Apprentissage du raisonnement assité par ordinateur

C. Raffalli et R. David
Laboratoire de Maths, Université de Savoie
{david,raffalli}@univ-savoie.fr
www.lama.univ-savoie.fr

1 Introduction.

La Licence et la Maîtrise de mathématiques de l'université de Savoie comportent des cours optionnels de logique [3]. Il y a plusieurs années, ces cours étaient une introduction classique aux bases de la logique mathématique, mais vu les difficultés de raisonnement des étudiants, l'objectif du cours de Licence s'est transformé petit à petit pour devenir un apprentissage du raisonnement mathématique.

L'introduction de PhoX [13], l'assistant de preuve écrit par C. Raffalli, a permis de faire des séances de travaux pratiques sur machine. Avec la machine, l'étudiant ne peut pas faire de preuve fausse. L'erreur est remplacée par un blocage que l'enseignant peut aider à résoudre. Notre expérience montre que cela apporte plus à l'étudiant que de signaler les phrases erronées de son raisonnement car celui-ci ne comprend pas toujours les explications de l'enseignant.

Après quatre années d'expérience, cet outil nous semble d'une grande utilité. Il montre clairement que, jusqu'à la Licence (et sans doute aussi la Maîtrise), les difficultés majeures des étudiants ne proviennent pas des concepts mathématiques qu'ils découvrent mais plutôt de leur incompréhension très forte de la nature du raisonnement mathématique.

Les préalables nécessaires à l'utilisation du logiciel sont faits, dans le cours de Licence, dans le cadre d'un exposé de logique formelle mais la différence avec une présentation informelle est très faible et une utilisation de PhoX dès la première année du Deug est envisageable : nous en tenterons l'expérience avec les étudiants de première année de Deug Mias au début du deuxième semestre de cette année.

Dans cet article nous décrivons les points essentiels de notre expérience. Le lecteur intéressé trouvera sur la page web des auteurs une version de cet article contenant trois sections supplémentaires : la première donne un exemple de preuve avec PhoX, la deuxième présente le fonctionnement général du logiciel. Cela doit en permettre une première approche. La dernière donne, de manière informelle, les règles de démonstration.

2 Preuves et Machines.

Les *assistants de preuves* comme ACL2 [9], COQ [7], HOL [6, 8], Isabelle [11], LEGO [12], PVS [10] ... permettent de réaliser sur machine des démonstrations mathématiques dont la correction est garantie. Il ne s'agit pas de démonstrateurs automatiques : l'utilisateur peut (et le plus souvent doit) guider la machine dans le raisonnement. Le logiciel se contente de vérifier que chacune des étapes de la démonstration est correcte.

La plupart des logiciels ci-dessus disposent d'algorithmes plus ou moins sophistiqués de démonstration automatique permettant d'aboutir sans intervention de l'utilisateur. Mais, cet aspect n'a pas ou peu d'intérêt pédagogique car c'est la vérification de chacune des étapes de raisonnement qui est intéressante. Ces systèmes ont été développés par des spécialistes visant essentiellement des applications de vérifications formelles de programmes, de circuits, de protocoles de communication, etc. Leur prise en main n'est pas simple et nécessite clairement trop de temps pour un étudiant de premier ou second cycle à l'université. Aucun d'entre eux n'est donc, en ce qui nous concerne, satisfaisant.

À notre connaissance, Phox est le seul assistant de preuves utilisé en France dans un but pédagogique. L'outil idéal est un système qui est à la fois suffisamment puissant pour permettre de faire des démonstrations de niveau Deug, et aussi simple et intuitif que possible. Ces objectifs sont difficiles à concilier car la puissance s'obtient, en général, par un grand nombre de commandes (chacune d'elles permet d'avancer dans une situation particulière) et l'apprentissage en est donc nécessairement long.

3 Notre expérience pédagogique.

3.1 Un peu de logique

Notre cours de Licence [3] présente *la logique du premier ordre* avec ses connecteurs et ses quantificateurs. On introduit aussi les *règles de démonstration* de la *déduction naturelle*.

Un tel exposé formel n'est probablement pas nécessaire avant d'utiliser PhoX mais une présentation informelle minimale décrivant ce qu'est un énoncé mathématique et ce qu'est une démonstration est indispensable. En effet :

- Le logiciel ne manipule pas des phrases en français mais des formules mathématiques. Il faut que l'étudiant puisse les lire. Cette notation symbolique n'ajoute pas de difficultés car les étudiants ne font pas de différence entre un symbole $\forall$ et le texte «pour tout» une fois ce symbole défini.
- Chaque *commande* du logiciel va correspondre à une étape précise d'un raisonnement mathématique et il faut introduire un minimum de vocabulaire pour expliquer ces commandes.

D'autre part et même si c'est ce qui est fait dans la plupart des cas, est-il raisonnable d'enseigner les mathématiques sans avoir, au préalable, défini, même informellement, ce qu'est une proposition ou une démonstration, alors que c'est la base même de toutes les mathématiques ! Si on ne le fait pas, il y a un clivage entre les étudiants qui le comprennent seuls (ça a été le cas de ceux qui sont devenus enseignants !) ... et les autres.

Nous donnons donc ci-dessous quelques réponses informelles et suffisantes pour l'utilisation du logiciel.

Qu'est ce qu'un énoncé mathématique ?

Les énoncés (ou formules) sont construits à partir de briques de base qui dépendent du type de mathématique que l'on fait : cela peut être une égalité, une inégalité, l'appartenance d'un objet à un ensemble, le parallélisme de deux droites, etc. De manière générale, ces briques de base sont les propriétés que, à ce niveau, on ne souhaite pas décomposer en briques plus élémentaires. Par exemple, on peut considérer que « n est premier » est une brique de base ou le décomposer en sa définition à partir de la division.

Pour assembler ces briques on dispose de connecteurs et de quantificateurs. Ce sont : la conjonction ($\wedge$), la disjonction ($\vee$), l'implication ($\rightarrow$), la négation ($\neg$), la quantification universelle ($\forall$) et la quantification existentielle ($\exists$).

Il est important qu'un étudiant puisse traduire un énoncé informel en un énoncé utilisant les symboles ci-dessus et vice-versa. Ceci est loin d'être immédiat. Dans le cours de logique, nous faisons de nombreux exercices de traduction énoncé formel versus énoncé informel et l'expérience montre qu'ils ne sont pas inutiles. Il est par exemple bien connu que les différentes manières de dire $A \rightarrow B$ ne sont pas comprises par les étudiants : « B est une condition nécessaire de A», « A est une condition suffisante pour avoir B », « si A alors B », etc. Cette difficulté doit être surmontée par l'étudiant : il doit savoir lire un énoncé et il doit comprendre ce qu'est un énoncé et ce qui n'en est pas un.

Qu'est ce qu'un énoncé vrai ?

Un étudiant de Mias 2 à qui on venait de démontrer que toute matrice symétrique réelle est diagonalisable avait du mal à être sûr qu'il ne «tomberait» jamais, par malchance, sur un contre-exemple alors qu'un autre disait «un énoncé est vrai ssi il n'a pas de contre-exemple».

Il n'est pas nécessaire de définir formellement cette notion de vérité (le logicien dit la *sémantique*) : on serait, en effet, conduit à dire $A \wedge B$ est vrai si et seulement si A est vrai et B est vrai et l'on n'a fait que définir la vérité de la conjonction à partir d'elle même, en utilisant un « si et seulement si » dont la nature n'est pas si claire ! Une telle définition (pourtant essentielle pour le logicien) n'est ni importante ni utile pour l'étudiant de Deug car le sens intuitif qu'à l'étudiant de la conjonction, de la quantification universelle suffit. Le sens de l'implication et le raisonnement par l'absurde posent cependant des problèmes particuliers sur lesquels nous insistons.

Il est, par contre, important de dire comment on fait pour établir la vérité d'un énoncé : en faisant des démonstrations !

Qu'est ce qu'une démonstration ?

Une démonstration est une suite d'étapes : chacune d'elles correspond à l'application d'une règle. A chaque étape d'une démonstration, on dispose d'une *liste des connaissances*. Dans la pratique courante on parle plutôt des *hypothèses* mais ce terme, que les logiciens utilisent couramment dans ce sens, peut être trompeur car il semble dénoter quelque chose de constant.

Cette liste évolue sans cesse. Au début d'une preuve, la liste des connaissances est vide ou plutôt ne contient que l'ensemble des résultats du cours (théorèmes, lemmes, axiomes, ...). Certaines règles permettent d'étendre la liste des connaissances pour la suite de la démonstration. Si une preuve nécessite l'étude de plusieurs cas, les listes de connaissances de chaque cas sont identiques au départ, mais elles évoluent indépendamment par la suite.

Pour chaque connecteur et chaque quantificateur, il y a deux règles :

L'une dite, d'*introduction*, qui sert à *démontrer* une formule commençant pas ce symbole. Par exemple, pour démontrer $A \to B$ on ajoute A à la liste de connaissances et on démontre B.

L'autre dite, d'*élimination*, qui permet d'*utiliser* une hypothèse (ou un lemme, un axiome, ...) commençant pas ce symbole. Par exemple, si $A \vee B$ fait partie de la liste de connaissances et qu'on veut démontrer C, il suffira de démontrer $A \to C$ *et* $B \to C$. On notera que le fait d'utiliser un « et » là ou il y a un « ou » perturbe beaucoup les étudiants !

Il peut paraître étonnant que cela suffise pour tout démontrer (c'est le théorème de complétude de Gödel) : ce n'est pas si difficile à exposer et pourtant beaucoup de mathématiciens l'ignorent !

On ne donnera pas ici, même de manière informelle, ces règles de démonstration. Le lecteur intéressé pourra soit chercher à les imaginer lui même (il y en a 14 et il n'est pas si simple de les trouver !) soit consulter un livre de logique (par exemple, [3])

3.2 Utilisation du logiciel

Dans notre cours de Licence, on commence à faire des démonstrations dès la première séance de TD. D'abord avec des formules sans quantificateurs. C'est en particulier l'occasion de montrer des *utilitaires* : par exemple, $[(A \wedge B) \to C] \leftrightarrow [A \to (B \to C)]$ qui signifie qu'avoir une hypothèse sous la forme d'un «et» ou avoir les deux hypothèses séparément est, en fait, le même chose. On continue ensuite avec des formules utilisant des quantificateurs. On commence là aussi avec des exemples simples tel que $\exists x(A \wedge B) \to \exists x A \wedge \exists x B$. Quand on leur demande de montrer la réciproque ... la moitié d'entre eux y arrivent parce qu'ils ont donné le même nom au x qui satisfait A et à celui qui satisfait B ! On avait pourtant insisté sur ce problème de nom dans la règle du $\exists$. Le fait de pouvoir leur montrer qu'ils ont mal appliqué la règle est satisfaisant pour eux car ils reconnaissent la cause précise de leur erreur.

Après 4 ou 5 séances de TD où l'on fait des preuves sur papier, on passe sur machine. Les étudiants ont peu de difficulté à maîtriser le logiciel : deux séances de TP suffisent.

L'utilisation de la machine permet de faire des exemples compliqués car l'écriture, à la main, de preuves formelles est souvent fastidieuse mais, surtout, la machine contrôle la correction de la preuve. En effet, l'étudiant dit à la machine quelle règle il veut appliquer : la machine l'applique ... si elle le peut ... et proteste si elle ne peut pas.

La machine n'intervient pas pour déterminer l'ordre d'utilisation des règles mais elle aide pour une utilisation de ces règles, en particulier pour le choix des noms des variables, ce qui est une des difficultés majeures des étudiants.

Dans la suite de cet article, il sera commode de pouvoir nommer quelques règles : une règle d'introduction (resp. d'élimination) est notée avec un i (resp. un e) en indice, par exemple $\forall_i$ est le nom de la règle d'introduction du $\forall$: pour montrer $\forall x\, A(x)$, je peux montrer $A(y)$ avec un *nouvel* y i.e. un y n'apparaissant pas avant.

Quand on utilise les règles $\forall_i$ et $\exists_e$ il faut choisir un nouveau nom et c'est la machine qui s'en charge : ainsi, l'étudiant ne pourra pas faire de preuve de $\exists x A \wedge \exists x B \rightarrow \exists x(A \wedge B)$ car quand on utilisera la règle $\exists_e$ avec $\exists x A$ d'une part, avec $\exists x B$ d'autre part, la machine donnera des noms différents aux deux objets. Quand on utilise les règles $\forall_e$ et $\exists_i$ la machine demande quel est l'objet avec lequel on utilise la règle en introduisant un « ? » et c'est à l'étudiant de dire ce qu'est le « ? ». La prise de conscience de cette *dualité* entre $\forall_i$ et $\exists_e$ d'une part et $\forall_e$ et $\exists_i$ d'autre part aide beaucoup les étudiants.

3.3 Des exemples

De manière générale la difficulté, pour nous, est de trouver des exemples pour lesquels :

- La formalisation est très proche du raisonnement "informel" vu dans le cours d'analyse ou d'algèbre.
- Les propriétés des objets utilisés ne nécessitent pas une axiomatisation compliquée.
- Le raisonnement, i.e. l'enchaînement des $\forall$ et des $\exists$ est, pédagogiquement, intéressant.

On notera qu'il est plus facile de trouver des exemples d'analyse que d'algèbre et que ces résultats sont souvent pédagogiquement plus intéressants car les formules contiennent une alternance de 3 quantificateurs ($\forall \varepsilon \exists \alpha \forall x$...) et c'est ce qui est difficile pour les étudiants. Voici les exemples que les étudiants ont traités sur machine pendant ces quatre années :

- le théorème des valeurs intermédiaires dans $\mathbb{R}$,
- unicité de la limite,
- l'adhérence de la réunion est égale à la réunion des adhérences (dans un espace métrique),
- la définition de la continuité avec des inégalités strictes (par exemple $|x - y| < \alpha$) est équivalente à celle avec des inégalités larges (i.e. $|x - y| \leq \alpha$),
- l'image d'un connexe par une fonction continue est un connexe (dans un espace métrique),
- deux permutations de E à support disjoints commutent,
- si la réunion de deux sous-ensembles de $\mathbb{R}$ est non bornée, alors l'un d'eux est non borné,
- dans un anneau principal, un idéal est maximal ssi il est premier,

– dans un anneau, un idéal premier est irréductible.

Chacun d'eux prend beaucoup de temps : 2 heures pour l'unicité de la limite ou l'adhérence de la réunion égale à la réunion des adhérences et plus de 4 heures pour le théorème des valeurs intermédiaires ou le résultat sur les anneaux principaux. Ce n'est pas la maîtrise du logiciel qui est en cause : quand on fait ces exemples, ils le maîtrisent déjà bien. C'est bien les détails de la preuve elle même qui ne sont pas clairs pour eux !

3.4 Les difficultés des étudiants

Les principales difficultés sont, pour les étudiants, les suivantes :

- Le fait que, au cours d'une démonstration, l'ensemble des hypothèses varie et il faut donc savoir, à tout moment, ce qui est disponible et ce qui ne l'est pas.
- La confusion entre hypothèse et conclusion : on surprend quelquefois les étudiants à chercher à montrer quelque chose qui est, en fait, une hypothèse !
- La gestion des variables libres et liées : savoir reconnaître quels sont les objets qui, à un moment de la démonstration, « existent ».
- Bien que la machine se charge de faire les renommages nécessaires, l'étudiant peut en faire lui aussi et il peut ainsi décider, à un moment de la preuve, de renommer x en y alors que y était un nom déjà "pris". C'est raisonnable et on le fait souvent si l'on sait que y ne servira plus. Mais on a eu le cas d'un étudiant qui trouvait que renommer y en x, permettait d'utiliser l'hypothèse $x = 0$ pour y ! Avant d'avoir compris ce qu'avait fait l'étudiant, on a eu du mal à comprendre pourquoi la machine refusait de prouver $x = 0$ alors que c'était une des hypothèses : la machine n'avait pas oublié qu'il y avait maintenant deux x distincts.
- La distinction, pour les quantificateurs, entre les règles d'introduction et d'élimination :
  - Pour montrer un $\forall x\, A$, il n'y a pas d'autre moyen que de prendre un *nouvel* x (c'est la machine qui le choisit !) et de montrer A pour cet x. Le problème est que cette affirmation n'est pas (toujours) vraie : on peut aussi faire un raisonnement par l'absurde !
  - Pour utiliser une hypothèse de la forme $\forall x\, A$, il faut l'utiliser avec un objet t et cet objet c'est à l'étudiant de le donner.
- Dans la preuve d'une formule de la forme $\exists M \forall n$ ... une erreur fréquente est que les étudiants donnent un M qui dépend du n. La machine est suffisamment souple pour accepter de ne connaître le M qu'à la fin de la preuve (c'est souvent utile en pratique quand on coupe les ε en 4) mais elle proteste si le M qu'on lui donne alors dépend du n. Quand l'étudiant dit à la machine, par exemple, je pose $M = 2n + 1$ et que la machine refuse, cela l'aide bien à comprendre son erreur alors que si c'est le prof qui dit quelque chose du genre «ton M n'a pas le droit de dépendre de n», il l'accepte beaucoup moins bien.

  Sur machine, cela correspond à vouloir faire un $\exists_i$ trop tôt, i.e. dire que le M qu'on cherche est un objet que la machine ne «connaît» pas encore.

- Bien que moins difficile, l'implication pose aussi des problèmes. La question traditionnelle sur le fait que la formule «faux $\to$ vrai» est vraie ne se pose pas ici puisqu'on ne parle que de preuve et pas de vérité. Par contre, penser que si on sait $\forall x(A[x] \to B[x])$ et qu'on veut montrer $B[t]$, il suffira de montrer $A[t]$ n'est pas quelque chose d'acquis spontanément et il faut faire de nombreux exemples.
- Appliquer deux fois une hypothèse de la forme $\forall x$, une première fois avec un x introduit ailleurs et une deuxième fois avec, par exemple, $f(x)$. Si le x introduit s'appelait x_0, ça poserait sans doute moins de problèmes !

De manière plus générale, les étudiants ont beaucoup de mal à prouver des faits très élémentaires sur lesquels, dans un cours d'algèbre ou d'analyse, on ne passerait que quelques minutes en Deug et on dirait simplement "évident" en licence. Le coté rigoureux de la machine est pourtant nécessaire à l'étudiant pour forger sa compréhension de ces notions qui, nous l'avons sans doute trop oublié, sont difficiles. En voici deux exemples.

- Pour la preuve du théorème des valeurs intermédiaires, nous avions repris la preuve classique : si $m = Sup\{x \in [a, b] \,/\, \forall y \in [a, x],\ f(y) < c\}$ alors $f(m) = c$ et nous avions décomposé la preuve en une dizaine de lemmes à montrer. La plupart des étudiants ont mis deux heures pour prouver le premier : si f est continue et $f(x) < c$ alors il existe un voisinage de x (i.e. il existe $\alpha > 0$ ...) sur lequel f est plus petite que c. Ils ont eu également beaucoup de mal à trouver un majorant de l'ensemble $\{x \in [a, b] \,/\, \forall y \in [a, x],\ f(y) < c\}$. L'utilisation concrète des deux propriétés de la borne supérieure (c'est un majorant et il n'y en a pas de plus petit) est un exercice difficile, surtout quand la machine impose une grande précision.
- Dans la preuve que les formules C_1 et C_2 ci-dessous sont équivalentes
$$C_1 : \forall x \forall e{>}0\, \exists a{>}0\, \forall y(|x - y| < a \to |f(x) - f(y)| < e)$$
$$C_2 : \forall x \forall e{>}0\, \exists a{>}0\, \forall y(|x - y| \leq a \to |f(x) - f(y)| \leq e)$$
pour montrer $C_1 \to C_2$, il faut, après avoir introduit x et e, utiliser C_1 avec x et e. C_1 donne alors un a. Ce n'est pas celui qu'on cherche (c'est $a/2$ qu'il faut prendre !) mais beaucoup d'étudiants partent pourtant dans cette direction et, bien sûr, n'aboutissent pas ! Nous avions fait cette preuve en détail sur papier en TD et nous l'avons redonné, sur machine, pour l'évaluation. Au bout d'une heure, aucun étudiant n'avait fait la preuve !

4 En guise de conclusion.

A la fin du semestre, nous demandons aux étudiants de donner leur avis sur l'intérêt de cet apprentissage. Cette petite enquête nous semble montrer que cette expérience est appréciée. Pourtant, nous ne faisons pas de démagogie car les notes qu'obtiennent les étudiants à l'examen n'ont rien d'exceptionnel.

4.1 Questionnaire

L'objectif du cours de Logique était double :

objectif 1 : vous aider (en formalisant la notion de démonstration) à faire des preuves correctes (également dans les autres cours) en particulier en manipulant correctement les notions de variables libres ou liées et les règles d'introduction et d'élimination de l'implication et des quantificateurs.

objectif 2 : faire un peu de logique en tant que telle, i.e.
- vous faire comprendre la différence entre la syntaxe (les démonstrations) et la sémantique (la notion de structure et de vérité)
- vous montrer comment on obtient le «début» des maths, à savoir les entiers et les ensembles.

Pouvez vous répondre aux questions suivantes. Elles ont pour but d'essayer d'améliorer ce cours pour l'an prochain.

Répondez à chaque question par une note de 0 à 5 :

1. En ce qui vous concerne (et indépendamment de votre note finale) pensez vous que l'objectif 1 a été atteint ? (0 = pas du tout, 5 = tout à fait)
2. Idem pour l'objectif 2 ? (0 = pas du tout, 5 = tout à fait)

   Les TP sur machine avaient pour but de vous aider à réaliser l'objectif 1
3. Pensez vous que ça vous a effectivement aidé ? (0 = pas du tout, 5 = tout à fait)
4. A-t-on passé trop de temps à manipuler ce logiciel ? (0 = trop, 5 = pas assez)
5. Comment avez vous trouvé l'apprentissage du logiciel ? (0 = très difficile, 5 = très facile)

4.2 Résultats

Le tableau ci-dessous donne les moyennes obtenues à chacune des questions. Ce ne sont bien sûr que des moyennes qui cachent le fait qu'à certaines questions il y a des 5 ... et des 0.

Question	1	2	3	4	5
98/99	2,3	2,3	2,5	2,3	1,9
99/00	2,8	2,1	2,9	3,5	3,1
00/01	3,2	2,8	3,3	3,3	3,0
01/02	3,3	3,0	2,8	3,1	3,0

4.3 Perspectives

Même si les résultats ne sont pas spectaculaires, cette expérience nous paraît extrêmement positive : elle nous a permis de mieux comprendre où sont les difficultés majeures des étudiants et elle les a fait progresser.

Pour que ça marche mieux, il faudrait sans doute pouvoir y passer beaucoup plus de temps et commencer à faire ce genre de travail dès le Deug. Notre expérience montre que l'acquisition de raisonnements aussi simples (pour nous) que ceux qui sont nécessaires pour démontrer l'unicité de la limite nécessite beaucoup de temps pour les étudiants : comme on ne peut envisager de passer une heure dans le cours d'analyse pour montrer ce résultat, il faut faire cet apprentissage ailleurs. Quand le processus d'imitation de l'enseignant fonctionne (cela a été le cas chez nous quand nous étions étudiants), cela suffit mais il semble bien que, chez la plupart des étudiants, ce processus ne suffise plus. Nos collègues en seront ils convaincus ?

Les étudiants peuvent utiliser PhoX en dehors de séances de TP encadrés. Ils peuvent donc faire des preuves détaillées de résultats vus dans les autres cours. Soyons réalistes : on devrait dire «ils pourraient» car, dans les faits, ils ne le font pas. Ils doivent en effet ingurgiter des tas d'autres choses incompréhensibles ! Pour le futur immédiat, nous allons :

- Essayer d'améliorer l'interface pour rendre le logiciel plus convivial et utilisable en particulier par les étudiants de Deug. Mais c'est un très gros travail ... et il est peu reconnu par la communauté universitaire !
- Tenter l'expérience d'une utilisation du logiciel en Deug Mias.

Références

[1] *The XEmacs editor.* `www.xemacs.org`.

[2] L. Damas et R. Milner. Principal type schemes for functional programs. In *Ninth Annual ACM Symposium on the Principles of Programming Languages*, pages 207–212. ACM, 1982.

[3] René David, Karim Nour, et Christophe Raffalli. *Introduction à la logique.* Masson, 2001.

[4] Yannis Delmas-Rigoutsos et René Lalement. *La logique ou l'art de raisonner.* Le pommier, 2000.

[5] Gilles Dowek. *La logique.* Flammarion, 1995.

[6] John Harrison. *HOL Light.* `www.cl.cam.ac.uk/Research/HVG/HOL`.

[7] INRIA. *The Coq Proof Assistant.* `coq.inria.fr`.

[8] Slind Konrad et al. *HOL98.* `www.cl.cam.ac.uk/Research/HVG/HOL`.

[9] J. Strother Moore et al. ACL2. `www.cs.utexas.edu/users/moore/acl2`.

[10] Sam Owre et Natarajan Shankar. *PVS.* `pvs.csl.sri.com`.

[11] Larry Paulson et al. *Isabelle.* `www.cl.cam.ac.uk/Research/HVG/Isabelle`.

[12] Randy Pollack. *LEGO.* `dcs.ed.ac.uk/home/lego`.

[13] Christophe Raffalli. *PhoX.* `www.lama.univ-savoie.fr/~RAFFALLI/phox.html`.

[14] Kleymann Thomas, David Aspinall, et al. *Proof General.* `zermelo.dcs.ed.ac.uk/~proofgen`.