler suivant la "nature" de a. On distingue trois cas :
C1 : a est une expression numérique (ex.: 4 , 1/3 , 2^0.5 ...)
C2 : a n'est , ni une expression numérique , ni une somme (ex.: x^3 , SIN(x) , y , ...)
C3 : a se présente comme une somme (ex.: x+y , x^2-3 , LN(x)+i)

Si NUMNUM est un multiple de 2 , R s'applique dans le cas C1.
Si NUMNUM est un multiple de 3 , R s'applique dans le cas C2.
Si NUMNUM est un multiple de 5 , R s'applique dans le cas C3.

Les différentes valeurs possibles pour NUMNUM (et qui induisent des transformations distinctes) sont donc :
-30,-15,-10,-6,-5,-3,-2,0,2,3,5,6,10,15,30.
La valeur par défaut de NUMNUM est 6.
Lors de l'affectation d'une expression à la variable E , la valeur de cette variable E sera l'expression transformée suivant l'état présent des variables de contrôle. Si , après affectation on désire connaître l'écriture de E avec d'autres valeurs pour les variables de contrôle , il faudra provoquer une nouvelle évaluation de E par EVAL(E);

Les variables de contrôle algébrique

Les principes ci-dessus s'appliquent aux variables suivantes :

variable	valeur > 0	valeur < 0	par défaut
NUMNUM	$a(b+c) \longrightarrow ab+ac$	$ab+ac \longrightarrow a(b+c)$	6
DENDEN	$\frac{1}{a} \times \frac{1}{b+c} \longrightarrow \frac{1}{ab+ac}$	$\frac{1}{ab+ac} \longrightarrow \frac{1}{a} \times \frac{1}{b+c}$	6
DENNUM	$\frac{b+c}{a} \longrightarrow \frac{b}{a} + \frac{c}{a}$	$\frac{b}{a}+c \longrightarrow \frac{b+ac}{a}$	6
NUMDEN	$\frac{a}{b+c} \longrightarrow \frac{1}{\frac{b}{a}+\frac{c}{a}}$	$\frac{1}{\frac{b}{a}+c} \longrightarrow \frac{a}{b+ac}$	0
BASEXP	$a^{(b+c)} \longrightarrow a^b . a^c$	$a^b . a^c \longrightarrow a^{(b+c)}$	30
EXPBAS	$(ab)^c \longrightarrow a^c . b^c$	$a^c b^c \longrightarrow (ab)^c$	30

D'autres variables de contrôle :

PWREXPD : sa valeur par défaut est 0. Si sa valeur est un multiple de 2 , les expressions (a+b+...)^n , avec n entier naturel non nul , sont développées ; si sa valeur est un multiple de 3 , les expressions (a+b+...)^(-n) , avec n entier naturel non nul , sont développées. Exemple : pour PWREXPD = 6 , on obtiendra

(a+1)^3/(x+1)^2 ---> (a^3+3a^2+3a+1)/(x^2+2x+1)

PBRCH : sa valeur par défaut est TRUE , et dans ce cas , même pour C non entier , la règle suivante est appliquée : (A^B)^C ---> A^(B^C). Si sa valeur est FALSE , alors cette règle n'est pas appliquée (pour C non entier) , ce qui peut être intéressant , car pour les exposants rationnels , il peut y avoir "logiquement" deux solutions. PBRCH est mis pour "Pick a BRanCH" (choisir une solution). Exemple :
Avec PBRCH = TRUE , (x^2)^(1/2) ---> x : tandis qu'avec PBRCH égale à FALSE , (x^2)^(1/2) est inchangé.

ZEROEXPT : si TRUE , applique A^0 ---> 1 (pour A non numérique) La valeur par défaut est TRUE.

ZEROBASE : si TRUE , n'applique pas 0^A ---> 0 (pour A non numérique) . La valeur par défaut est FALSE.

Lorsque le logiciel rencontre 0^0 ou 0^A avec A < 0 , un message est affiché et l'évaluation est interrompue (BREACK).

Le système des variables de contrôle de muMATH permet donc de contrôler, de façon assez fine, l'application des règles de calcul. Cependant :

certaines transformations (la plupart du temps souhaitables) se font automatiquement et ne peuvent pas être bloquées.

le système est assez "complexe" : il y a de nombreuses valeurs possibles pour de nombreuses variables de contrôle et le jeu des transformations standard est assez réduit

mais surtout, le grand inconvénient de ce système est que, une règle de calcul étant sélectionnée par les valeurs choisies pour les variables de contrôle, celle-ci s'applique à tous les endroits et à tous les niveau dans l'expression. On ne pourra pas appliquer telle associativité dans telle partie seulement de l'expression : c'est partout ou nulle part.

LE SYSTEME HP 28C de TRANSFORMATIONS ALGEBRIQUES

La calculatrice HP 28C possède un mode de calcul symbolique. Dans ce mode, un certain nombre de simplifications sont effectuées automatiquement et l'utilisateur dispose de "fonctions" permettant des transformations "licites" sur les expressions mathématiques.

LES SIMPLIFICATIONS AUTOMATIQUES . Elles ont lieu dans les cas suivants :

-(-x)	--->	x	INV(INV(x))	--->	x
0 + x	--->	x	x + 0	--->	x
x - 0	--->	x	0 - x	--->	-x
x - x	--->	0			
0 * x	--->	0	x * 0	--->	0
1 * x	--->	x	x * 1	--->	x
-1 * x	--->	-x	x * (-1)	--->	-x
-1 * (-x)	--->	x	-x * (-1)	--->	x
-x * INV(y)	--->	-(x/y)	-x * y	--->	-(x*y)
x * INV(y)	--->	x/y			
x/1	--->	x	0/x	--->	0
1/INV(x)	--->	x	1/x	--->	INV(x)
1^x	--->	1	x^0	--->	1
x^1	--->	x			

Certaines commandes modifient la forme d'une expression algébrique : nous n'examinons ici que celles d'entre elles qui ne modifient pas la "valeur" de l'expression.

LES TRANSFORMATIONS GLOBALES :

COLCT : (mis pour COLLECT) regroupe les termes numériques, évalue les sous-expressions numériques, combine les termes qui ne diffèrent que par leur coefficient numérique, combine les facteurs identiques ...Exemple : 1+x+2+3*x ---> 3 + 4*x . L'algorithme de réorganisation de COLCT a été choisi pour sa vitesse d'exécution et non pour sa conformité à une forme standard ou évidente.

EXPAN : (mis pour EXPAND) a pour tâche essentielle de developper. Plus précisément, EXPAND applique la distributivité de la multiplication (et de la division) par rapport à l'addition ((a+b)/c ---> a/c + b/c), développe les puissances dont les exposants sont des sommes (a^(b+c) ---> a^b*a^c), développe les puissances dont les exposents sont des entiers positifs (x^5 ---> x*x^4), développe le carré d'une somme. La commande EXPAND n'essaye pas d'effectuer tous les développements possibles en une seule fois ; elle travaille rang après rang dans la hiérarchie des sous-expressions en effectuant le premier développement possible.

LES TRANSFORMATIONS ELEMENTAIRES :

Grâce à la commande FORM, on a accès à un éditeur interactif d'expressions qui permet la réorganisation d'une expression algébrique conformément aux règles habituelles des mathématiques.
Dans ce mode le curseur se déplace sur les "objets" qui composent l'expression et non pas sur les caractères affichés (les parenthèses par exemple ne sont pas selectionnables par le curseur) ; il identifie cet objet qui apparaît en vidéo-inverse ainsi que la sous-expression qui le contient (sans affichage particulier).
L'affichage de l'expression diffère de l'affichage habituel : des parenthèses supplémentaires sont insérées pour expliciter l'ordre de priorite des opérateurs ; cela permet de reconnaître plus facilement la sous-expression sélectionnée, ce qui est primordial puisque toutes les opérations accesssibles dans ce mode FORM opèrent sur la sous-expression choisie.
Un objet (et la sous-expression qui le contient) étant choisi, on dispose d'un certain nombre d'opérations dont la liste dépend de l'objet choisi : toutes les transformations ne sont donc pas disponibles à tout moment ; seules les transformations pouvant s'appliquer à l'objet choisi sont proposées et la transformation n'est effectuée que si elle s'applique effectivement à la sous-expression selectionnée.
Exemple : si * est sélectionné dans (a + b)*c , la distributivité à droite est accessible et opérante ; si * est sélectionné dans (a/b)*c , la distributivité à droite est accessible mais non opérationnelle.

Les commandes COLCT et EXPAN sont encore disponibles, mais elles s'appliquent dans ce mode à la seule sous-expression sélectionnée.

Les transformations envisagées sont regroupées sous des dénominations globales (et symbolisées par un petit graphisme) On trouve ainsi :
Commutativité, associativité, et distributivité :

<---> : commutativité des arguments d'un opérateur (pour +-*/)
<--A : associativité à gauche (pour +-*/^)
-->A : associativité à droite (pour +-*/^)
-->(): distributivité de l'opérateur préfixe (pour - ou INV)
<--D : distributivité à gauche (pour */^)
D--> : distributivité à droite (pour */^)
<--M : mise en facteurs des arguments gauches (pour +-*/)
-->M : mise en facteurs des arguments droits (pour +-*/)

Double changement de signe et double-inversion :

DNEG : double changement de signe (pour un objet)
-() : DNEG suivi de -->() (pour +-*/)
DINV : double-inversion (pour un objet
1/() : DINV suivi de 1/() (pour */^)

Opérations neutres :

*1 : multiplication par 1 à droite (pour un objet)
/1 : division par 1 (pour un objet)
^1 : élévation à la puissance 1 (pour un objet)
+1-1: transforme A en (A + 1) - 1

Addition des fractions :

AF : réduction au dénominateur commun (pour +)

Voici la liste des transformations disponibles en fonction de l'opération sélectionnée :

Pour l'addition :

<---> :	(a+b)	(b+a)
	(-(a)+b)	(b-a)
<--A :	(a+(b+c))	((a+b)+c)
	(a+(b-c))	((a+b)-c)
A---> :	((a+b)+c)	(a+(b+c))
	((a-b)+c)	(a-(b-c))
<--M :	((a*b)+(a*c))	(a*(b+c))
M---> :	((a*c)+(b*c))	((a+b)*c)
	((a/c)+(b/c))	((a+b)/c)
-() :	(a+b)	-(-(a)-b)
	-(a)+b	-(a-b)
AF :	(a+(b/c))	(((a*c)+b)/c)
	((a/b)+c/d))	(((a*d)+(b*c))/(b*d))
	((a/b)+c)	((a+(b*c))/b)

Pour la soustraction :

<---> :	(a-b)	(-(b)+a)
<--A :	(a-(b+c))	((a-b)-c)
	(a-(b-c))	((a-b)+c)
A---> :	((a+b)-c)	(a+(b-c))
	((a-b)-c)	(a-(b+c))
<--M :	((a*b)-(a*c))	(a*(b-c))
M---> :	((a*c)-(b*c))	((a-b)*c)
	((a/c)-(b/c))	((a-b)/c)
-() :	(a-b)	-(-(a)+b)
	(-(a)-b)	-(a+b)
AF :	(a-(b/c))	(((a*c)-b)/c)
	((a/b)-c/d))	(((a*d)-(b*c))/(b*d))
	((a/b)-c)	((a-(b*c))/b)

Multiplication :

<--> :	(a*b)	(b*a)
	(INV(a)*b)	(b/a)
<--A :	(a*(b*c))	((a*b)*c)
	(a*(b/c))	((a*b)/c)
-->A :	((a*b)*c)	(a*(b*c))
	((a/b)*c)	(a/(b/c))
<--D :	((a+b)*c)	((a*c)+(b*c))
	((a-b)*c)	((a*c)-(b*c))
D--> :	(a*(b+c))	((a*b)+(a*c))
	(a*(b-c))	((a*b)-(a*c))
<--M :	((a^b)*(a^c))	(a^(b+c))
M--> :	((a^c)*(b^c))	((a*b)^c)
-() :	(a*b)	-(-(a)*b)
	(-(a)*b)	-(a*b)
1/() :	(a*b)	INV(INV(a)/b)
	(INV(a)*b)	inv(a/b)

Division :

<--> :	(a/b)	(INV(b)*a)
<--A :	(a/(b*c))	((a/b)/c)
	(a/(b/c))	((a/b)*c
A--> :	((a*b)/c)	(a*(b/c))
	((a/b)/c)	(a/(b*c))
<--D :	((a+b)/c)	((a/c)+(b/c))
	((a-b)/c)	((a/c)-(b/c))
D--> :	(a/(b+c))	INV((INV(a)*b)+(INV(a)*c))
	(a/(b-c))	INV((INV(a)*b)-(INV(a)*c))
<--M :	((a^b)/(a^c))	(a^(b-c))
-->M :	((a^c)/(b^c))	((a/b)^c)
-() :	(a/b)	-(-(a)/b)
	(-(a)/b)	-(a/b)
1/() :	(a/b)	INV(INV(a)*b)

Puissance :

<--A :	(a^(b*c))	((a^b)^c)
A--> :	((a^b)^c)	(a^(b*c))
<--D :	((a*b)^c)	((a^c)*(b^c))
	((a/b)^c)	((a^c)/(b^c))
D--> :	(a^(b+c))	((a^b)*(a^c))
	(a^(b-c))	((a^b)/(a^c))
1/() :	(a^b)	INV(a^-(b))
	(a^-(b))	INV(a^b)

Remarque : la transformation effectuée, le curseur se positionne sur un nouvel opérateur.

Ce système est donc très complet et permet toutes les transformations possibles, mais au prix de nombreux appels aux transformations très élémentaires disponibles.

LE SYSTEME FRATX de TRANSFORMATIONS ALGEBRIQUES

Ce système permet de travailler sur une fonction d'une variable définie par l'utilisateur. Les transformations accessibles, changeant la forme de l'expression sans en changer la valeur sont les suivantes :

E : simplifications évidentes

Il s'agit d'une transformation qui effectue les calculs numériques, regroupe les termes et facteurs semblables, ordonne les polynômes, ...

D : développement

Cette transformation développe les polynômes et extrait la partie entière des fractions rationnelles.

F : factorisation simple

Mise en facteur des facteurs communs, réduction au même dénominateur,...

I : factorisation intermédiaire

En plus des factorisations effectuées par F, cette transformation recherche les racines évidentes entre -3 et 3, factorise les polynômes du second degré, utilise les identités remarquables, recherche les racines multiples (avec le polynôme dérivé),...

M : factorisation maximale

Cette transformation utilise également la recherche de racines par des méthodes itératives (méthode de Bairstow, méthode de Newton pour un polynôme complexe, ...)

Ce logiciel n'est pas purement un logiciel de calcul formel, et les transformations accessibles ont un effet global puissant mais offrent peu de manipulations précises.

LE SYSTEME de TRANSFORMATIONS ALGEBRIQUES DU PROJET BRM

Le choix des transformations que nous souhaitons mettre à la disposition des élèves se fonde sur plusieurs préoccupations :

*rester proche des pratiques habituelles (non pas seulement créer un ensemble de transformations permettant de générer toutes les transformations possibles, mais proposer en priorité celles qui sont habituellement utilisées)

*introduire une souplesse d'utilisation variable suivant le niveau de l'élève ou les objectifs pédagogiques de l'activité (telle transformation "puissante" ne sera accessible qu'à partir d'un certain niveau)

*suivant sa nature, chaque transformation est prévue à 1, 2 ou 3 stades : sur l'expression en cours, sur une sous-expression de l'expression en cours ou sur un ensemble de sous-expressions sélectionnées à l'aide de leurs niveaux dans la structure.
Exemple : la suppression des termes nuls peut être demandée dans l'expression (lorsque celle-ci est une somme dont l'un des termes est nul) , ou dans une sous-expression de celle-ci, ou à tous les niveaux de l'expression.

*ne pas introduire trop de difficultés supplémentaires dues uniquement à un vocabulaire spécifique au système.

*rester proche du concept de structure d'une expression et proposer des transformations en fonction de la structure souhaitée pour l'expression transformée.

*regrouper les transformations par grands thèmes pour en faciliter l'accès : l'accès à une transformation donnée pourrait se faire par des choix successifs dans un jeu de "menus" et "sous-menus".(sans pour autant qu'il y ait unicité de l'accès à une transformation donnée).

Pour une situation donnée, (c'est-à-dire : une expression E et une sous-expression SE éventuelle étant sélectionnées) les transformations peuvent classées de la manière suivante :

1)les transformations qui ne peuvent pas s'appliquer (les structures de E et de SE ne sont pas conformes à celles attendues par la transformation)
2)les transformations qui peuvent s'appliquer, et alors :
2.1)les transformations qui ne sont pas actives (qui n'effectuent aucun changement sur l'expression)
2.2)les transformations actives, et alors :
2.2.1)les transformations non pertinentes (à l'égard de l'objectif visé)
2.2.2)les transformations pertinentes. (la reconnaissance de cette qualité risque d'ailleurs de ne pas être très simple)

Du point de vue de l'interface entre le système et l'utilisateur, il reste plusieurs choix à faire :

*le système doit-il commenter les inadéquations entre la situation et la transformation demandée ? (et si oui, quel type de commentaire ?)

*le système doit-il réaliser la transformation ou la demander à l'élève puis la vérifier ?
*le système doit-il proposer à tout moment toutes les transformations possibles, ou seulement les transformations adaptées à la situation ?

D'autre part , il est absolument indispensable d'imaginer et de réaliser un moyen simple permettant à l'élève de sélectionner une expression , une sous-expression , une transformation.

Substitution

Remplacement dans une expression E d'une sous-expression SE par une expression E1 (REMPL)
exemple: **(1+x)**+5*(x+y) ---> 2y+5*(x+y)
Remplacement dans une expression E d'un identificateur par une valeur. (SUB)
exemple: (1+x)+5*(x+y) ---> (1+4)+5*(4+y)
Remplacement dans une expression E de tous les identificateurs par leurs valeurs actuelles. (CALC)
exemple: 2*A+B+3 ---> 2*(x+y)+(x-y)+3

Simplification

Suppression de parenthèses : SUPAR
Il s'agit de la suppression des parenthèses d'une expression "sur-parenthésées"
exemple: (x+y) ---> x+y
x+(x+y) ---> x+(x+y) (sans changement : voir commutativité-associativité pour un changement)
((x+y))+z ---> (x+y)+z
(a*b)+c ---> a*b + c
Suppression d'un terme nul : SIMP0A
aussi bien dans le cas d'une addition que d'une différence
exemple: x+0+y ---> x+y
0-x ---> -x
x-0 ---> x
Annulation d'un produit contenant un facteur nul : SIMP0M
exemple: x*(x+y)*0 ---> 0
0/x ---> 0
(dans le cas d'une division par 0 : message d'erreur)
Simplification dans le cas d'une puissance : SIMP0P
exemple: 0^3 ---> 0
3^0 ---> 1
Simplification d'une opération dont un des argument est nul
SIMP0 : application de SIMP0A, SIMP0M et SIMP0P

Suppression du facteur 1 : SIMP1M
exemple: x*1*y ---> x*y
x/1 ---> x
Simplification dans le cas d'une puissance : SIMP1P
exemple: 1^3 ---> 1
x^1 ---> x
Simplification d'une opération dont un des argument est 1 :
SIMP1 : application de SIMP1M et de SIMP1P

Simplification d'une opération dont un des argument est 1 ou 0
SIMP01 : application de SUPAR, de SIMP0 et de SIMP1.

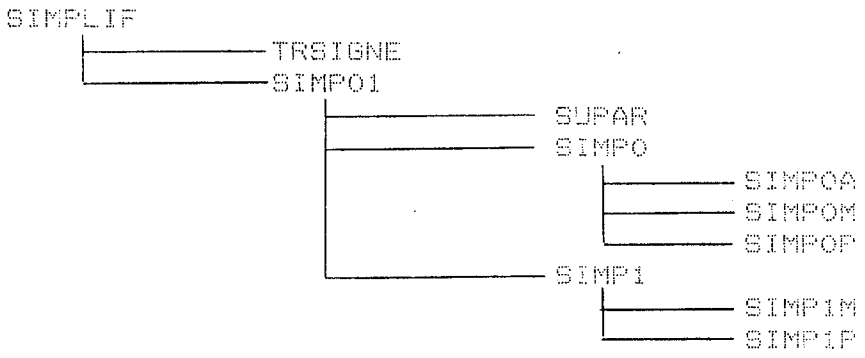
Transformation d'une expression en vue de simplifier l'intervention des signes : TRSIGNE.

Cette transformation est assez "complexe"; voici, sur des exemples, comment elle opère :

exemples:

```
-(-2x) ---> 2x
-x-y-(x+y) ---> -(x+y+(x+y))
-x+y+z ---> (y+z)-x
x(-y) ---> -x*y          x(-y)(-z) ---> x*y*z
-x-(-y) ---> y-x              -x-y ---> -(x+y)
x-(-y) ---> x+y          x/(-y) ---> - x/y
(-x)/(-y) ---> x/y            (-x)/y ---> - x/y
(-x)^2 ---> x^2          (-x)^3 ---> -x^3
```

Enfin, toutes ces transformations sont regroupées en une seule, **SIMPLIF**, qui applique TRSIGNE puis SIMP01. On obtient le schéma d'arbre suivant :



Chacune de ces transformations agit :

soit sur l'expression sélectionnée

soit sur la sous-expression sélectionnée d'une expression sélectionnée

On précisera en plus à quel niveau cette transformation doit agir : le niveau 0 signifie que l'action porte directement sur l'expression ou la sous-expression sélectionnée ; le niveau 3 par exemple signifie que l'action porte sur toutes les sous-expressions de niveau inférieur ou égal à 3 (dans la structure d'arbre) de l'expression ou de la sous-expression sélectionnée ; le niveau maximum signifie que l'action porte sur toutes les sous-expressions : enfin, tous les niveaux signifie une application récursive de la transformation avec substitution de l'expression par le résultat de la transformation lorsque celle-ci a agit.

Pour d'autres transformations contenant également une idée de simplification, voir calcul, commutativité et associativité, ainsi que changement de structure.

Changement de la structure

L'idée qui domine les transformations regroupées dans ce thème est la suivante : on porte son attention sur la structure de l'expression (c'est une somme, ou un produit, ...) et on désire la transformer en une structure prévue (une somme, un quotient, ...).Toutes les transformations à priori imaginables ne sont pas possibles ou ne revêtent pas un caractère unique (transformer x^2 en somme ! : x^2 + 0 ...); certaines

sont typiques d'objectifs classiques (x*(x+1) à transformer en somme par un développement); d'autres enfin nécessitent pour être intéressantes l'examen de la structure des arguments de l'opération de tête de l'expression sélectionnée (la transformation en somme de x*(x*y) -sic- n'est pas du même niveau que celle de x*(x+y)).

Le passage de somme en différence amène à distinguer parmi les sommes celles dont le nombre d'arguments est 2 de celles dont le nombre d'arguments est strictement supérieur à 2.

Voici une liste (incomplète) des transformations de cette catégorie :

Transformation d'une expression en un opposé : TREXOP
exemple: x-y ---> -(y-x)
a*b ---> -(-a*b)

Transformation d'une somme de 2 termes en une différence : TRSO2D1
exemple: x+y ---> x-(-y) x+(-y) ---> x-y

Transformation d'un opposé en somme : TROPSO
exemple: -(x-y) ---> -x+y

Transformation d'une différence en somme : TRDISO
exemple: x-y ---> x+(-y)

Transformation d'une somme en somme : TRSOSO
exemple: x+(y+z) ---> x+y+z

(on retrouvera les 3 transformations précédentes regroupées en une seule dans le cadre de l'associativité)

Transformation d'une puissance en produit : TRPUPR et TRPUPRN
exemple: (x+y)^3 ---> (x+y)*(x+y)^2 ou (x+y)*(x+y)*(x+y)

Transformation d'un quotient en produit : TRQUPR
exemple: (x*y)/z ---> x*y*(1/z)

Transformation d'un produit en produit : TRPRPR
exemple: x*(y*z) ---> x*y*z
(On retrouvera cette transformation dans le cadre de l'associativité)

Commutativité-associativité

Changement de l'ordre des termes dans une addition: obtenu en indiquant le nouvel ordre souhaité ; les termes sont repérés dans la somme sélectionnée par leurs rangs .
exemple: CHGTERM (x+2y-z , (1,3,2)) ---> x-z+2y

Changement de l'ordre des facteurs dans un produit: obtenu en indiquant le nouvel ordre souhaité ; les facteurs sont repérés dans le produit sélectionné par leurs rangs.
exemple: CHGFACT (x^2yz,(2,3,1)) ---> yzx^2

Regroupement des termes indiqués dans une somme donnée: obtenu en indiquant les rangs des termes dont on souhaite le regroupement.
exemple: REGTERM (x+2y-z , (1,3)) ---> (x-z)+2y

Regroupement des facteurs indiqués dans un produit donné: obtenu en indiquant les rangs des facteurs dont on souhaite le regroupement.
exemple: REGFACT (xy^2z , (1,3)) ---> $(xz)y^2$

Ces quatre transformations s'appliquent soit à l'expression sélectionnée, soit à la sous-expression sélectionnée dans une expression sélectionnée.

Les trois transformations suivantes agissent :
soit sur l'expression sélectionnée
soit sur la sous-expression sélectionnée d'une expression sélectionnée
On précisera également le niveau auquel cette transformation doit agir. (voir ci-dessus)

Transformation d'un produit dont l'un des facteurs est lui-même un produit: application de l'associativité de la multiplication.
exemple: TRPRPR ((2x)y) ---> 2xy

Transformation d'une expression additive (somme, différencé ou opposé) dont l'un des termes est lui-même une expression additive: application de l'associativité de l'addition.
exemple: TRADADSO ((x-y)+2z) ---> x-y+2z
a+(b-c)-(d-e) ---> a+b-c-d+e

Ces deux transformations sont regroupées en une seule: SIMPAR

Calcul

Regroupement des termes (ou facteurs) numériques dans une somme (ou un produit): REGNUM
exemple: x+3-y-1 ---> (3-1)+x-y
x*2*y*3 ---> (2*3)*x*y

Calcul d'une expression numérique (ne contenant que des entiers : calcul sur des rationnels ; résultat rationnel): CALCNUM
exemple: (3-1)/4 ---> 1/2

Regroupement des termes ou facteurs numériques et calcul du regroupement opéré: REGNUMCALC
exemple: x+3-y-1 ---> 2+x-y
x*2*y*3 ---> 6*x*y

On appellera terme numériquement semblable à un expression donnée E, une expression se présentant sous la forme k*E, où k est un nombre.
On appellera facteur semblable à une expression donnée E, une expression se présentant sous la forme E^n, où n est un entier naturel.

Pour les transformations regroupées dans le thème **calcul**, "semblable" signifiera numériquement semblable.

Regroupement des termes semblables à une expression donnée: REGTSA
exemple: x+y+2x,x ---> (1+2)x+y

Regroupement des termes semblables à une expression donnée et calcul du regroupement: REGTSACALC
exemple: x+y+2x,x ---> 3x+y

Regroupement des facteurs semblables à une expression donnée: REGFSA (et calcul du regroupement: REGFSACALC):
exemple: $x*y*x^2$,x ---> x^3*y

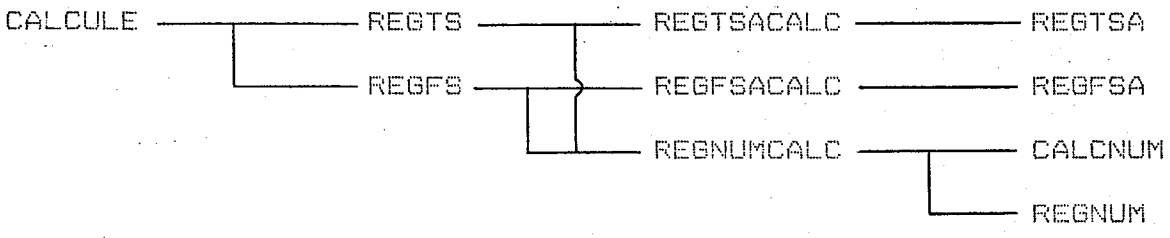
Regroupement des termes semblables (avec calcul): REGTS
exemple: 2x+y+x-3y ---> 3x-2y
x+xy+3x+2xy ---> 4x+3xy

Regroupement des facteurs semblables (avec calcul): REGFS
exemple: $x(x+y)x^2(x+y)$ ---> $x^3(x+y)^2$

Enfin CALCULE applique REGTS ou REGFS suivant le cas.
Si la variable de contrôle #calcule (voir ci-dessous) a une valeur supérieure ou égale à 3, CALCULE agit de la façon suivante : application de REGTS après application de l'associativité de l'addition puis application de REGFS sur chaque terme de la somme obtenue après application de l'associativité de la multiplication.

Chacune de ces transformations agit :
soit sur l'expression sélectionnée
soit sur la sous-expression sélectionnée d'une expression sélectionnée
On précisera également le niveau auquel cette transformation doit agir (voir ci-dessus).

On obtient finalement le schéma suivant pour ces transformations:



Les transformations ci-dessus sont à compléter avec CALC vu lors des substitutions.

Développement

L'ensemble des transformations de ce thème est régit, quant aux "simplifications" exécutées automatiquement, par la valeur d'une variable de contrôle: #calcule.
Lorsque la valeur de celle-ci est FALSE, aucune transformation (autre que celle du développement demandé) n'est effectuée.

Pour la valeur 0, les facteurs numériques dans un produit de 2 facteurs sont mis "en avant" :
exemple: x*2 ---> 2x
Pour la valeur 1, on a en plus l'application de la règle des signes et le calcul du produit lorsque celui-ci est numérique:
exemple: 2*3 ---> 6 x*(-2) ---> -2x
Pour la valeur 2, on a en plus le regroupement des facteurs semblables:
exemple: x*x ---> x^2
Pour la valeur 3, on obtiendra en plus des transformations précédentes réalisées après l'application de l'associativité de la multiplication,, l'application du regroupement des termes numériquement semblables:
exemple: x+x*(2) ---> 3x
Pour la valeur 9, c'est l'évaluation du logiciel muMATH qui sera appliquée.

Développement du produit de 2 facteurs dont chacun est une somme de 2 termes: DIST22SO
exemple: (a+b)(c+d) ---> ac+ad+bc+bd

Développement du produit de 2 facteurs dont chacun est une somme ou une différence de 2 termes: DIST22
exemple: (a+b)(c+d) ---> ac+ad+bc+bd
(a-b)(c+d) ---> ac+ad-bc-bd
(a-b)(c-d) ---> ac-ad-bc+bd
(a+b)(c-d) ---> ac-ad+bc-bd

Application de la distributivité à gauche: DISTG
exemple: (a+b-c)*d ---> ad+bd-cd

Application de la distributivité à droite: DISTD
exemple: a(b-c+d) ---> ab-ac+ad

Application complète de la distributivité sur un produit de 2 expressions additives (somme de 2 ou plusieurs termes, ou différence de 2 termes): DIST2N
exemple: (a+b)(c-d+e) ---> ac-ad+ae+bc-bd+be

Application de l'identité remarquable $(a+b)(a-b)=a^2-b^2$ et de $(a-b)(a+b)=a^2-b^2$: DVPID1

Application de l'identité remarquable $(a+b)^2=a^2+2(ab)+b^2$ et de $(a-b)^2=a^2-2(ab)+b^2$:DVPID2

Développement d'un produit de 2 facteurs (ou du carré d'une somme ou d'une différence de 2 termes), en appliquant éventuellement les identités remarquables: DIST2

Développement d'un produit de 2 ou plusieurs facteurs, par application récursive de DIST2: DISTITER

Développement d'un carré d'une somme de 2 ou plusieurs termes, ou d'une différence : DVPPU2

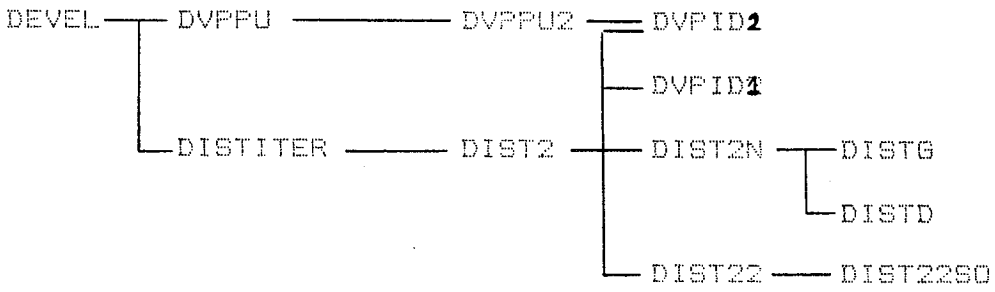
Développement d'une expression additive (somme ou différence) élevée à une puissance entière naturelle : DVPPU

Enfin, la transformation DEVEL applique suivant le cas DVPPU ou DISTITER.

Les transformations DVPPU2, DIST2, DISTITER, DVPPU et DEVEL peuvent agir :
soit sur l'expression sélectionnée
soit sur la sous-expression sélectionnée d'une expression sélectionnée
On précisera également le niveau auquel cette transformation doit agir (voir ci-dessus).

Les autres transformations agissent :
soit sur l'expression sélectionnée
soit sur la sous-expression sélectionnée d'une expression sélectionnée

On obtient finalement le schéma suivant:



Il resterait peut-être à créer également la transformation permettant la distribution de telle expression désignée sur telle autre expression sélectionnée.

Il reste également à développer l'ensemble des transformations qui correspondent aux :
Règles sur les quotients
Règles sur les exposants
ainsi qu'à la
Factorisation (avec le point de vue suivant : factoriser telle expression dans telle expression donnée; par exemple, permettre la factorisation de x+1 dans l'expression (x+1)+2a(x+1) ou de 2ax dans $6ax^2-2axy$.)

Dans ce document les transformations portent les noms de leurs réalisations en "muSIMP" (langage du type LISP, dans lequel a été écrit "muMATH").Ce ne sont pas bien sûr ces noms qui doivent être utilisés pour communiquer avec les élèves; les transformations seront désignées par leurs actions et accompagnées , si possible , par un exemple type.
L'ensemble des transformations ainsi opérationnelles est trop vaste et trop complexe. Pour chaque séquence de travail, et en fonction des objectifs fixés, il ne faudra rendre accessibles que les transformations pertinentes. Pour une initiation par exemple, on mettra à la disposition de l'élève les transformations les plus élémentaires en lui interdisant l'accès aux transformations les plus élaborées; dans une phase ultérieure on laissera l'accès aux transformations avancées en supprimant les transformations élémentaires inutiles. On peut imaginer que le logiciel offre lui-même des "configurations" initiales par défaut pour en permettre une utilisation simple.