Internet et calcul formel

Dans son utilisation de base, un logiciel de calcul formel tente de donner une réponse mathématique à une question qu'on lui pose. Cela sous-entend :

- que l'utilisateur pose une question mathématique *recevable* ;
- qu'il connaît la syntaxe permettant de poser cette question au logiciel ;
- qu'il connaît suffisamment le langage mathématique pour interpréter la réponse fournie ;
- qu'il connaît suffisamment le logiciel pour comprendre les *réponses* (réactions du logiciel) éventuellement surprenantes ou les messages d'erreur que le logiciel serait amené à produire.

Plaçons-nous dans un cadre d'utilisation du logiciel *Mathematica,* à savoir celui d'une utilisation pédagogique...

On s'intéresse ici au professeur qui voudrait intégrer un logiciel de calcul formel dans des scénarios pédagogiques, autrement qu'en faisant appel aux commandes de base du logiciel...

Certains résultats renvoyés par le logiciel coïncident avec l'idée qu'on peut se faire du résultat :

```
1 + 1
```

2

```
Expand[(a + b)^2]
```

$a^2 + 2\,b\,a + b^2$

On ne comprend pas trop pourquoi on n'obtient pas $a^2 + 2\,a\,b + b^2$, mais peu importe !

Les élèves ne seront peut-être pas gênés et ils ne seront pas choqués (mais ils le devraient...) de ne pas voir apparaître de signe d'égalité !

Effectuons la démarche inverse, qui consiste à factoriser une expression :

```
Factor[(a + b)^2]
```

$(a + b)^2$

```
Factor[a^2 + 2 * a * b + b^2]
```

$(a + b)^2$

```
Factor[x^2 - 4]
```

$$(x-2)(x+2)$$

```
Factor[x^2/4 - 9]
```

$$\frac{1}{4}(x-6)(x+6)$$

Premier résultat non attendu !

Continuons...

```
Factor[x^2 - 2]
```

$$x^2-2$$

Et de deux...

Pour mieux comprendre comment fonctionne cette commande Factor, on peut demander de l'aide :

```
? Factor
```

```
Factor[poly] factors a polynomial over the integers.
   Factor[poly, Modulus->p] factors a polynomial modulo
   a prime p. Factor[poly, Extension->{a1, a2, ... }]
   factors a polynomial allowing coefficients that are
   rational combinations of the algebraic numbers ai. Plus...
```

```
?? Factor
```

```
Factor[poly] factors a polynomial over the integers.
   Factor[poly, Modulus->p] factors a polynomial modulo
   a prime p. Factor[poly, Extension->{a1, a2, ... }]
   factors a polynomial allowing coefficients that are
   rational combinations of the algebraic numbers ai. Plus...

Attributes[Factor] = {Listable, Protected}

Options[Factor] = {Extension → None,
  GaussianIntegers → False, Modulus → 0, Trig → False}
```

On comprend alors mieux pourquoi on a obtenu les deux réponses précédentes...

On peut alors améliorer la commande de factorisation :

```
terme = Input["Entrez l'expression à factoriser"];
attribution = Solve[terme == 0, x];
solutions = x /. attribution;
factorisation = Factor[terme, Extension → solutions];
If[solutions ∈ Reals, SequenceForm["Pour tout réel ", x,
  ", ", terme, " = ", factorisation], SequenceForm[
  "On ne sait apparemment pas encore factoriser ",
  terme, " ."]]
```

Pour tout réel x, $x^2 - 4 = (x-2)(x+2)$

Voire élaborer un petit programme qui créerait une expression d'une forme donnée, proposerait à l'utilisateur de la factoriser et vérifierait que la solution proposée est correcte.

Essayons...

```
nbalea := Random[Integer, {1, 10}];
generer := (choix = Random[Integer, {1, 3}];
  Which[
   choix == 1,
   expr = Expand[(nbalea * x + nbalea)^2],
   choix == 2,
   expr = Expand[(nbalea * x - nbalea)^2],
   choix == 3,
   expr = nbalea^2 * x^2 - nbalea^2];
  SequenceForm[
   "Donnez une expression factorisée de ",
   a[x], " = ", expr])
```

```
generer
```

Donnez une expression factorisée de $a(x) = 16\,x^2 - 64\,x + 64$

```
generer
```

Donnez une expression factorisée de $a(x) = 36\,x^2 - 16$

generer

Donnez une expression factorisée de $a(x) = 81\,x^2 - 64$

generer

Donnez une expression factorisée de $a(x) = 9\,x^2 + 18\,x + 9$

Mais on se heurtera à une difficulté de taille : Comment tester que la réponse proposée est effectivement une forme factorisée ?

Mathematica dispose d'un nombre important de tests d'expressions et deux d'entre eux vont plus particulièrement nous intéresser : Equal (==) et SameQ (===) :

- Une équation d'inconnue réelle x écrite sous la forme $a(x) == b(x)$ renvoie True lorsque l'ensemble solution de l'équation $a(x) = b(x)$ est $\mathbb{R}$;
- $a(x) === b(x)$ renvoie True lorsque les deux expressions sont syntaxiquement identiques.

Apprécions la nuance sur quelques exemples :

```
(x - 2)^2 == (x - 2)^2
```

True

```
(x - 2)^2 === (x - 2)^2
```

True

```
(2 - x)^2 == (x - 2)^2
```

$(2 - x)^2 == (x - 2)^2$

```
(2 - x)^2 === (x - 2)^2
```

False

```
(x - 2) (x + 2) == (x - 2) (x + 2)
```

True

```
(x + 2) (x - 2) == (x - 2) (x + 2)
```

True

```
(x - 2/3) (x + 2/3) == Factor[x^2 - 4/9]
```

$$\left(x-\frac{2}{3}\right)\left(x+\frac{2}{3}\right)==\frac{1}{9}(3x-2)(3x+2)$$

```
(x - 2/3) (x + 2/3) === Factor[x^2 - 4/9]
```

False

Avez-vous compris où est le problème ?

On aurait pu s'en douter :

- parce qu'on sait que *Mathematica* donne, lorsque c'est possible, une factorisation à coefficients entiers ;
- parce qu'on sait surtout que la factorisation d'une expression n'est pas unique !!!

Finalement, ce qu'on souhaite, c'est revenir sur le principe même de la factorisation, à savoir d'écrire une expression sous forme d'un produit ou d'une puissance...

Pour cela, on dispose d'une fonction très utilisée sur le site académique...

```
forme[1] = (a + b)^2;
FullForm[forme[1]]
TreeForm[forme[1]]
```

Power[Plus[a, b], 2]

Power[| , 2]
Plus[a, b]

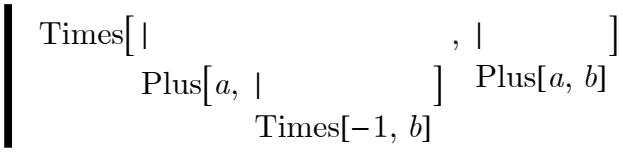
```
forme[2] = a^2 + 2 a b + b^2;
FullForm[forme[2]]
TreeForm[forme[2]]
```

Plus[Power[a, 2], Times[2, a, b], Power[b, 2]]

Plus[| , | , |]
Power[a, 2] Times[2, a, b] Power[b, 2]

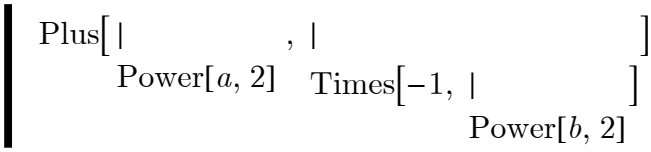
```
forme[3] = (a - b) * (a + b);
FullForm[forme[3]]
TreeForm[forme[3]]
```

```
Times[Plus[a, Times[-1, b]], Plus[a, b]]
```



```
forme[4] = a^2 - b^2;
FullForm[forme[4]]
TreeForm[forme[4]]
```

```
Plus[Power[a, 2], Times[-1, Power[b, 2]]]
```



Il ne reste donc plus qu'à tester si l' « en-tête » de l'expression proposée par l'élève est celle d'un produit ou d'une puissance...

- Si l' « en-tête » de l'expression élève est Times ou Power et si son développement est identique (===) à celui proposé par le logiciel, la réponse fournie est une réponse correcte ;
- sinon, la réponse fournie est incorrecte et on peut mettre en place un système d'aide.

Petite remarque : si les deux expressions sont identiques, on peut indiquer à l'élève que le travail ne consiste pas à recopier une expression...

Comme tout outil, un logiciel de calcul formel recèle quelques fonctionnalités imprévues et son comportement peut parfois étonner.

Plaçons-nous au niveau terminale, et utilisons « naturellement » les fonctions de référence logarithme et exponentielle...

```
Exp[1]
```

e

```
Exp[Log[2]]
```

2

```
Exp[-3]
```

$$\frac{1}{e^3}$$

```
Log[e]
```

$$1$$

```
Log[1]
```

$$0$$

```
Log[3]
```

$$\log(3)$$

```
Log[10, 3]
```

$$\frac{\log(3)}{\log(10)}$$

```
Log[-1]
```

$$i\,\pi$$

Les ennuis continuent...

Le résultat peut paraître surprenant si l'on ignore que certains autorisent : pour tout $z \in \mathbb{C}^*$, $\log z = \log r + i\varphi$ où $r = |z|$ et $\varphi = \operatorname{Arg} z$...

```
Sqrt[2]
```

$$\sqrt{2}$$

```
Sqrt[4]
```

$$2$$

```
Sqrt[-1]
```

$$i$$

La notation utilisée (raccourci de la fonction Sqrt) $\sqrt{x}$ est interprétée comme une puissance par le logiciel $\left(x^{\frac{1}{2}}\right)$ et l'on retrouve le problème précédent...

```
FullForm[Sqrt[x]]
```

Power[x, Rational[1, 2]]

Quel est l' « en-tête » d'expressions telles que $\ln(x)$, $\exp(x)$, $\sin(x)$, ... , ?

```
Head[Log[x]]
```

Log

```
Head[Exp[x]]
```

Power

```
Head[Sin[x]]
```

Sin

Cette fonction élémentaire Head donne une idée : pour adapter les fonctions élémentaires du logiciel à des exigences mathématiques adaptées au cursus, une technique informatique «simple» consiste... à lui apprendre à faire des mathématiques de niveau lycée !

On s'appuie sur les résultats suivants :

– la fonction ln est définie sur $]0 ; +\infty[$;

– la fonction racine carrée est définie sur $[0 ; +\infty[$;

– la fonction inverse est définie sur $]-\infty ; 0[\cup]0 ; +\infty[$.

Et on va programmer *Mathematica* pour effectuer les mêmes opérations qu'un élève rigoureusement formé par son enseignant, à savoir :

– rechercher ces différentes formes dans l'expression introduite par l'utilisateur ;

– en extraire le contenu ;

– poser les conditions d'existence ;

– résoudre ces conditions ;

pour ensuite obtenir l'ensemble de définition de la fonction définie par l'utilisateur.

Le principe est le suivant : on décompose toute fonction numérique en composée de fonctions élémentaires, et, à chaque fois que l'une des trois fonctions ln, inverse et racine carrée est rencontrée, on extrait l'expression qui lui correspond et on définit la condition qui doit être vérifiée.

Les fonctions élémentaires les plus fréquemment utilisées à ce niveau sont :

– Plus$[a, b, ...] = a + b + ...$

– Times$[a, b, ...] = a \times b \times ...$

– Power$[a, b] = a^b$

– Rational$[a, b] = \frac{a}{b}$ avec $a \in \mathbb{Z}^*$ et $b \in \mathbb{N}^*$

– Log$[a] = \ln(a)$

toute autre expression algébrique se déduisant de celles-ci.

Tout cela peut avoir un intérêt pédagogique ; si l'on n'y prend pas garde, les résultats affichés par *Mathematica* peuvent se révéler faux (au lycée) :

```
Solve[Log[x^2 - 2 x] == Log[3 x - 4], x]
```

$\{\{x \to 1\}, \{x \to 4\}\}$

Les résultats obtenus sont bien les solutions de l'équation, mais dans $\mathbb{C}$!

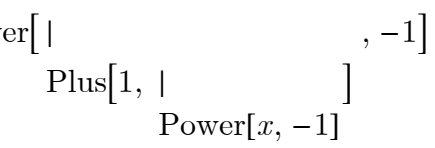
Finalement, deux possibilités s'offrent à l'utilisateur enseignant de mathématiques en lycée :

- Vérifier, comme il apprend à le faire à ses élèves, les résultats obtenus ;
- programmer le logiciel pour l'adapter à ses propres exigences.

```
f[x_] := 1/(1 + 1/x)
```

```
FullForm[f[x]]
TreeForm[f[x]]
```

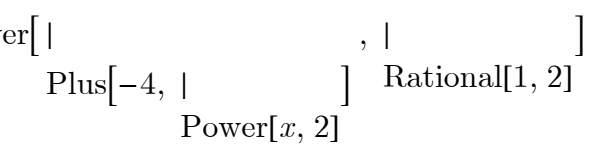
Power[Plus[1, Power[x, −1]], −1]




```
f[x_] := Sqrt[x^2 - 4]
```

```
FullForm[f[x]]
TreeForm[f[x]]
```

Power[Plus[−4, Power[x, 2]], Rational[1, 2]]




```
f[x_] := Log[Exp[x] - 1]
```

```
FullForm[f[x]]
TreeForm[f[x]]
```

```
Log[Plus[-1, Power[E, x]]]
```

```
Log[ |                    ]
    Plus[-1, |           ]
             Power[e, x]
```

L'utilisation d'un logiciel de calcul formel tel que *Mathematica* pose d'autres problèmes. Parce qu'il s'adresse à une communauté scientifique internationale, ce dernier ne prend pas en compte les spécificités de l'enseignement français des mathématiques.

On a déjà rencontré le problème lorsqu'on a effectué des calculs logarithmiques :

```
Log[2]
```

```
log(2)
```

Il en va de même pour les intervalles : la notation internationale est la même qu'en France pour les intervalles fermés, mais les intervalles semi-ouverts ou ouverts sont notés respectivement : $(a; b]$, $[a; b)$, $(a; b)$ (par analogie avec les notations géométriques des demi droites ?).

Le traitement mathématique des inéquations pose d'autres problèmes lorsqu'on utilise *Mathematica* :

```
<< Algebra`InequalitySolve`
```

```
InequalitySolve[x - 1 > 0, x]
```

$x > 1$

```
InequalitySolve[(x - 1)^2 > 0, x]
```

$x < 1 \vee x > 1$

```
InequalitySolve[(x - 1)^2 ≥ 0, x]
```

True

```
InequalitySolve[(x - 1)^2 < 0, x]
```

False

```
InequalitySolve[(x - 1)^2 ≤ 0, x]
```

$x == 1$

```
InequalitySolve[(x - 1)^2 (x + 2) ≤ 0, x]
```

$x \leq -2 \vee x == 1$

```
InequalitySolve[x^5 + x + 1 > 0, x]
```

$x > \text{Root}[\#1^3 - \#1^2 + 1 \,\&, 1]$

Il faut donc apprendre au logiciel à traduire ces propositions logiques en termes de singletons, d'intervalles, de réunions de tels ensembles... ou de réponse adaptée signalant que le logiciel ne parvient pas à résoudre l'inéquation proposée.

Puisque nous en sommes aux problèmes de traduction, signalons quelques difficultés, résolues ici et sur le serveur académique :

Le premier est celui qui a été soulevé en début de présentation (traduction de log en ln, ...).

Le second est plus délicat : *Mathematica*, comme tout logiciel, possède sa propre syntaxe : les fonctions élémentaires sont écrites en anglais et commencent par une majuscule, les arguments de ces fonctions sont écrits entre crochets.

On peut chercher à harmoniser cette syntaxe avec :

- la syntaxe mathématique française ;
- la syntaxe mathématique traditionnelle ;
- la syntaxe des tableurs les plus fréquemment utilisés.

C'est ce qui a été réalisé sur le serveur euler dans les derniers scénarios mis en ligne.

Mais l'utilisateur, même averti, peut encore se laisser surprendre : sur le serveur, demandez, par exemple, de déterminer une primitive (ou la dérivée) de la fonction $f : x \mapsto x(x+1)$ écrite sous cette forme et essayez d'analyser la réponse du logiciel...

Le calcul d'une dérivée ne pose pas de problème, encore faut-il choisir la forme affichée la plus adaptée : doit-elle être simplifiée, développée, factorisée, ou laissée telle quelle ?

L'étude du signe de la dérivée s'effectuera sur l'ensemble de définition préalablement obtenu par le programme. Sinon, on obtiendrait des résultats comme celui-ci :

```
f[x_] := Log[x (x - 1)]
InequalitySolve[f'[x] ≥ 0, x]
```

$$0 < x \leq \frac{1}{2} \bigvee x > 1$$

Suffit-il de résoudre le système d'équations et d'inéquations $\begin{cases} x \in \mathrm{E} \\ f'(x) \geq 0 \end{cases}$ pour connaître les intervalles sur lesquels la fonction est croissante ? La réponse est non...

```
f[x_] := Log[x]^3 / x^2
```

```
InequalitySolve[f'[x] ≤ 0, x]
```

InequalitySolve::npi : A nonpolynomial equation or inequality encountered. The solution set may be incorrect.

$$x == 1 \bigvee x \geq e^{3/2}$$

Dans certains cas, des résultats non attendus à ce niveau peuvent apparaître :

```
f[x_] := Exp[2 x] - (x + 1) Exp[x]
f'[x]
```

$$-e^x (x+1) - e^x + 2\, e^{2x}$$

```
Simplify[f'[x]]
```

$$e^x (-x + 2\, e^x - 2)$$

```
InequalitySolve[f'[x] ≤ 0, x]
```

InequalitySolve::npi : A nonpolynomial equation or inequality encountered. The solution set may be incorrect.

$$-2 - \mathrm{ProductLog}\left[-\frac{2}{e^2}\right] \leq x \leq -2 - \mathrm{ProductLog}\left[-1, -\frac{2}{e^2}\right]$$

ProductLog[y] donne la solution principale de l'équation d'inconnue x : $y = x\,\mathrm{e}^x$.

Il y a un autre problème : dans certains cas, les variations d'une fonction numérique ne peuvent être obtenues que par l'étude du signe d'une fonction auxiliaire, et il faudra l'indiquer.

```
f[x_] := x^3 / 3 - 3 x - 9 Exp[-x / 3]
f'[x]
```

$$x^2 + 3\, e^{-x/3} - 3$$

```
InequalitySolve[f'[x] ≥ 0, x]
```

InequalitySolve::npi : A nonpolynomial equation or inequality encountered. The solution set may be incorrect.

InequalitySolve::nineq : $x^2 + 3\,e^{-x/3} - 3 \geq 0$ is not a formula constructed with univariate polynomial equations and inequalities in x.

```
InequalitySolve(x^2 + 3 e^(-x/3) - 3 ≥ 0, x)
```

Notons respectivement P, P_n, N, N_n les ensembles solutions des systèmes suivants :

$$P : \begin{cases} x \in \mathrm{E} \\ f'(x) > 0 \end{cases} \qquad P_n : \begin{cases} x \in \mathrm{E} \\ f'(x) \geq 0 \\ f''(x) \neq 0 \end{cases}$$

$$N : \begin{cases} x \in \mathrm{E} \\ f'(x) < 0 \end{cases} \qquad N_n : \begin{cases} x \in \mathrm{E} \\ f'(x) \leq 0 \\ f''(x) \neq 0 \end{cases}$$

• Si $P = \emptyset$ et $N_n = \mathbb{R}$,

f est strictement décroissante sur $\mathbb{R}$.

• Si $P_n = \mathbb{R}$ et $N = \emptyset$,

f est strictement croissante sur $\mathbb{R}$.

• Si $P_n \neq \mathbb{R}$ et $N_n = \emptyset$,

f est strictement croissante sur chacun des intervalles inclus dans E sur lesquels $f'(x) \geq 0$ (et il faut supprimer les singletons solutions éventuelles de cette inéquation).

• Si $P_n = \emptyset$ et $N_n \neq \mathbb{R}$,

f est strictement décroissante sur chacun des intervalles inclus dans E sur lesquels $f'(x) \leq 0$ (et il faut supprimer les singletons solutions éventuelles de cette inéquation).

• Si $P_n \neq \mathbb{R}$, $N_n \neq \mathbb{R}$ et les deux inéquations peuvent être résolues algébriquement,

f est strictement croissante sur chacun des intervalles inclus dans E sur lesquels $f'(x) \geq 0$ (et il faut supprimer les singletons solutions éventuelles de cette inéquation) et f est strictement décroissante sur chacun des intervalles inclus dans E sur lesquels $f'(x) \leq 0$ (en supprimant les singletons solutions éventuelles de cette inéquation).

• Si les deux inéquations ne peuvent être résolues algébriquement, un texte doit indiquer qu'on

peut étudier le signe d'une fonction auxiliaire.

Les extrema de f sont les solutions sur E du système $\begin{cases} f'(x) = 0 \\ f''(x) \neq 0 \end{cases}$.