Création d'un générateur d'expressions algébriques

Arnaud Libelin, Aurélien Morel, Marie Lefevre, Stéphanie Jean-Daubias

Université de Lyon, CNRS

Université Lyon 1, LIRIS, UMR5205, F-69622, France

Marie.Lefevre@liris.univ-lyon1.fr, Stephanie.Jean-Daubias@liris.univ-lyon1.fr

Résumé : Ce document présente une recherche sur la conception d'un générateur d'exercices portant sur les expressions algébriques. La conception a porté sur l'identification des exercices pouvant être générés et sur l'interface permettant aux enseignants de contraindre la génération des exercices. Le rapport présente les résultats obtenus à la suite de cette recherche et l'intégration qui en a été faite dans le module ADAPTE de l'environnement EPROFILEA.

Mots-clés : générateurs aléatoires, interface, ergonomie

1. Présentation du travail de recherche

1.1. Le projet PERLEA et son environnement

Le projet PERLEA (Profils d'Elèves Réutilisés pour L'Enseignant et l'Apprenant) ainsi que son environnement EPROFILEA (environnement d'Exploitation de PROFILs par les Enseignants et les Apprenants) vise à gérer des profils d'apprenants hétérogènes et à proposer des exploitations de ces profils [JEAN-DAUBIAS et al. 2009]. L'architecture de cet environnement est donnée en figure 1 de l'Annexe A.

1.2. Le module ADAPTE

Le module ADAPTE permet de fournir à chaque apprenant des activités adaptées à son profil tout en respectant les choix pédagogiques de son enseignant. Grâce à ces activités, il pourra acquérir de nouvelles compétences ou corriger des problèmes dans un domaine particulier.

Notre action s'est plus particulièrement effectuée au sein de la partie d'ADAPTE qui a pour but de générer des exercices personnalisés pour chaque apprenant en fonction du choix pédagogique de l'enseignant. Le module contient plusieurs générateurs qui génèrent chacun un type d'exercice bien particulier allant de la conjugaison de verbes jusqu'à l'élaboration de table de multiplication en passant par la génération d'expressions algébriques qui était le sujet de ce travail.

1.3. Le générateur d'expressions algébriques

Avant de nous avancer plus dans les explications du générateur, il convient de définir et d'expliquer ce qu'est concrètement une expression algébrique :

> ***On appelle expression algébrique, un ensemble de lettres et de nombres reliés entre eux par des signes indiquant les opérations à effectuer.***

On comprend bien là qu'une expression algébrique est une façon d'écrire un calcul avec des variables qui sont inconnues à l'utilisateur en utilisant des opérateurs de base (addition, soustraction…). Ces paramètres inconnus sont représentés par des lettres [DAN].

Notre générateur à intégrer dans le module ADAPTE doit au final générer un énoncé d'exercice (sur les expressions algébriques), ainsi que sa solution, correspondant aux souhaits de l'enseignant, exprimés par des contraintes sur les exercices. Ce générateur vient donc s'intégrer directement dans le module et vient compléter la variété d'exercices proposés. Nous avions donc pour but de créer un générateur et résolveur complet pour des expressions algébriques de type : « Développer », « Factoriser », « Équation », « Inéquation » et « Système ». Le but étant de faire en sorte que des contraintes de génération puissent être imposées à chaque type d'exercices que l'enseignant souhaite générer.

Le but était de concevoir l'ensemble des générateurs définis ci-dessus et au final la tâche à accomplir étant importante, nous nous sommes limités à une partie de l'interface dans son intégralité (notre choix s'est porté sur développer), de mener notre recherche complète sur celle-ci et de générer une interface totalement fonctionnelle, ainsi qu'une génération d'exercice et de sa solution. Si cette partie fonctionnait à la fin du projet, les autres interfaces nécessiteraient juste une adaptation de cette version, car toutes les interfaces suivaient plus ou moins le même schéma de contraintes. Le générateur était donc l'aspect de notre travail qui nécessiterait le plus d'approfondissement et de recherche.

2. État de l'art

Dans le cadre de l'élaboration d'exercices sur les expressions algébriques et leur corrigé, il est nécessaire d'utiliser des générateurs permettant à partir de contraintes de définir des énoncés aléatoires, si possible « ouverts ». Les générateurs permettent de générer des énoncés et les résolveurs pour les résoudre le plus souvent avec un système logique comme le PROLOG. Des logiciels de calcul formel peuvent également être utilisés dans le cas des mathématiques, car ils permettent de transformer des expressions sous une autre forme et donc de les résoudre. Il existe plusieurs générateurs et résolveurs dans les domaines de la recherche et du libre qui doivent être étudiés pour voir si ceux-ci sont compatibles avec les besoins d'ADAPTE et dans le cas contraire, voir si les aspects positifs de certains peuvent être réutilisés. Ceux que nous avons trouvés représentent probablement une liste non exhaustive de ceux qui peuvent exister, mais elle contient cependant les plus adéquates pour ADAPTE.

2.1. Les Générateurs-Résolveurs

2.1.1. APLUSIX

APLUSIX est un EIAH d'apprentissage de l'algèbre. Il utilise des patrons d'exercices pour proposer des types d'exercices correspondant aux multiples domaines de l'algèbre. À partir de ces patrons, il permet de générer des exercices comportant une partie aléatoire et d'en créer un corrigé [CHAACHOUA, 04]. Ce résolveur a l'avantage de regrouper les expressions algébriques sous forme de patrons génériques pour ensuite créer des exercices aléatoires. L'enseignant peut effectivement choisir le type d'expression algébrique aléatoire qu'il souhaite générer, mais il ne peut pas appliquer un ensemble de contraintes sur son expression algébrique pour la restreindre et symboliser ses attentes.

2.1.2. CAMELIA et ÉLISE

CAMELIA (Calcul Algébrique, Mathématiques Elémentaire et Intelligence Artificielle) est un résolveur de problèmes de calcul algébrique conçu pour établir des preuves et conduire des calculs symboliques en faisant appel à des algorithmes et à des heuristiques [GRANDBASTIEN, 01]. Il utilise des plans de résolution et des métarègles (qui sont des règles régissant d'autres règles, elles permettent de choisir entre l'application de telle ou telle règle face à une certaine situation.). Ces métarègles permettent d'éliminer les plans de résolution inutiles et de classer ceux restants avec une note fonction de l'ordre d'intérêt et du coût d'exécution. Ce résolveur a pour intérêt de détailler toutes les étapes intermédiaires avant de parvenir au résultat. D'une part, ce n'est pas nécessaire dans le cadre de notre travail (nous souhaitons juste obtenir l'état final) et d'autre part, réhabiliter CAMELIA pourrait être un travail long dans le sens où celui-ci a été développé sous MAC avec un Apple II. Il faudrait donc se servir du code de base pour le réécrire entièrement en Delphi.

ÉLISE a été élaboré à partir du résolveur CAMELIA. Le public ciblé par ce résolveur est avant tout l'apprenant [DELOZANNE, 92]. Dans notre cas, c'est plutôt celui de l'enseignant que nous souhaitons aborder. Élise permet d'agir sur l'aspect résolution et guidage de l'apprenant, or cet aspect ne nous intéresse pas. Nous ne nous focalisons pas sur la partie explication, mais sur la partie résolution. De plus, Élise n'est pas générique et ne génère que certains types d'expressions.

2.1.3. SYRCLAD

SYRCLAD est quant à lui une architecture de résolveurs de problèmes qui permet d'expliciter les connaissances de classification, de reformulation et de résolution [GUIN, 97]. Il est développé par Nathalie Guin et permet une grande adaptabilité. Les paramètres en entrée qui représentent les données du problème à résoudre sont à réécrire en PROLOG : cela revient à décomposer le problème de façon logique

puis à le reformuler sous une forme compréhensible par PROLOG. Dans le cadre de ce travail, SYRCLAD pourrait s'avérer utile couplé avec les patrons d'APLUSIX, mais long à mettre en place. De plus, SYRCLAD ne donne pas de résultat à proprement parler, mais fournit plus un plan pour parvenir au résultat.

2.2. Les logiciels de calcul formel

Un logiciel de calcul formel est un logiciel qui facilite le calcul symbolique (calcul sous forme mathématique). Il permet la manipulation des expressions mathématiques sous leur forme symbolique.

2.2.1. REDUCE

C'est un logiciel de calcul formel utilisé par CAMELIA. Il permet de faire le calcul de chaque étape intermédiaire lorsque CAMELIA les considère comme les étapes suivantes du problème [GRANDBASTIEN, 01]. Il permet donc de créer le fil de la résolution. Dans notre cas, ces étapes ne sont pas nécessaires, car nous souhaitons juste obtenir le résultat de l'expression de départ.

2.2.2. XCAS/GIAC

GIAC est un moteur de calcul formel permettant de résoudre tout type d'expressions mathématiques qui est utilisé par le logiciel XCAS [Parisse, 07]. Il est développé en C++ et dispose d'un mode de compatibilité pour les formats Maple et MuPAD. GIAC est capable de simplifier, résoudre des équations, dériver, intégrer, décomposer des éléments en fractions rationnelles. Il contient donc toutes les spécificités nécessaires à notre application et pourrait servir à des intégrations futures dans EPROFILEA, car il permet de traiter des cas dans le domaine de la géométrie également.

2.3. SYNTHÈSE

Les générateurs existants répondent tous aux attentes initiales, mais ne permettent pas de résoudre de façon semi-automatique les problèmes. Ils sont tous orientés vers la génération automatique (donc sans contraintes). Dans notre cas, l'enseignant doit pouvoir interagir avec l'environnement et non pas être spectateur de la génération. En plus de générer des exercices, ils permettent à l'apprenant d'avoir une correction de son raisonnement étape par étape, alors que notre seul intérêt est la génération d'une solution sans tenir compte des étapes que pourrait suivre l'élève. Sachant que les étapes qu'ils fournissent seraient très probablement inadaptées au contexte pédagogique qui est le nôtre, nous avons donc décidé de programmer nous-mêmes la partie génération des expressions tout en utilisant l'aspect des patrons définis par APLUSIX. L'expression générée devant ensuite être résolue, nous avons choisi un logiciel de calcul formel pour résoudre les expressions algébriques générées. Notre choix s'est porté sur GIAC du fait de sa flexibilité, de son adaptabilité et de sa résolution rapide des calculs. Les fonctions qu'il implémente répondent à toutes les attentes nécessaires concernant notre générateur de problèmes du module d'ADAPTE.

3. Les recherches sur la génération des expressions algébriques

Nous allons définir ce que sont des expressions algébriques et les membres dont elles sont composées pour pouvoir par la suite comprendre les patrons que nous avons dû utiliser et les contraintes qu'il est possible d'appliquer dessus.

3.1 Les expressions algébriques

Une expression algébrique est constituée (le plus souvent) d'un ou plusieurs **monômes**. Ces monômes possèdent un facteur numérique (le **coefficient)** et un facteur littéral (la **partie littérale)**. Ainsi, dans le monôme **$3a^2b$, 3** est le coefficient et **a^2b** est la partie littérale.

Un ensemble de monômes reliés entre eux par des signes d'addition et de soustraction est un **polynôme**. Les polynômes les plus courants sont les binômes (ensemble de 2 monômes) et les trinômes (ensemble de 3 monômes).

Ainsi, $\mathbf{3x^3 + 5x^2y + 2xy^2 + 2x^3 - 4x^2y + 2xy^2 = 5x^3 + x^2y + 4xy^2}$ est un polynôme (la partie droite représente le même polynôme que la gauche, mais d'une façon simplifiée)

Les **produits remarquables** sont des façons rapides et efficaces de simplifier des expressions algébriques possédant une forme particulière. Ces expressions peuvent permettre à un enseignant de créer des exercices typiques pour voir si l'apprenant a bien assimilé cette méthode. Les produits remarquables les plus utilisés sont les suivants :

- Carré de la somme de 2 nombres : $\mathbf{(a + b)^2 = (a + b)\ (a + b) = a^2 + 2ab + b^2}$
- Carré de la différence de 2 nombres : $\mathbf{(a - b)^2 = (a - b)(a – b) = a^2 - 2ab + b^2}$
- Produit de la somme de 2 nombres par leur différence : $\mathbf{(a + b)(a – b) = a^2 - b^2}$
- Cube de la somme de 2 nombres : $\mathbf{(a + b)^3 = a^3 + 3a^2b + 3ab^2 + b^3}$
- Cube de la différence de 2 nombres : $\mathbf{(a - b)^3 = a^3 - 3a^2b + 3ab^2 + b^3}$
- Formule générale du produit remarquable : $(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k}b^k$

Pour générer des expressions algébriques telles que définies ci-dessus, nous avons été amenés à trouver une solution simple pour permettre à l'enseignant de choisir parmi toutes ces possibilités de façon la plus générique possible pour qu'il puisse créer le plus d'expressions possible.

3.2. Les patrons génériques

Pour cette partie étant donné le nombre de possibilités d'expressions algébriques existantes, nous avons utilisé les patrons d'expressions algébriques définis dans APLUSIX. Ces patrons étant développés pour l'environnement APLUSIX, nous avons dû pratiquer une rétroconception du code de leur patron pour les intégrer dans ADAPTE. Il a donc fallu les étudier de façon à en dégager des patrons globaux (patrons qui peuvent regrouper des expressions de même forme). APLUSIX contient dans ses fichiers les patrons génériques en dessous desquels ils définissent les expressions pouvant en découler. Ainsi, leurs fichiers de patrons contiennent de nombreuses parties comme ci-dessous (cf Annexe B page 13 pour un exemple plus complet) :

```
{[nom PatronExpADev21]
 [sorteDe PatronExpADev2]
 [patron <<(ax+b)-(dx+e)>>]
 [domaine ((a entier* petit) (b entier* petit) (d entier* petit) (e entier* petit))]
 [frequence 2]}
```

Figure 1 – Un patron générique de APLUSIX

Le schéma mis en place par les créateurs d'APLUSIX est plutôt clair pour les variables représentant une expression donnée. On connait le « patron père » dont découle cette expression et ce qui a été fait avant pour y arriver. Cela leur permet de faire une résolution étape par étape en les utilisant. Nous avions uniquement besoin des types génériques nous concernant, car les expressions en découlant sont chez nous générées aléatoirement grâce aux contraintes de l'enseignant. Tous les patrons qui ont été extraits des fichiers de APLUSIX sont présents en annexe C page 14 à 15.

3.3. Les contraintes

Nous avons été amenés à réfléchir aux contraintes pouvant être utiles à l'enseignant pour définir ses besoins et ses attentes vis-à-vis de ses expressions algébriques à générer. Pour cela, nous avons regardé ce qui existait déjà dans les interfaces des autres générateurs d'ADAPTE pour nous aider à définir celles qui pourraient avoir également du sens dans l'interface de notre générateur. Nous avons également dû mettre en place nos propres contraintes. Un aperçu complet de ces contraintes est disponible en Annexe C. Par défaut, si l'utilisateur décide de ne sélectionner aucune valeur, c'est une valeur aléatoire qui sera appliquée pour chaque champ de contraintes présent dans l'interface.

La première partie que nous traitons est celle de l'inconnue qui compose le monôme (figure 2 – 1). Il est possible de laisser l'utilisateur choisir s'il souhaite introduire une inconnue dans son monôme ou non, par défaut la valeur est à « Valeur numérique », ce qui signifie que l'utilisateur n'insère pas d'inconnue pour ce monôme. Il peut choisir la lettre (par défaut seul x, y et z sont disponibles), le choix « Autre… » lui permet de saisir une autre lettre. Il peut également choisir le degré à appliquer sur cette inconnue.

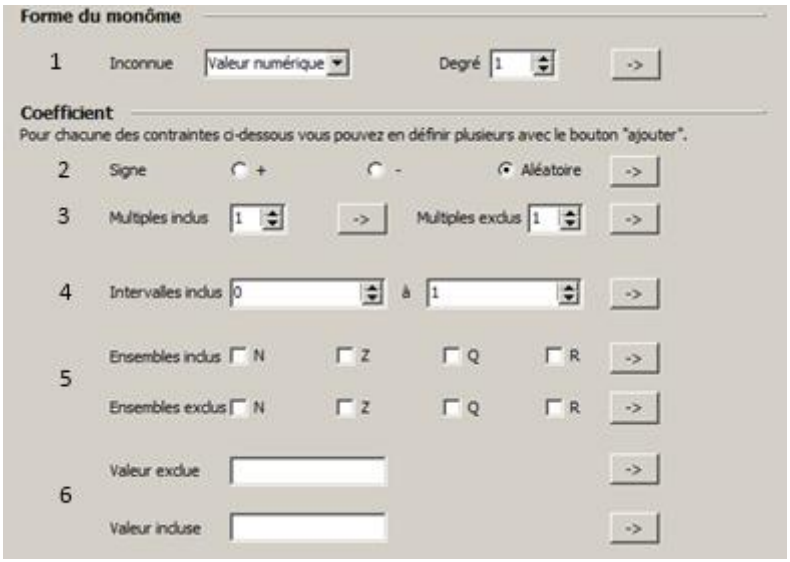


Figure 2 – Contraintes sur un monôme

La seconde partie est celle qui concerne le coefficient du monôme, nous nous sommes rendu compte que le signe de chaque élément de l'expression à définir était important. Le choix des signes permet de contraindre le signe du monôme (Figure 2 - 2). L'utilisateur a par défaut un choix de signe aléatoire, s'il souhaite en changer il lui suffit de choisir entre le « + » et le « - ».

Dans les autres possibilités de contraintes, nous avons défini les multiples (Figure 2 – 3). Cela permet de choisir de générer un monôme qui contiendra comme coefficient un nombre qui est un multiple de celui choisi (multiple inclus). Il peut également choisir de générer un nombre qui ne doit pas être un multiple d'un autre (multiple exclu).

Si l'enseignant ne souhaite pas un ensemble de valeurs, mais une zone de génération plus ciblée pour un coefficient, nous proposons des intervalles de valeurs qui permettent à l'utilisateur de choisir une valeur aléatoire non pas sur l'infini, mais d'une borne fixe à une autre (Figure 2 – 4).

Les valeurs des coefficients ont dû être mises en place de façon à guider les choix de l'utilisateur sans le bloquer. Au départ, nous avions prévu de lui attribuer des plages de valeurs parmi lesquelles ces coefficients seraient définis, mais nous nous sommes rendu compte que cela était beaucoup trop restrictif

comme choix. De plus concernant des valeurs plus complexes comme les racines carrées, cette solution ne serait pas viable. Notre recherche s'est donc focalisée sur le choix le plus large possible et donc sur les ensembles de valeurs, nous avons donc utilisé les ensembles mathématiques connus (Figure 2 – 5) pour lui permettre de créer ces choix ($\mathbb{N}$,$\mathbb{Z}$,$\mathbb{Q}$,$\mathbb{R}$). Nous avons décomposé nos contraintes en 2 champs pour représenter les ensembles à inclure et les ensembles à exclure. L'enseignant peut cocher les ensembles qu'il souhaite inclure et en complément, les ensembles du choix opposé sont grisés. Cela évite à l'utilisateur d'inclure et d'exclure en même temps, car cela représenterait un non-sens et ne générerait pas de solution.

Pour finir sur les coefficients, si l'enseignant souhaite saisir lui-même la valeur que doit avoir son coefficient, il peut le faire et cette action prime sur toutes celles qui auraient pu être définies juste auparavant. Il a à sa disposition 2 champs (inclus et exclu). Dans la partie vide (ci-dessous) apparaît la formule saisie par l'utilisateur dans le champ central (Figure 2 - 6).

4. Intégration dans ADAPTE

Nous développons ici ce que nous avons pu mettre en place suite à notre recherche. On présente ce dont nous sommes en charge pour le module que nous nous sommes vus attribuer.

4.1. Modification d'ADAPTE pour l'intégration du générateur d'expressions algébriques

Nous avons développé pour cette intégration le générateur aléatoire d'exercices avec prise en compte des contraintes. Sur l'interface de génération des feuilles d'exercices, il nous a fallu intégrer et donc développer une partie de la création de la solution. Pour le module de contraintes, nous avons dû nous inspirer de ceux développés précédemment dans ADAPTE, ce qui nous a permis de faire une intégration plutôt rapide de notre code dans celui existant. Quant au module concernant la feuille d'exercice, cela n'a pas été aussi simple, car il nous a fallu modifier en partie l'existant pour pouvoir ajouter notre partie.

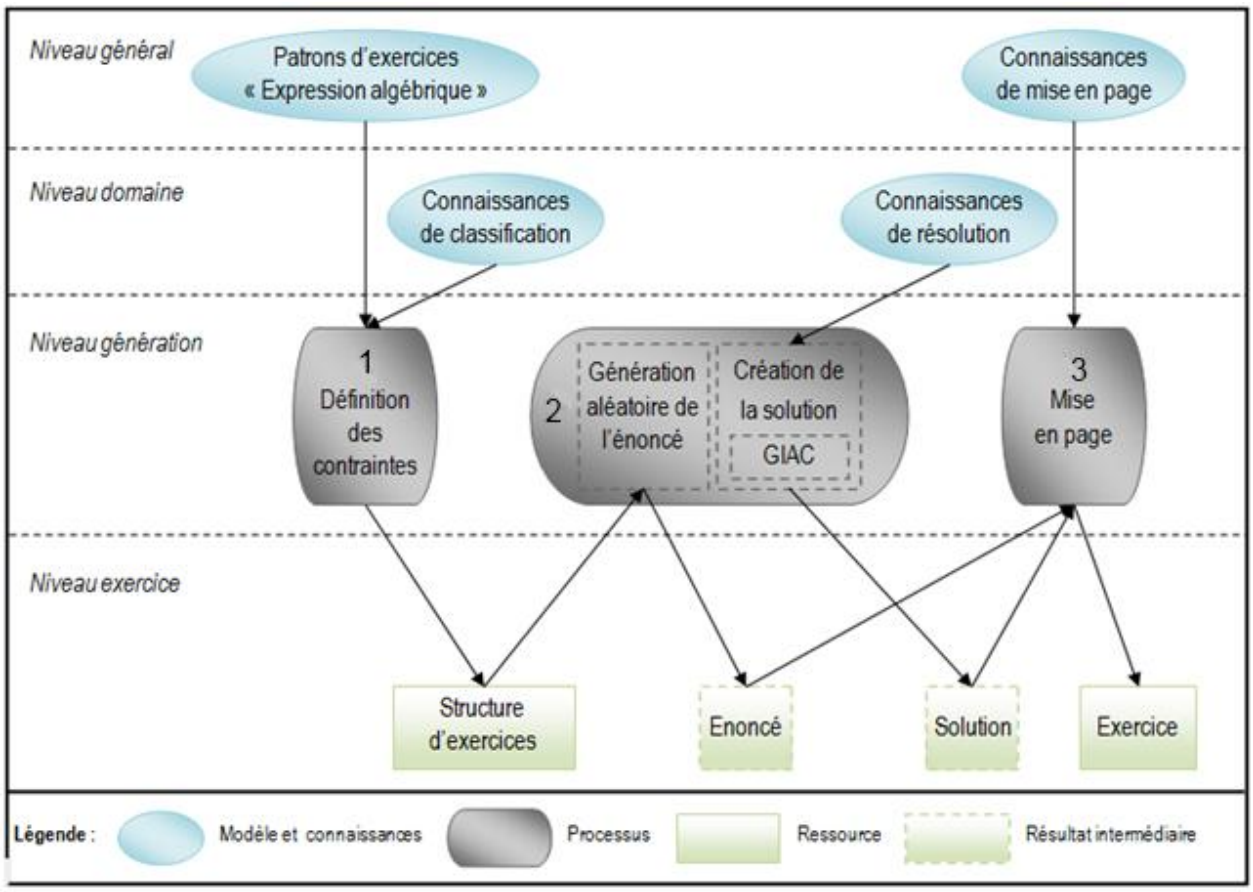


Figure 3 - Architecture générale du générateur

La figure 3 donne l'architecture de notre générateur dans sa version finale, il faut se reporter à l'annexe E pour connaitre sa précédente définition. Le niveau général (partie haute de la figure 3) contient les connaissances indépendantes du domaine pour lequel on souhaite générer un exercice, par exemple les patrons qui nous permettront de générer notre énoncé. Le niveau domaine contient les connaissances propres au domaine d'application, ici par exemple des connaissances concernant la résolution du calcul ou la classification des expressions algébriques sous forme de patron APLUSIX. Le niveau génération contient les processus propres à la création d'un exercice : la définition des contraintes sur un patron d'exercices mémorisées dans une structure d'exercices, l'instanciation de cette structure pour générer un exercice et sa solution et pour finir la mise en page permettant de fournir des exercices ayant une présentation homogène. Enfin, le niveau exercice contient les ressources propres à l'exercice créé parmi lesquelles la structure d'exercices et son instanciation qui contient un énoncé d'exercice et sa correction [LEFEVRE, 09].

Les patrons génériques sont utilisés pour la définition des contraintes (cf 1 sur la figure 3). Grâce à ce patron et ses contraintes associés, est générée une structure d'exercice sous forme d'arbre (cf 2 sur la figure 3). Cette structure est utilisée pour générer aléatoirement un énoncé et sa solution. Pour terminer son cheminement, l'exercice ainsi que sa solution sont mis en forme (cf 3 sur la figure 3) pour donner des sujets ayant toujours la même présentation.

4.2. Création du générateur d'expressions

Nous avons créé le générateur d'expressions dans son intégralité, car aucun des générateurs existants ne pouvait prendre en compte des contraintes pour générer leur expression. Nous avons donc décidé d'écrire un algorithme pour générer des nombres aléatoires en fonction des contraintes définies par l'utilisateur. L'idée n'était pas de créer un générateur d'expressions à proprement parler, mais plutôt un

générateur de nombres soumis à des contraintes. La génération de ces expressions intervient non pas au moment de la création de la fiche de contraintes, mais lors de la création de la feuille d'exercices à la fin du cheminement dans ADAPTE. Pour éviter des problèmes lors de la génération finale, nous effectuons un test au moment d'exporter toutes nos contraintes en format « .xml » pour être sûr qu'une expression viable peut être générée avec ce qu'il a demandé. Dans le cas contraire, l'utilisateur (enseignant) est averti d'un problème et est incité à revoir ces contraintes.

La préoccupation principale concernant l'interface du générateur d'exercices était de pouvoir afficher une expression à l'utilisateur qu'il sache lire et parcourir sans qu'il ait à s'adapter à une mise en forme particulière. Nous avons envisagé d'utiliser un composant ActiveX développé par Mattieu de Villiers, membre de l'équipe KAT/SKA au département de sciences et de technologie de Cap Town en Afrique du Sud [DE VILLIERS]. Mattieu de Villiers a accepté de l'adapter pour notre projet en y ajoutant des fonctionnalités qui nous seraient utiles pour faciliter son utilisation par l'utilisateur. Malgré ces modifications, Delphi ne gérant pas certains évènements sur ce composant (OnClick et OnFocus notamment), nous avons été forcés de le délaisser au profit d'un autre choix.

L'autre technologie d'affichage et de mise en forme d'expressions qui nous a semblé la plus pratique et intuitive pour l'utilisateur a été de générer une image grâce à du Latex (Figure 4). Initialement, l'expression aurait dû rester sous forme de syntaxe LaTeX. Mais, il a finalement été possible de ne plus afficher ces symboles et de laisser l'expression sous sa forme mathématique. Nous nous sommes orientés vers la seconde solution, cela permet à l'enseignant de ne pas avoir à s'adapter à une nouvelle notation et cela lui facilite la lecture de l'expression qu'il a souhaité obtenir.

Sans mise en forme, l'équation avait une forme peu lisible « **Sqrt(2x) + (3/4)x^2 + 30** » mais après notre conversion sous forme d'image grâce à l'utilisation de LaTeX nous obtenons le résultat suivant :

$$\sqrt{(2x)} + \frac{3}{4}x^2 + 30$$

Figure 4 — Expression en LaTeX

On comprend donc bien qu'après cette génération, l'enseignant aura beaucoup moins de difficultés à s'approprier cet affichage plutôt qu'un affichage sous forme de balise LaTeX. Il pourra alors visualiser directement si la forme de l'expression lui convient.

4.3. Modification de l'interface de génération des feuilles d'exercices

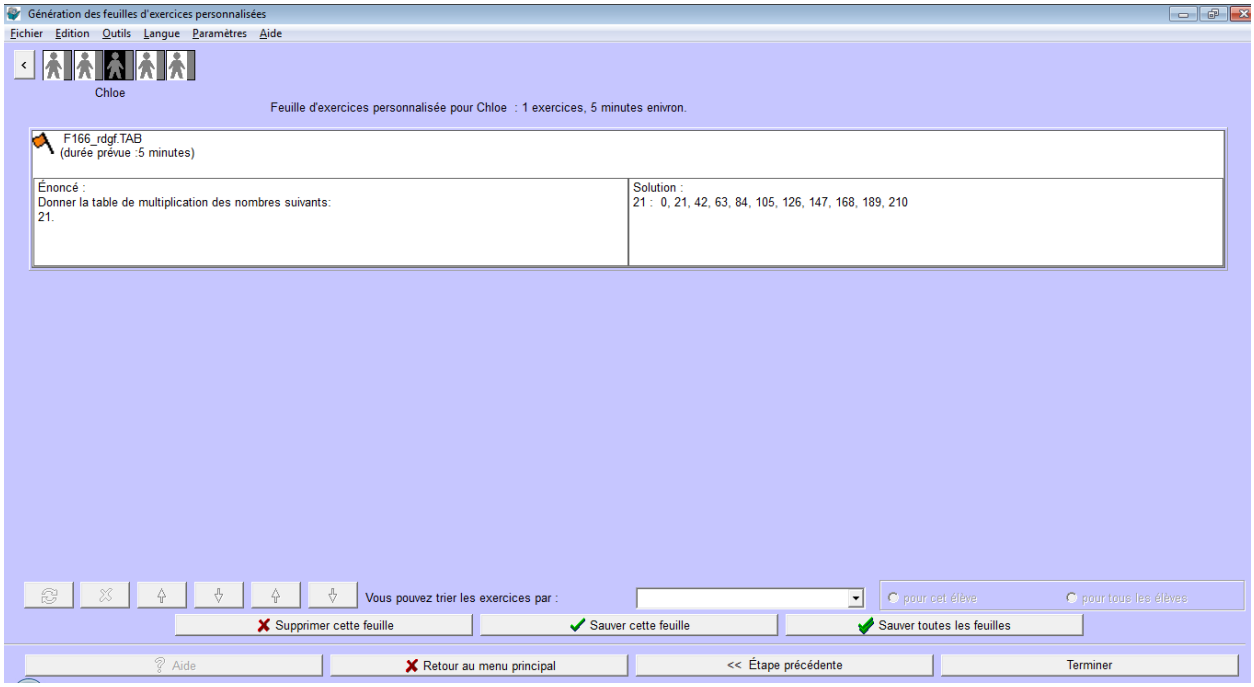


Figure 5 - Interface de génération des feuilles d'exercices

La partie créant les feuilles d'exercices intégrait par défaut une interface comme celle de la figure 5 ci-dessus. Elle prenait en compte les exercices déjà mis en place comme ceux de conjugaison et les tables de multiplication. Nous avons dû intégrer un affichage correct de notre générateur et notamment de l'exercice généré au sein de ces feuilles. La tâche était de pouvoir y insérer une image pour conserver une cohérence de présentation avec ce que nous venons de définir dans la partie précédente. Tout ceci en plus du texte, de façon à pouvoir afficher la formule mathématique sous une forme lisible par l'enseignant (figure D des annexes, page 16).

Conclusion

Nous avons effectué un travail de recherche dans lequel nous avons dû définir le concept d'expressions algébriques pour se l'approprier et pouvoir l'intégrer au module ADAPTE de l'environnement EPROFILEA. Cela nous a amenés à mettre en place les différentes parties de génération d'expression algébrique au sein du module ADAPTE, la modification de la génération des feuilles d'exercices déjà en place pour permettre d'y afficher nos expressions. Nous avons partiellement mis en œuvre la partie génération sur l'ensemble du module. La partie développement d'une expression algébrique est complètement terminée, les contraintes sont correctement prises en compte et le générateur fabrique bien un exercice. Ensuite, le logiciel de calcul formel réussit à analyser l'expression et à en donner une forme développée. Les autres parties restent à faire en se basant ce modèle. Il suffira de réutiliser le code de « Développer » et d'adapter les patrons en conséquence (les contraintes étant les mêmes en règle générale, celles-ci seront les mêmes).

Les parties restantes pourront faire partie d'un travail ultérieur d'adaptation et d'intégration de notre code pour les parties « Équations/Inéquations » et « Système ».

Références

Bibliographie

[CHAACHOUA, 04] H. Chaachoua, « *Le logiciel APLUSIX Standard comme environnement d'apprentissage du raisonnement par équivalence dans le cas des équations et systèmes d'équations.* » Journées de l'APMEP, Orléans, 2004.

[DELOZANNE, 92] E. Delozanne, « *Explications en EIAO : Études à partir d'ELISE, un logiciel pour s'entrainer à une méthode de calcul de primitives.* », Université du Maine, p32-69, 1992.

[GRANDBASTIEN, 01]E. Bruillard, E. Delozanne, P. Leroux, Paul Delannoy, X. Dubourg, P. Jacoboni, J. Lehuen, D. Luzzati, P. Teutsch « *Quinze ans de recherche informatique sur les sciences et techniques éducatives au LIUM* » dans STE Volume 7 – n°1/2000 – Education et Informatique, p 87-145 (p94-97), Direction : E. Bruillard, M. Grandbastien, Édition : Hermes, ISBN 2-7462-0232-8, 2001.

[GUIN, 97] N. Guin, « *Reformuler et classer un problème pour le résoudre. L'architecture SYRCLAD et son application à quatre domaines* », Paris, 1997.

[JEAN-DAUBIAS et al. 2009]Jean-Daubias, S., Lefevre, M., Guin, N. « *Adapte, un outil générique pour proposer des activités pédagogiques personnalisées* ». Workshop Prise en Compte de l'Utilisateur dans les Systèmes d'Information (PeCUSI), INFORSID 2009, Toulouse, France, pp. 51-62, 26 mai 2009.

[LEFEVRE, 08] M. Lefevre, S. Jean-Daubias, N. Guin, « *Generation of exercices within the PERLEA project.* », RR-LIRIS-2008-011, Lyon, 2008.

[LEFEVRE, 09] M. Lefevre, « *Processus unifié pour la personnalisation des activités pédagogiques : méta-modèle, modèles et outils. »*, Thèse doctorale, Lyon, 2009.

Netographie

[DE VILLIERS] http://www.dip.ee.uct.ac.za/~mdevill/

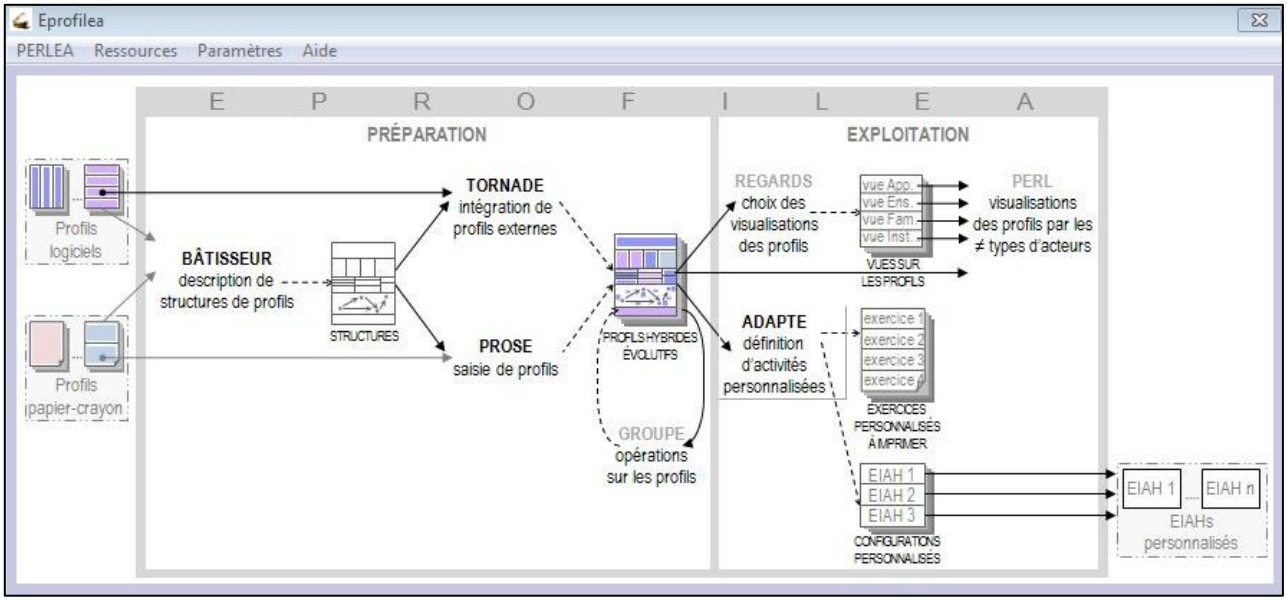
[PARISSE, 07] B. Parisse, C. Serret, M. Gandit, R. De Graeve, http://revue.sesamath.net/spip.php?article90 & http://revue.sesamath.net/spip.php?article98, MathémaTICE N°6 & 7 Septembre 2007.

[DAN] http://premiumorange.com/daniel.robert9/Premiere_partie_algebre.html

Annexes

Annexe A – Architecture de l'environnement EPROFILEA & Module ADAPTE

Cette annexe montre l'enchainement des modules contenu dans l'environnement EPROFILEA pour aller de la construction de la structure de profil jusqu'aux environnements personnalisés pour chaque apprenant à la fin.



Eprofilea est un environnement regroupant un ensemble de modules formant un tout. Les modules situés en amont du module ADAPTE permettent de gérer les profils des apprenants aussi bien propre à Eprofilea ou venant d'une application extérieure. Ces profils peuvent ensuite être visualisés par des acteurs extérieurs du système, ou ils peuvent être utilisés dans le module ADAPTE pour créer des sessions d'exercices personnalisées pour les apprenants. Les exercices ainsi générés peuvent alors être imprimés suite à leur génération dans des fichiers « .rtf » ou exportés grâce à leur génération en format « .xml » facilement réutilisable.

Annexe B – Un type de patron dans APLUSIX

```
;A et D égaux à 1 ou -1
;Cela correspond aux exercices dits de suppression de parenthèses
{[nom PatronExpADev2]
 [Com "Les expressions polynômes à développer de forme générale A(B + C) +D (E + F), A, B, C, D, E et F peuvent être constantes ou monômes de degré 1 ou plus"]
 [sorteDe PatronExpADevelopper]}

{[nom PatronExpADev21]
 [sorteDe PatronExpADev2]
 [patron <<(ax+b)-(dx+e)>>]
 [domaine ((a entier* petit) (b entier* petit) (d entier* petit) (e entier* petit))]
 [frequence 2]}
```

Les patrons dans APLUSIX sont définis comme ci-dessus. Il possède un nom qui reflète ce qui devra être fait sur l'expression et un numéro différencie chaque patron pour chaque genre. Ici le genre est Dev (Développer). Le terme « sorteDe » permet de connaître le père du patron. Son père est placé juste au-dessus et permet de définir le type général de l'expression, ici on peut voir que ce sont les équations de la forme : « A(B + C) +D(E + F) ». Dans le champ « Patron » est affiché un type détaillé de ce qui peut exister pour chaque forme. Dans notre exemple, on voit qu'A et D sont fixés à une valeur de 1, que C et F n'ont pas d'inconnue de définie et que les 2 derniers monômes ont des coefficients aléatoires et des inconnues de degré 1. La fréquence 2 permet de définir si cette expression doit être utilisée souvent par rapport aux autres qui peuvent avoir des degrés différents.

Pour notre générateur, nous avons gardé les patrons « père » des fichiers de patrons d'APLUSIX, ici nous avons donc récupérés « **A(B + C) +D (E + F)** ».

Annexe C – Les Patrons Génériques

Ci-dessous se trouve l'ensemble des patrons génériques ayant pu être dégagé des fichiers de patrons APLUSIX. Les didacticiens avaient dégagé les patrons principaux et tous les patrons sous-jacents et particuliers pouvant en découler [CHAACHOUA, 04]. Nous nous sommes donc limités aux principaux.

C.1. Patrons génériques pour développer une expression

A(B+C)
A(B+C)+D(E+F)
(A+B)+(C+D)
E(A+B)+(C+D)+F(G+H)+(I+J)

C.2. Patrons génériques pour factoriser une expression

A + B
A + B + C

A(B + C) + D(E + F)
A(B + C) + D(E + F) + G(H + I)
A(B + C) + (D + E)(F + G)
A(B + C)(D + E) + F + G
A(B + C)(D + E) + F(G + H)
A(B + C)(D + E) + F(G + H)(I + J)

(A + B)(C + D) + E + F
(A + B)(C + D) + E(F + G)
(A + B) + C(D + E)(F + G)
(A + B)(C + D) + (E + F)(G + H)
(A + B)(C + D) + E(F + G)(H + I)
(A + B)(C + D) + E(F + G) + H(I + J)
(A + B)(C + D) + E(F + G) + H + I

(A + B)^2 + C
(A + B)^2 + C + D
(A + B)^2 + (C + D)^2
(A + B)^2 + (C + D)(E + F)
(A + B)^2 + C(D + E)(F + G) + (H + I)
(A + B)(C + D) + E(F + G)^2
(A + B)(C + D) + E^2 + F + G
(A + B)(C + D) + E + F + G + H + I
(A + B)(C + D) + E(F + G + H)
(A + B)^2 + C + D + E
(A + B + C) + (D + E + F)
(A + B)^2 + (C + D)^2 + (E + F)(G + H)

C.3. Patrons génériques pour des expressions d'équations et d'inéquations

A ~ B(C + D) + E
A + B ~ C(D + E)
A + B ~ C(D + E) + F
A + B + C ~ D(E + F) + G
A(B + C) ~ D
A(B + C) ~ D(E + F)
A(B + C) ~ D + E(F + G)
A(B + C) + D(E + F) + G ~ H(I + J)
A(B + C) + D(E + F) + G ~ H(I + J) + K(L + M)
A(B + C) + D(E + F) + G ~ H(I + J) + K(L + M) + N

A ~ B
(A/B) ~ C
A + B ~ C
(A/B) ~ (C/D)
A + B ~ C + D
A + B ~ C + D
A + B + C ~ D
A + B + C ~ D + E
A + B + C + D ~ E
A + B + C ~ D + E
A + B + C ~ D + E + F
A + B + C + D + E + F ~ G
A + B + C + D ~ E + F + G + H

C.4. Patrons génériques pour des systèmes d'expressions

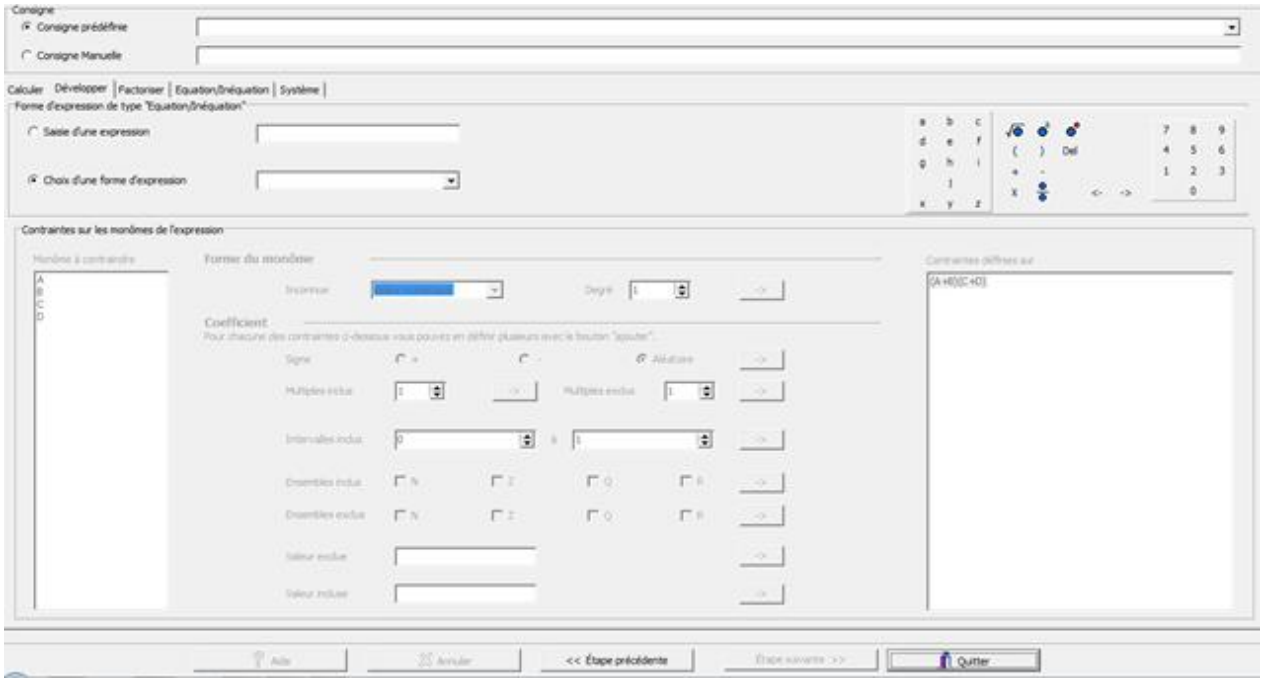
A + B = C AND D + E = F {ax+by=e _and_ cx+dy=f}

A + B + C + D + E AND F + G = H + I + J {ax+by=gx+hy+e _and_ cx+dy=ix+jy+f}

A = B + C AND D + E = F {by=e+[_sqrt_a]x _and_ cx+d[_sqrt_a]y=f}
A = B + C AND D = E + F {bx=e+[_sqrt_a]y _and_ cx=d[_sqrt_a]y+f}
A + B = C AND D = E + F {bx-e=[_sqrt_a]y _and_ 0=cx+d[_sqrt_a]y+f}

Annexe D – L'interface des contraintes

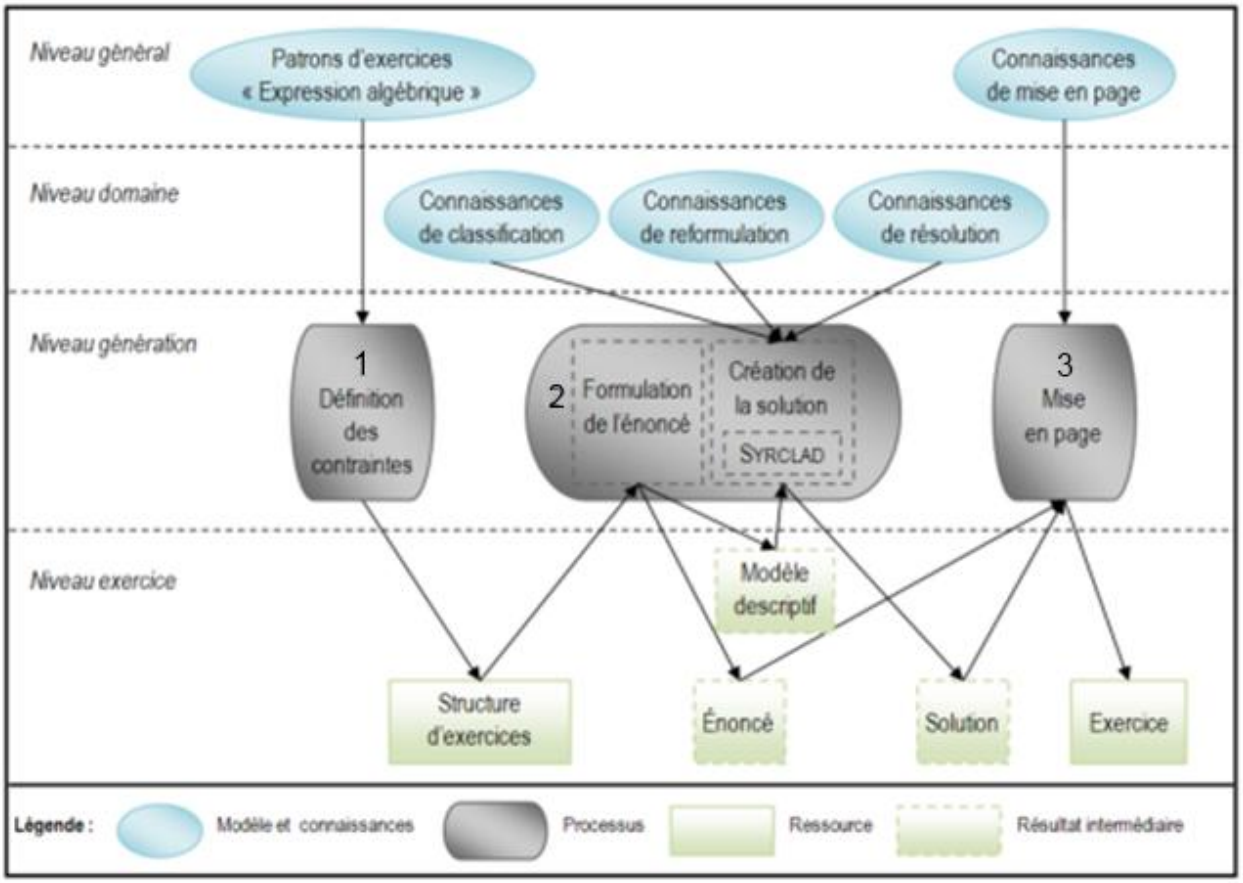
Dans l'image ci-dessous on peut voir la fenêtre de contraintes qui a pu être mise en place et qui sera donc présente dans le module final.



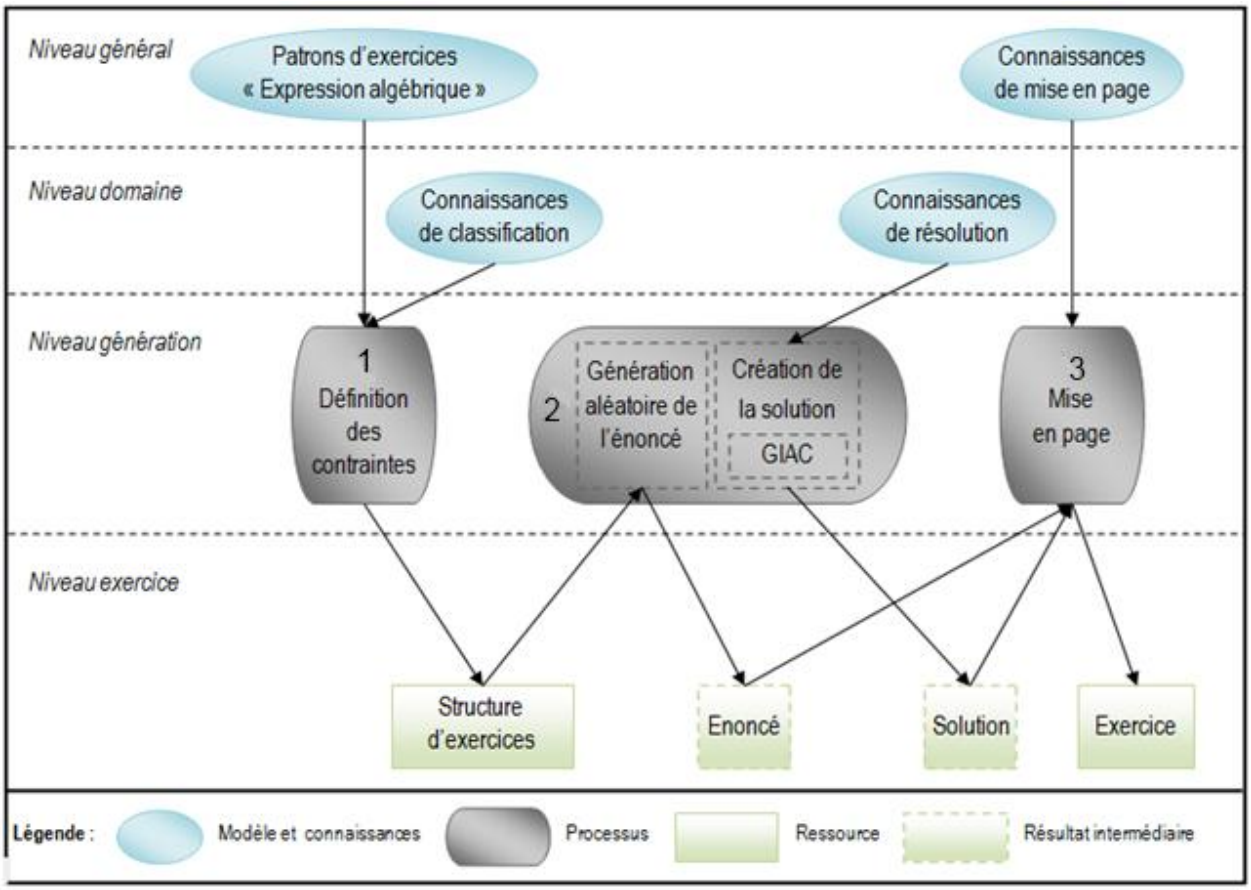
Dans la partie haute de la fenêtre se trouve la saisie de la consigne, l'utilisateur a donc le choix d'utiliser soit une consigne qui a été prédéfinie en fonction des types d'exercices existants ou de saisir sa propre consigne.

Une fois le type de consigne choisi, l'utilisateur peut choisir parmi les patrons génériques proposés et imposer (ou non) des contraintes sur le patron qu'il aura choisi. Pour faciliter sa saisie lors des expressions contenant des caractères particuliers, il a à sa disposition un clavier virtuel avec les caractères tel que les racines carrées, les puissances…

Annexe E – Vision du générateur précédente et actuelle



Avant la mise en place de notre version, le générateur avait été pensé comme ci-dessus. On peut constater que la seule différence notable se trouve au niveau du générateur de l'énoncé et de sa solution. Auparavant, celui-ci était censé utiliser SYRCLAD en tant que résolveur avec un moteur d'inférence en PROLOG.



Après la mise en place de notre version du générateur, peu de choses ont changé. Le seul grand changement est la mise en place d'une génération aléatoire de l'énoncé (cf 2 ci-dessus) combiné avec le logiciel de calcul formel GIAC et non plus SYRCLAD comme il avait été défini précédemment.