Sur la correction automatique

des exercices de mathématiques

Conférence donnée au séminaire
Moocs : de la correction automatique à la personnalisation des cursus
INRIA 13 Janvier 2014

Alain Prouté([1])

Ce texte est téléchargeable à l'adresse :
`http://www.logique.jussieu.fr/~alp/conf-MOOC-INRIA.pdf`

Table des matières

1 Qu'est-ce qu'un exercice de mathématiques ?

Il est en fait plus pertinent de demander ce qu'est une « question » d'un exercice de mathématique. Il en existe de plusieurs sortes. La classification ci-dessous n'a rien d'officiel. Elle résulte juste de ma propre expérience de l'enseignement.

type 1 : Le plus souvent, une question commence par l'expression « Montrer que… » (ou une expression équivalente). Ce qui suit cette expression est un « énoncé » au sens où on l'entend en mathématiques. La « solution » attendue pour une telle question n'est rien d'autre qu'une démonstration de cet énoncé. Une question d'un exercice de mathématiques a donc le plus souvent la même forme qu'un « théorème », c'est-à-dire un couple formé d'un énoncé et d'une démonstration de cet énoncé. Bien sûr, on ne fournit que l'énoncé et on en attend « une » démonstration([2]) de la part de l'étudiant.

type 2 : Il arrive qu'on demande à l'étudiant de « déterminer » ou de « décrire » l'unique objet ayant telle ou telle propriété. Par exemple : *Déterminer le point d'intersection des droites D et D'.* Ce qui est demandé est en realité une expression décrivant cet objet. Par exemple, dans le cas d'un point intersection de deux droites, il peut s'agir d'un point dont le nom est déjà connu (et c'est ce nom qu'on attend), ou de coordonnées cartésiennes, etc…, et souvent on ne précise pas sous quelle forme la réponse doit être donnée. Il se peut aussi que l'objet en question ne soit pas unique. Dans ce cas, il y a de nombreuses réponses valables. Bien souvent, pour éviter d'avoir à vérifier dans des copies des calculs tous différents, on pose les questions de telle façon qu'il y ait unicité de la solution. Cette précaution n'est plus nécessaire en correction automatique (l'ordinateur étant en principe infatigable). Il se peut aussi que la réponse doive être justifiée. On écrit souvent dans les énoncés des problèmes des phrases comme : *On justifiera soigneusement chaque réponse.* Dans ce cas, on attend non seulement une expression décrivant l'objet, mais la preuve d'un énoncé de la forme $E[a]$ où a est l'objet en question et où $x \mapsto E[x]$ est le prédicat caractérisant (pas forcément de manière unique) cet objet.

1. Université Paris-Diderot
2. Et non pas « la » démonstration, ce qui est bien sûr un point essentiel, et c'est pour cela qu'on a besoin d'un algorithme qui, d'une certaine façon, « comprend » les démonstrations.

type 3 : Il s'agit de la « question de cours », dans laquelle on ne demande pas de démonstration mais seulement l'énoncé d'un théorème ou la définition d'un concept. La difficulté inhérente à ce genre de question pour le concepteur de l'exercice est de trouver la bonne façon de désigner sans ambiguïté le théorème ou la définition sans en révéler le contenu. Généralement, on n'utilise ce genre de question que pour les théorèmes ou concepts ayant un nom. Par exemple « Donner la définition d'un groupe » (sous-entendu, de « la structure de groupe »), ou « Énoncer le théorème de d'Alembert ». Mais on dispose d'autres moyens, comme par exemple « Énoncer le théorème de diagonalisation des matrices symétriques réelles », etc...

type 4 : Il arrive aussi parfois qu'on pose des questions de la forme « Que pensez-vous de... » ou « Quelle conclusion peut-on tirer de ... ». Ce genre de question vient généralement en fin d'exercice, et le but visé n'est pas de faire produire une démonstration, mais plutôt l'énoncé d'un théorème dont la démonstration découle plus ou moins trivialement des questions précédentes. Il y a là un coté un peu subjectif, car il y a peut-être plusieurs conclusions à tirer des questions qui précèdent. Mais le plus souvent le contexte montre bien ce qui est attendu.([3])

type 5 : Enfin, très exceptionnellement, on peut poser une question comme « Trouver la faute dans le raisonnement suivant. ... », c'est-à-dire qu'on inverse les rôles de l'étudiant et du correcteur.

2 Que veut dire « corriger » un exercice ?

type 1 : Il s'agit juste de pouvoir vérifier qu'un texte qui prétend être une démonstration de l'énoncé de la question en est effectivement une. Bien entendu, tout professeur de mathématiques est habitué à ce travail (rarement enthousiasmant d'ailleurs), mais il le fait le plus souvent avec un certain laxisme, en ce sens qu'il accorde facilement des points à des ébauches de démonstrations qui ne sont pas des démonstrations acceptables sur le plan formel (par exemple, qui ne seraient pas acceptables dans un livre de mathématiques). C'est ici un point important pour ce qui nous concerne, c'est-à-dire la question de la correction « automatique » des exercices. Un algorithme de correction n'aura pas la même souplesse qu'un professeur, en particulier concernant des détails sans grande importance, comme les erreurs de syntaxe, qui risquent pourtant d'être rédibitoires pour la suite de la correction, c'est-à-dire la partie qu'on peut qualifier de « sémantique ». Il est donc probablement nécessaire, pour rendre la correction des exercices de mathématiques automatique, d'utiliser coté étudiant un mode de saisie interactif indiquant les fautes de syntaxes en temps réel, la « correction » proprement-dite (et l'attribution des points) ne portant alors plus que sur la sémantique.([4]) Nous discutons plus loin cette question de la correction « sémantique ».

type 2 : Il s'agit juste de vérifier que l'expression donnée par l'étudiant est « trivialement égale » à la réponse proposée par le correcteur, du moins dans le cas où une telle réponse ne nécessite pas de démonstration ni de calcul. Si au contraire une justification est requise, il s'agit de vérifier une preuve de l'énoncé $E[a]$ dont il est question dans la section précédente.

type 3 : La correction d'une question de cours ne doit pas être une simple comparaison syntaxique (même un peu perfectionnée). Il est nécessaire d'être plus souple que cela. Il semble que le bon compromis (celui qui évitera les injustices qu'une méthode purement syntaxique produira nécessairement) est que l'énoncé donné par l'étudiant soit « trivialement équivalent » (dans le cas où la question porte sur un théorème) à celui qui est attendu (c'est-à-dire à la solution fournie par le concepteur de la question). Il est donc nécessaire de disposer d'un algorithme capable de démontrer automatiquement des énoncés censés

3. J'ai souvent posé la question très courte : « Conclure. », comme dernière question d'un exercice dont l'ensemble des questions constitue un raisonnement par l'absurde. Il est rare que les étudiants répondent correctement à cette question, car ils ne se rendent pas compte que dans le préambule de l'exercice on a fait une hypothèse, et que le but de l'exercice était d'en prouver la négation.

4. On peut aussi imaginer un algorithme tentant de réconstituer une syntaxe correcte à partir d'une syntaxe incorrecte. C'est une autre approche qui n'est peut-être pas complètement irréaliste.

être « évidents » (ici l'équivalence des deux énoncés à comparer). Un tel algorithme est d'ailleurs aussi nécessaire pour les exercices de type 1. Nous discutons d'un tel algorithme plus loin. Dans le cas d'une question de cours portant sur une définition, il s'agit de vérifier l'« équivalence » des deux définitions. Ce n'est pas le même type d'équivalence que l'équivalence (logique) entre énoncés. Il peut par exemple s'agir d'une structure définie par des opérations et des axiomes. Il faut donc vérifier la présence de ces opérations (et d'elles seules) en faisant preuve d'une certains souplesse dans leurs appellations, puis vérifier que les axiomes sont équivalents aux axiomes attendus. En réalité la situation est moins simple, car par exemple dans le cas de la structure de groupe, on peut introduire l'inverse soit comme une opération, soit en l'intégrant dans un axiome à l'aide d'un quantificateur existentiel. Nous n'avons pas pour le moment de réponse précise à apporter sur ces difficultés.

type 4 : L'énoncé fourni par l'étudiant doit de toute façon résulter de manière triviale des question précédentes. C'est donc encore ce même algorithme de démonstration automatique des énoncés « évidents » qui intervient. Reste quand-même que l'énoncé fourni par l'étudiant n'est peut-être pas trivialement équivalent à celui qui était attendu. On peut donc réutiliser notre algorithme pour le vérifier. La seule chose qui reste à faire est de savoir comment noter la question en fonction de ces résultats. Il semble que des points doivent être alloués si l'énoncé proposé résulte effectivement trivialement des questions précédentes (encore que le nombre de ces points devrait pouvoir dépendre de l'« intérêt » de ce qui est proposé par l'étudiant, et cela est clairement plus difficile à déterminer), et que des points supplémentaires doivent être accordés si l'énoncé proposé est trivialement équivalent à celui qui était attendu. En fait, ce calcul de points dépend aussi clairement des indications (aides) données aux étudiants en marge de l'exercice.

type 5 : Le problème ici est de savoir sous quelle forme l'étudiant peut indiquer la faute commise. On a toujours la solution du QCM([5]). On propose une série de fautes possibles et l'étudiant doit cocher la bonne réponse. Une autre solution raisonnable, en supposant que l'énoncé concerné soit démontrable, consiste à lui faire produire une preuve correcte du même énoncé (qui bien entendu a été donné explicitement dans le préambule de la question). Mais il peut aussi arriver que l'énoncé ne soit pas démontrable. Il m'est en effet arrivé d'écrire : « Trouver la faute dans le raisonnement suivant, dont la conclusion est clairement fausse ». Dans ce dernier cas, il semble que le mieux qu'on puisse faire soit le QCM, éventuellement accompagné d'une demande de justification, c'est-à-dire d'une preuve de la négation de l'affirmation incriminée par l'étudiant dans la démonstration proposée.

En conclusion, certaines choses sont clairement faisable à l'aide des algorithmes indiqués, et quelques cas particulier sont plus problématiques. Il est probable qu'on se passera de ces cas particulier la plupart du temps. Ils nécessitent sans doute une programmation spécifique au cas par cas, qui ne sera pas toujours justifiée.

3 Comment vérifier une démonstration ?

Bourbaki écrivait dans le premier tome de son célèbre traité que la vérification d'une démonstration est un processus « en quelque sorte mécanique ». On sait aujourd'hui (et on se doute d'ailleurs facilement si on est un peu habitué à vérifier des preuves mathématiques) que la vérification du fait qu'un texte donné est bien une démonstration d'un énoncé donné est non seulement algorithmique mais aussi de temps d'exécution essentiellement linéaire par rapport à la taille de ce texte.

Au contraire, la recherche d'une preuve pour un énoncé donné n'est que semi-algorithmique en ce sens que le mieux qu'on puisse faire est de produire un algorithme (au moins théorique, c'est-à-dire inutilisable en pratique) qui trouvera une preuve si une telle preuve existe et qui tournera indéfiniment si elle n'existe pas, ne nous permettant donc pas dans ce cas de savoir si l'énoncé est démontrable ou non. En fait, la situation est plus mauvaise que cela, car le semi-algorithme en question est très vite victime d'explosion

5. Questionnaire à choix multiples.

combinatoire dans la plupart des cas. En pratique, la recherche de preuve doit donc utiliser des algorithmes « bridés », c'est-à-dire renonçant systématiquement à explorer nombre de possibilités. Ainsi, un algorithme efficace en matière de recherche de preuve peut très bien échouer sur un énoncé pourtant démontrable. Malgré tout, ce genre de « bridage » définit de fait des classes récursives d'énoncés démontrables (ceux qui sont démontrables par l'algorithme en question) pourvu que le bridage soit fait de telle façon que l'algorithme termine toujours, et permet donc en particulier, dans le cas d'un bridage fort, de définir le sens de l'expression « énoncé évident ». Nous discutons cela plus loin.

Vérifier qu'un texte donné est bien une preuve d'un énoncé donné ne doit donc pas être très compliqué, et de fait il s'agit juste d'un cas particulier d'un processus bien plus général maintenant bien connu des concepteurs de compilateurs de certains langages de programmation modernes, à savoir l'« inférence de type ». Pour faire simple, disons que l'inférence de type est plus ou moins la même chose que l'assemblage d'un puzzle dont les pièces seraient paramétrées (des « schémas de pièces » en fait). On le comprendra bien sur un exemple. Imaginons qu'on ait plusieurs fonctions $f : X \to Y$, $g : U \to V$, ... et plusieurs objets mathématiques $a \in Y$, $b \in X$, $c \in U$, ... A priori, le « terme » $f(b)$ est correctement construit car b appartient à l'ensemble de départ de f. Mais le terme $f(a)$ est mal construit si les ensembles X et Y sont (disons) disjoints. On voit que les deux « pièces » f et b s'assemblent correctement comme les pièces d'un puzzle, alors que f et a ne s'assemblent pas correctement (les bords des pièces n'ont pas des formes compatibles). On a affaire à des schémas de pièces quand ces pièces dépendent de types inconnus qu'il faut déterminer, ce qui est réalisé par « unification », mécanisme extrêmement efficace, qui donne même parfois l'impression que l'inférence de type est un processus « intelligent ».

Pour que la vérification des preuves ne soit qu'un cas particulier de ce processus, il suffit que le système de typage du langage formel utilisé ait (en plus des types nécessaires pour la représentation des objets mathématiques) d'une part un « type des énoncés », et d'autre part pour chaque énoncé un type des preuves de cet énoncé. Si E est un énoncé, et en notant $W(E)$ le type de ses preuves, la vérification du fait qu'un texte p est bien une preuve de E se résume à vérifier que p est un terme bien formé de type $W(E)$. Il existe quelques systèmes, appelés « assistants de preuve », ayant ces caractéristiques (pour l'essentiel). La plupart d'entre eux sont basés sur la théorie des types de Martin-Löf ou sur des théorie similaires comme celle des constructions inductives. Toutefois, tous ces systèmes ont un gros défaut pour l'utilisation qu'on veut en faire ici. Le plus connu d'entre eux, le système COQ (de l'INRIA) qui existe depuis plusieurs décennies n'a eu aucun succès auprès des mathématiciens. C'est dû à mon avis au caractère ésotérique aussi bien de la syntaxe que de la sémantique de ce système (ça fait peur et ça ne ressemble pas du tout à des maths !).

4 Comment formaliser les mathématiques « usuelles » ?

Tout en étant cruciale pour la question de la correction automatique des exercices des mathématiques, cette question dépasse très largement le cadre de cet exposé, et je me contenterai donc de quelques remarques. On pourra consulter [3] pour des informations plus détaillées.

Comme signalé ci-dessus, la plupart des assistants de preuve actuels ne formalisent pas les mathématiques usuelles mais des mathématiques qu'on peut qualifier d'« exotiques »,([6]) et c'est là bien évidemment la cause de leur échec auprès des mathématiciens. En particulier, ces systèmes parlent de « types » alors que les mathématiciens ne parlent jamais de « types ». Les mathématiciens parlent d'« ensembles ». Ceci dit, ils sont pour la plupart persuadés que la fondation de leur discipline est la théorie des ensembles de Zermelo-Frænkel (ZF), une théorie absolument non typée puisqu'elle veut que tout objet mathématique soit un ensemble. On est dans une situation assez paradoxale, car les mathématiques sont clairement fortement typées, et ce typage est absolument nécessaire pour faire des mathématiques de la manière usuelle. Par exemple, un étudiant en algèbre linéaire qui écrit une égalité entre un scalaire et un vecteur

6. Et même « ésotériques » selon Yves Lafont.

commet ce que tout professeur de mathématiques considérera comme une faute. Or, si l'on en croit ZF, ce scalaire et ce vecteur sont deux ensembles, et deux ensembles sont égaux s'ils ont les mêmes élements. Ainsi, d'après ZF, une égalité entre un scalaire et un vecteur a bien un sens. Certains mathématiciens, qui ont été amenés à réfléchir à cette question, se sont rendu compte de diverses « absurdités » auquelles mène ZF. Par exemple, on trouve dans le célèbre « Cours d'Algèbre » de Roger Godement la remarque que l'égalité (entre produits cartésiens d'ensembles) $X \times (Y \times Z) = (X \times Y) \times Z$ est fausse selon ZF, assortie de l'affirmation qu'en pratique on la considère comme exacte. On pourrait multiplier les exemples.

Nous avons donc d'un coté des techniques informatiques maintenant parfaitement maîtrisées de vérification de preuve basées sur le typage, et de l'autre coté des mathématiciens pour lesquels le mot « type » n'a pas de sens, du moins pas de sens conscient. En lréalité, les mathématiciens typent fortement les concepts qu'ils définissent et ils vont même jusqu'à « cloner » des types de données. C'est ce qu'ils font par exemple quand ils donnent le nom de « suite » et le nom de « série » à des concepts qu'ils définissent comme des fonctions de $\mathbb{N}$ vers $\mathbb{R}$. Nous sommes ici en présence de trois concepts clairement distincts dans l'esprit du matheux, à savoir ceux de fonction, suite et série, distincts parce que ne se manipulant pas de la même manière (par exemple, pour ce qui est de la multiplication) et pourtant définis de la même manière. Ce dont il faut se rendre compte ici est que le typage en mathématiques est un épiphénomène du fait de donner un nouveau nom (par exemple, « suite » ou « série ») à des objets qu'on connait déjà sous un autre nom (par exemple, « fonction de $\mathbb{N}$ vers $\mathbb{R}$ »), et que ce typage est essentiel à une bonne compréhension des mathématiques, et bien entendu à leur vérification automatique.

Il existe diverses théories candidates à la formalisation des mathématiques. Pour des raisons diverses (voir par exemple [3]), les théories à la Martin-Löf sont facilement mises hors jeu, de même que ZF. Que nous reste-t-il alors ? Eh bien, il reste une théorie dont certains savent depuis les années 80 (voir par exemple [2], [4] et surtout [1]) qu'elle est le bon choix pour bien formaliser les mathématiques usuelles. Elle est capable de concilier la question du typage et les questions concernant les ensembles. Il s'agit de la théorie des topos.([7]) C'est sur cette théorie qu'est fondé le projet Saunders, dont je préconise l'utilisation pour la correction automatique des exercices de mathématiques. On aura une idée assez détaillée de ce projet en consultant [5].

5 Qu'est-ce qu'un énoncé « évident » ?

Comme on l'a vu, la notion d'« énoncé évident » joue un rôle central dans la question de la vérification des exercices de mathématiques. En ce qui concerne un système de formalisation des mathématiques, par exemple un MOOC avec correction automatique des exercices, la notion d'« énoncé évident » est définie par le fait qu'un certain algorithme bien bridé de recherche de preuve parvient à en trouver une preuve. Voici comment on peut brider un algorithme générique (et combinatoirement explosif) de recherche de preuve.

Comme on l'a dit plus haut, rechercher une preuve d'un énoncé E revient à trouver un terme du type qu'on a appelé $W(E)$. Toutefois, il s'agit d'un processus récursif, et la recherche d'un terme d'un certain type peut amener à rechercher des termes d'autres types et en particulier de types qui ne sont pas de la forme $W(E)$. L'algorithme en question doit donc traiter tous les types du système. Essentiellement, dans le langage formel (appelé « langage W ») qui constitue les « intérieurs » du système Saunders, les types sont les suivants : des types récursifs définis par des grammaires (comme le type des entiers naturels), des produits dépendants, le type des parties d'un type (dont les termes représentent des « ensembles »), un type Ω des énoncés,([8]) et pour chaque énoncé E un type $W(E)$ des preuves de cet énoncé.([9]) Les

7. Il s'avère, comme cela est montré dans [1], que vue sous cet angle, la théorie des topos n'est rien d'autre qu'une formalisation très précise de la théorie « naïve » des ensembles, théorie qui est reconnue, même par les théoriciens de ZF, comme le véritable fondement des mathématiques usuelles.

8. Il s'agit bien sûr du classifiant du foncteur des sous-objets de la théorie des topos.

9. Il n'y a pas de types fonctionnels et pas de sous-types, mais on peut définir des ensembles fonctionnels et des sous-

types de la forme $W(E)$ ont cette particularité que deux termes de ce type sont toujours égaux.([10]) Ceci veut dire que quand on cherche une preuve d'un énoncé, peu importe celle qu'on trouve pourvu qu'on en trouve une. On applique donc dans le cas de $W(E)$ des techniques bien connues et systématiques de recherche de preuve ne nécessitant pas de retour en arrière (backtrack), mais qu'on ne détaillera pas ici.

Au contraire, si on doit rechercher un élément dans $\mathbb{N}$ par exemple pour prouver un énoncé de la forme $\exists_{n\in\mathbb{N}}\ E$, il faut trouver le bon n qui permettra de prouver E. Comme $\mathbb{N}$ a une infinité d'éléments, on voit que le problème n'est pas simple (et on voit même clairement ici que le fait que $\mathbb{N}$ soit infini entraîne que ce problème est semi-algorithmique). Une façon de le rendre simple (et algorithmique, c'est-à-dire d'obtenir un algorithme qui termine toujours) est de n'essayer pour remplacer n que les expressions du bon type (i.e. représentant des entiers naturels dans le cas de l'exemple) qui sont déjà connues dans le contexte courant et même seulement dans le contexte « local ». Il se peut qu'il y en ait plusieurs, mais cela limite considérablement les possibilités de recherche. D'une certaine façon, ceci revient à chercher sans faire preuve d'imagination puisqu'on n'essaye que ce qu'on a sous la main. Cette méthode conduit à une notion d'« énoncé évident » qui est précisément celle du système Saunders.

Cette notion est d'ailleurs assez large puisqu'il s'avère que (à supposer qu'on fouille aussi le contexte global) pratiquement tous les exercices qu'on pose dans les problèmes d'examen sont évidents au sens de cet algorithme, ce qui impose bien sûr de le brider fortement pour la correction d'exercices, en n'essayant que les objets trouvés dans le contexte local, et peut-être même seulement le plus récent d'entre eux. Ceci doit bien sûr être paramétrable en fonction du niveau exigé des étudiants. Notons pour terminer que la culture mathématique qu'on essaye de faire acquérir aux étudiants est pour une très grande part une bonne connaissance du contexte global (constitué de toutes les définitions et théorème établis jusque là). Ainsi, en renonçant à fouiller le contexte global lors de la correction automatique, on teste effectivement le degré d'assimilation du contexte global par les étudiants, et c'est précisément là le rôle des examens.

Références

[1] **J. L. Bell** *Toposes and Local Set Theories. An introduction.* Dover 2008.

[2] **J. Lambek, P. J. Scott** : *Introduction to higher order categorical logic.* Cambridge University Press, Cambridge 1986.

[3] **A. Prouté.** *Sur quelques liens entre théorie des topos et théorie de la démonstration.* `http://www.logique.jussieu.fr/~alp/luminy_05_2007.pdf`

[4] **A. Prouté** *Introduction à la logique catégorique.* `http://www.logique.jussieu.fr/~alp/cours_2010.pdf`

[5] **The Saunders Team.** *Getting started with the writing of mathematics in the Saunders syntax.* `http://www.logique.jussieu.fr/~alp/Getting-started.pdf`

ensembles (de même que des ensembles quotients).

10. Ce qui n'est pas le cas dans les théories à la Martin-Löf, ce qui a pour conséquence que dans ces théories il existe des inclusions canoniques dont l'injectivité n'est pas démontrable, ce qui est évidemment inacceptable par les matheux. Le problème a été corrigé en COQ relativement récemment par l'introduction facultative d'un principe d'indiscernabilité des preuves. Toutefois, demander à un mathématicien s'il accepte ou non un tel axiome est comme lui parler dans une langue qu'il ne connait pas.