数理解析研究所講究録
第1909巻 2014年 165-175

数学 e-ラーニングのための問題バンクの構築

[1] 北里大学一般教育部 谷口 哲也 (Tetsuya Taniguchi)
[2] 名古屋大学大学院情報科学研究科 中村 泰之 (Yasuyuki Nakamura)
[3] 三玄舎 中原 敬広 (Takahiro Nakahara)
[1] tetsuya@kitasato-u.ac.jp [2] nakamura@nagoya-u.jp [3] nakahara@3strings.co.jp

1. はじめに

自然科学系科目の学習を支援するために, ラーニング・マネジメント・システム (LMS) のオンラインシステムを活用する場合, 従来の, 正誤解答方式, 多肢選択解答方式, 数値入力方式だけではなく, 数式で入力された解答の自動採点を行う, 数式入力解答方式が求められる. さらに, 正誤評価だけではなく, 学生の様々な解答に対して適切なフィードバックを与えることが重要である. しかし, それらの問題作成に費やされる時間は決して無視できない. 本文では, 全国の理工系の教員のための, 数式入力解答方式のオンラインテスト (Moodle と STACK を利用) の導入方法と使用法, 並びにそれらの問題を効率的に蓄積し, 共有していくことのできる問題データベースシステム mathbank を紹介する.

2. STACK 3.1 のインストール

STACK 2 のインストールは少々煩雑であったが, STACK 3 以降は Moodle のプラグインとして実現されており, インストールは比較的簡単になり, 実行速度もかなり改善された. ここでは, `https://github.com/maths/moodle-qtype_stack/blob/master/doc/en/Installation/index.md` によるインストール方法 (CentOS 6.4 で検証済) を紹介する.

(1) サーバの `/var/www/html/moodle/` に Moodle 2.5 がインストールされ, `/var/www/moodle/` にデータディレクトリがあり, `http://dokoka.univerity.ac.jp/moodle/` のような形で, Moodle 2.5 にアクセスできるような環境にする.

(2) maxima をインストール.

```
# wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
# rpm -ivh ./epel-release-6-8.noarch.rpm
# yum install maxima.x86_64
```

(3) (スキップしてよい) mathjax をインストール.
`https://github.com/maths/moodle-qtype_stack/blob/master/doc/en/Developer/Mathjax.md` の Option 2 にしたがって, mathjax-MathJax-v2.3-9-g78ea6af.zip をダウンロード.

```
# unzip ./mathjax-MathJax-v2.3-9-g78ea6af.zip
# mv ./mathjax-MathJax-78ea6af/  /var/www/html/moodle/lib/mathjax/
```

(4) いったん, 所有者を変更.

```
# chown -R apache:apache /var/www/html/moodle/
```

(5) https://moodle.org/plugins/ にアクセスし, Select Moodle version: を 2.5, Search plugins の所を STACK にして検索し, 以下の 6 つをクライアントにダウンロード.

```
qbehaviour_dfexplicitvaildate_moodle25_2013071100.zip
qbehaviour_dfcbmexplicitvaildate_moodle25_2013071100.zip
qbehaviour_adaptivemultipart_moodle25_2013071100.zip
qtype_stack_moodle25_2013070900.zip
qformat_stack_moodle25_2013070900.zip
quiz_stack_moodle25_2013070900.zip
```

(6) (スキップしてよい) 日本語化したい人向け. (5) の上から 4 番目をサーバにコピー.

```
# unzip ./qtype_stack_moodle25_2013070900.zip
# cd ./stack/lang/en/
```

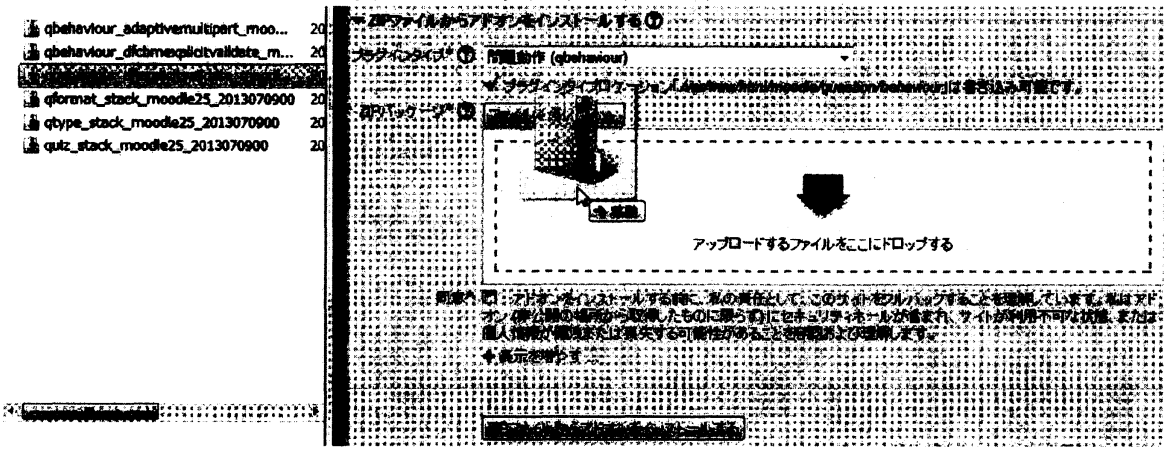
この場所にあるファイル `qtype_stack.php` において, vi や emacs 等で, 英語の箇所を好みの日本語 (UTF-8) に変更. 例えば, 642 行目のところで, 「 Your last answer was interpreted as follows」を「あなたの入力した解答は」に変更.

```
# cd ../../../
# mv ./qtype_stack_moodle25_2013070900.zip ./qtype_stack_moodle25_2013070900.zip.org
# zip -r qtype_stack_moodle25_2013070900.zip ./stack/
```

このあらたな `qtype_stack_moodle25_2013070900.zip` を (5) の クライアント上にある 4 番目のファイルに置き換える.

(7) admin で Moodle にログインし, サイト管理 – プラグイン – アドオンをインストールする に移動し, まず, `qbehaviour_dfexplicitvaildate_moodle25_2013071100.zip` をプラグインタイプを問題動作 (qbehaviour) にして, クライアント上で, ドラッグアンドドロップ (Figure 1). 同意にチェックし, 「ZIP ファイルからアドインをインストール」クリックして, インストール.

FIGURE 1. プラグインインストール



後, 同様に, `qbehaviour_dfcbmexplicitvaildate_moodle25_2013071100.zip` と `qbehaviour_adaptivemultipart_moodle25_2013071100.zip` を問題動作 (qbehaviour) でインストール. 次に, `qtype_stack_moodle25_2013070900.zip` を問

題タイプ (qtype) にして, インストール. このとき, 「新しい設定-STACK」という画面がでるが, なにもせず, 下の方にある「変更を保存する」をクリック.

(8) Moodle の admin でサイト管理 – アピアランス – 追加 HTML に移動し, HEAD タグ内を Figure 2 のように設定. インストール作業 (3) をスキップした場合は,

FIGURE 2. mathjax の設定

HEADタグ内
additionalhtmlhead

```
<script type="text/x-mathjax-config"> MathJax.Hub.Config({
MMLorHTML: { prefer: "HTML" },
tex2jax: {
displayMath: [['\[', '\]']],
inlineMath: [['\(', '\)']],
processEscapes: true
},
TeX: { extensions: ['enclose.js'] }
});
</script>
<script type="text/javascript" src="http://dokoka.university.ac.jp/moodle/lib/mathjax/MathJax.js?config=TeX-AMS_HTML">
</script>
```

`src="http://dokoka.university.ac.jp/moodle/lib/mathjax/MathJax.js?config=TeX-AMS_HTML"` の箇所を
`src="http://cdn.mathjax.org/mathjax/latest/MathJax.js?config=TeX-AMS_HTML"`
に変更しておく.

(9) Moodle の admin で サイト管理 – プラグイン – 問題タイプ – STACK に移動. Platform type を Linux, Maxima version を 5.23.2, CAS connection timeout を念のため, 50 ぐらいに設定 (もっと増やしてもよい), Instant validation にチェック, Maths filter を MathJax に, Replace `$ and $$` にチェック, 3 か所の Standard feed back for ***** のところを以下のように自分の好みに設定,

`Correct answer, well done.` -> 正解です, よくできました.
`Your answer is partially correct.` -> あなたの答は部分的にあっています.
`Incorrect answer.` -> 不正解です.

「変更を保存する」をクリック.
healthcheck script をクリックして実行. 少々時間がかかるが, STACK のテストが実行される. 下の方にある Clear the cache をクリック.

(10) `https://github.com/maths/moodle-qtype_stack/blob//doc/en/CAS/Optimising_Maxima.md` の SBCL の項にしたがって, 次の手順で maxima を最適化する.

`# cd /var/www/moodledata/stack/`
`# maxima` とコマンドを入力し実行. プロンプトが変わる. maxima 上で,
`load("maximalocal.mac");` と入力し, リターンキーを押す. 同じく, maxima 上で,
`:lisp (sb-ext:save-lisp-and-die "maxima-optimised" :toplevel #'run :executable t)`
と入力し, リターンキーを押す. しばらくすると, maxima が終了し,
`/var/www/moodledata/stack/` にサイズが 90Mb から 100 Mb の maxima-optimised というファイルが出来上がる.

(11) Moodle の admin で サイト管理 – プラグイン – 問題タイプ – STACK にもどり, Platform type を Linux (optimised) に変更.
「変更を保存する」をクリック.
healthcheck script をクリックして実行. 今度は STACK のテストが (9) より速く終了する. 下の方にある Clear the cache をクリック.

(12) Moodle の admin で, サイト管理 - プラグイン - アドオンをインストールするに移動し, (7) と同様 に, `qformat_stack_moodle25_2013070900.zip` を, プラグインタイプを 問題インポート/エクスポートフォーマット (qformat) にして, インストール. さらに, `qformat_stack_moodle25_2013070900.zip` を, プラグインタイプを小テスト/レポート (quiz) にして, インストール.

(13) サーバにて, 所有者を元に戻し, インストール完了.

```
# chown -R root:root /var/www/html/moodle/
```

3. STACK3.1 における問題作成例

ここでは, 簡単な足し算の問題を例に挙げながら, STACK3.1 における基本的な問題作成方法を紹介する.

$4+5=?$

簡単な確認問題であるが, このままでは, すでに解き終えた人から得た正答が入力されしまうので, 次のように一部を乱数化したい.

$a+b=?$

---

a, b を乱数にすることにより, 問題が乱数化される. また, 正答を $k1$ とおくと, $k1=a+b$ は乱数 a, b に依存するので, 解答も様々変化することに注目しよう.

STACK ではこのような問題が簡単に作成でき, Moodle と組み合わせることによって, オンライン上で学生に解答させることができる. では実際にその方法を紹介する.

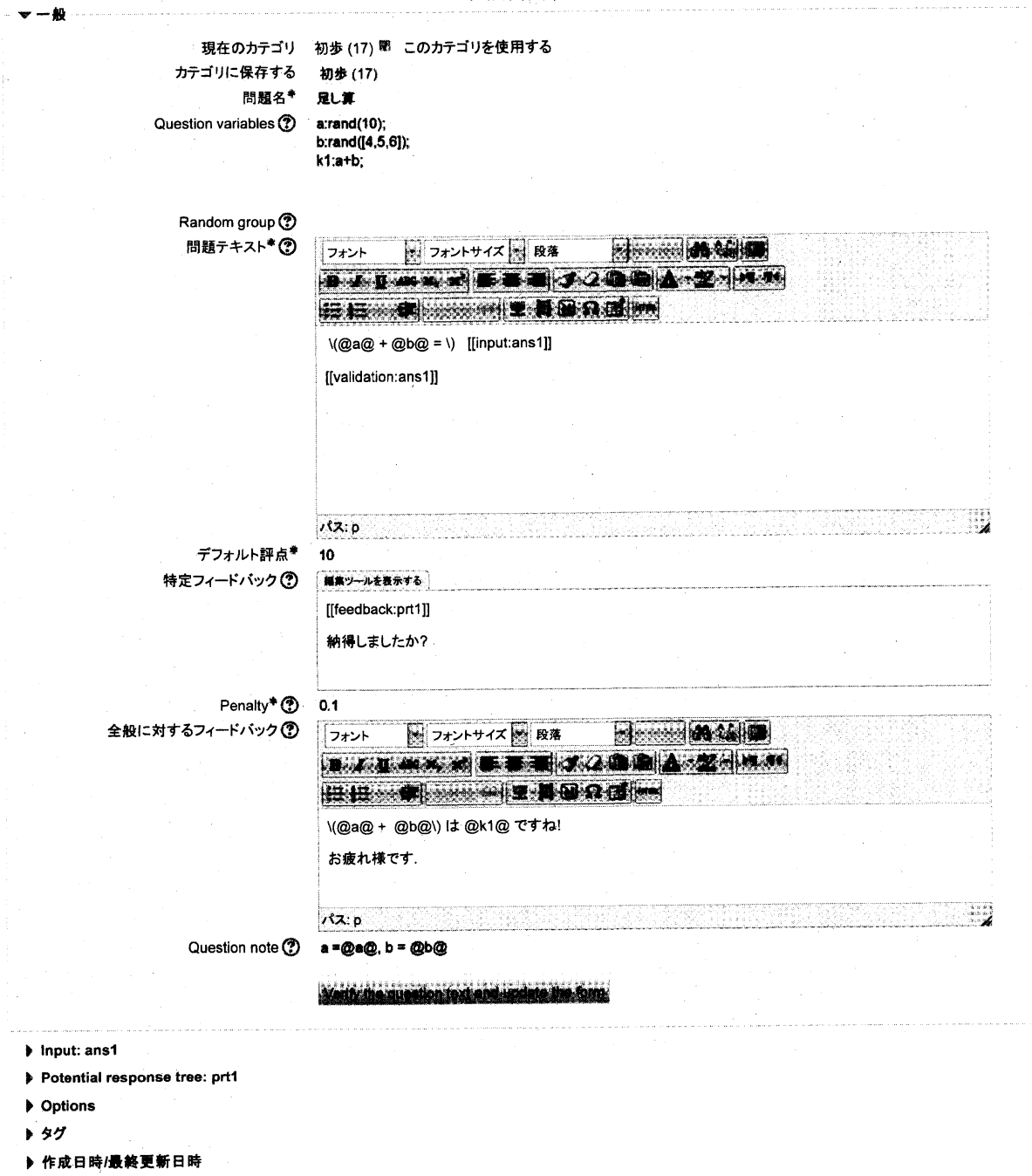
ブラウザから 管理 - コース管理 - 問題バンク - 問題で「あたらしい問題」ボタンをクリックし, 「追加する問題タイプを選択する」において STACK を選択して問題作成画面にいく. すると Figure 3 のように, **一般**, **Input**:ans1, **Potential response tree**:prt1, **Options**, タグ, 作成日時/最終更新日時 の 6 つ項のうち, 一般の項が展開された画面が出現する. 5 つの項のそれぞれについて, クリックするごとに, 展開されたり, 折りたたまれるので, 編集する項だけを展開しておくと編集しやすいであろう.

問題名欄に各自決めた問題名を入力する. **Question variables** の欄では, a:rand(10); と入力することにより, a には 0 から 9 までの整数のいずれが乱数として, 代入される. また, b:rand([4,5,6]); と入力することにより, b には 4, 5, 6 のいずれかが乱数として代入される.

ついでに, Question variables の欄で, k1:a+b; と入力し, 足し算の問題に対する解答 $a+b$ を $k1$ に代入しておく.

次に, **問題テキストの欄**において, 実際, 学生が目にするであろう問題文をここに入力する. ほとんど, TEX と同様な感覚で入力することでができる. mathjax により, 数式は \(と \) で囲えばきれいに出力される. また, \[と \] で囲えば別行立てに出力される. 1 つ注意してもらい点がある. 問題文において, TEX ではみられない, @a@ のような箇所があるが, 実際には @a@ とは出力されず, Question variables の欄で, a として $a=5$

FIGURE 3. 問題編集画面



と乱数が代入されたときは, @a@ は 5 として出力される. また, [[input:ans1]] は解答を入力するための場所を表すものである.

さらに, 問題テキストの欄にある [[validation:ans1]] は 入力された解答を STACK が即時に再表示し, 学生に再確認を促す場所を示す. 例えば, [[input:ans1]] の欄に学生が, CAS の文法に従って, `x^5+6*x +2` と入力すれば, [[validation:ans1]] の箇所に

「Your last answer was interpreted as follows: x^2+6x+2」

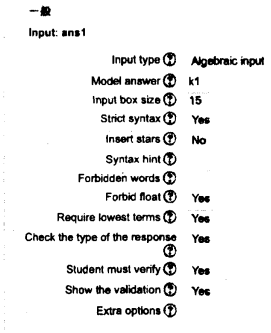
と STACK なりに解釈されたものが表示される. インストール作業 (6) を行った場合は,

「あなたの入力した解答は: x^2+6x+2」

と表示される. CAS の文法に慣れていない学生にとっては有用である. ただ, 解答が 8 や 5/4 のような簡単な数値の場合はかえって, 邪魔になるだけなので, この場合であれば Input: ans1 をクリックし, Figure 4 の Show the validation の項を No にすれば表示されなくなる.

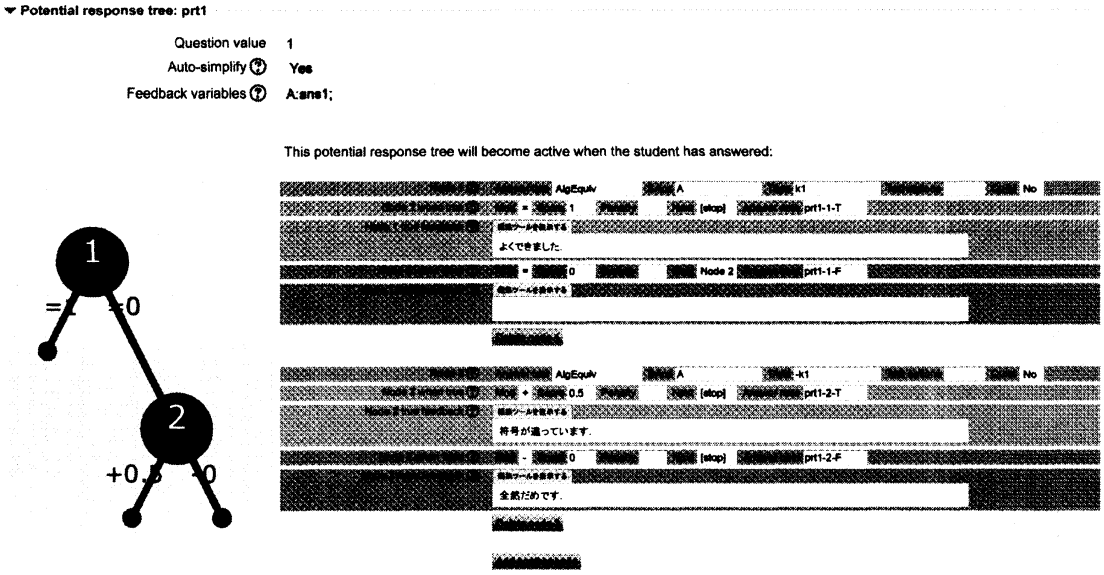
特定フィードバックの欄にある [[feedback:prt1]] は学生が入力した解答に対して, 正解か不正解かあるいは部分的にあっている等のコメントが出力される場所となる. 一般に [[feedback:xxx]] があれば, Figure 3 の「Verify the question text and update the form」ボタンをクリックすると, 必ず, 編集画面の下の方に Potential response tree: xxx が対になって出現する. よってこの場合, Potential response tree: prt1 が下の方に出現しているが, そこをクリックすると Figure 5 のように, [[feedback:prt1]] に出力されるフィードバックの内容と配点方法等が設定されている. これについては, 後に詳しく説明する.

FIGURE 4



次に, 左図の **Input: ans1 の Input type の欄**では [[input:ans1]] に対応する入力形式 **Algebraic Input** を選択し, **Model answer の欄**に正答として, $k1$ を入力する. (もし, 解答が**行列**であれば, Input type は **Matrix** となる.) 実は, Model answer として入力した $k1$ は, 教員プレビュー画面で, 「正解を表示する」をクリックしたときのみ表示されるもので, 学生の目に触れることがない. したがって, 解答の型 (行列であれば, 2x3 等) に注意して, 値に関してはさほど神経質になる必要はない.

FIGURE 5. Potential response tree: prt1



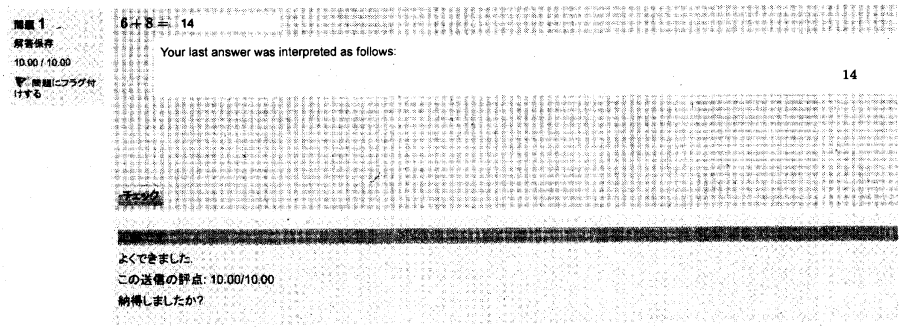
特定フィードバックの欄の説明に戻ろう．この欄に場所に出力されるフィードバックが，どのように設定されるかを見てみよう．まず，学生が check ボタンを押すと，この欄の行頭に **Options** で設定された 3 種類のフィードバックのうちの 1 つが最初に出力される．Options では，インストール作業 (9) での設定内容が引き継がれている．したがって，その問題に対して，満点であれば，Standard feedback for correct の欄で設定された「正解です.」が表示され，0 点であれば，Standard feedback for incorrect の欄で設定された「不正解です.」が表示される．また，満点と 0 点の間であれば，Standard feedback for partially correct の欄で設定された「部分的にあっています.」が表示される．

特定フィードバックの欄では説明文等が入力されていればそのまま出力される．CAS テキストも使用可である．また，[[feedback:xxx]] がある場合は，対応する Potential response tree xxx で設定されたフィードバックが出力される．これについて，Figure 5 の Potential response tree prt1 を例にとり詳しく説明する．

(ケース 1) ans1 に正解である $k1$ の値が入力されたとき．

Node 1 で，ans1 が代入された A と答えである $k1$ が等しければ，Mod = Score 1 となっているので，まず，配点は 1 点に設定され，「よくできました.」というフィードバックが出力される．Next の欄が [stop] なので，そこで終了となる．「この送信評点」は デフォルト評点 × 配点合計/(1 点 × Potential response tree の総数) = $10 \times 1/(1 \times 1) = 10$ 点と計算される．

FIGURE 6. ケース 1

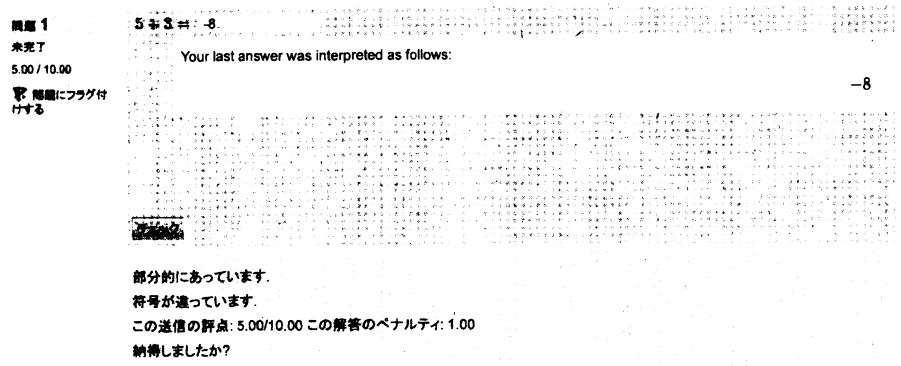


(ケース 2) ans1 に正解である $k1$ の逆符号の値が入力されたとき．

Node 1 で，ans1 が代入された A と答えである $k1$ が等しくなければ，Mod = Score 0 となっているので，まず，配点は 0 点に設定され，フィードバックの欄が空欄なので，なにもが出力されず，Next の欄が Node 2 なので，Node 2 へジャンプする．

Node 2 で，ans1 が代入された A と答えの逆符号である $-k1$ が等しければ，Mod + Score 0.5 となっているので，まず，Node 1 で設定された 0 点に 0.5 点を加えた 0.5 点が新たに配点として設定され，「符号が違っています.」というフィードバックが出力される．Next の欄が [stop] なので，そこで終了となる．「この送信評点」は デフォルト評点 × 配点合計/(1 点 × Potential response tree の総数) = $10 \times 0.5/(1 \times 1) = 5$ 点と計算される．

FIGURE 7. ケース 2



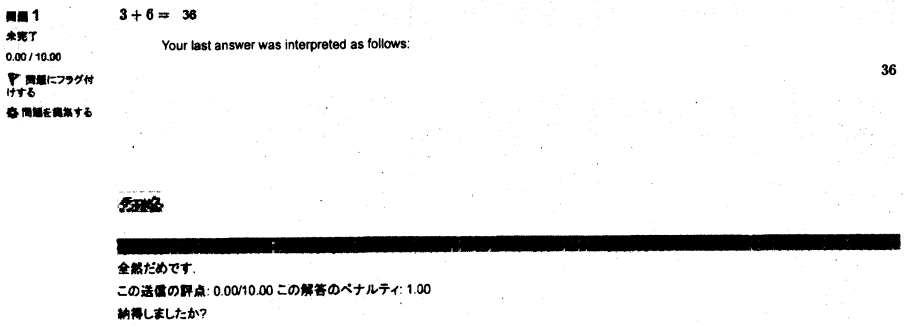
(ケース 3) ケース 1 でも ケース 2 でもないとき.

Node 1 で, ans1 が代入された A と答えである $k1$ が等しくなければ, Mod = Score 0 となっているので, まず, 配点は 0 点に設定され, フィードバックの欄が空欄なので, なにもが出力されず, Next の欄が Node 2 なので, Node 2 へジャンプする.

Node 2 で, ans1 が代入された A と答えの逆符号である $-k1$ が等しくなければ, Mod - Score 0 となっているので, Node 1 で設定された 0 点から 0 点を引いた 0 点が新たに配点され, 「全然だめです.」というフィードバックが出力される.

Next の欄が [stop] なので, そこで終了となる. 「この送信評点」は デフォルト評点 × 配点合計/(1 点 × Potential response tree の総数) $= 10 \times 0/(1 \times 1) = 0$ 点と計算される.

FIGURE 8. ケース 3



実は, 特定フィードバックの欄にある [[feedback:xxx]] を問題文の欄に配置してもよい. ただし, フィードバックの出力の方法が次のように若干違う. 問題文の欄にある [[feedback:xxx]] の箇所ごとに, Options で設定されたフィードバックと対応する Potential response tree xxx で設定されたフィードバックが一緒に出力がされるので注意されたい.

学生は「テスト終了...」ボタンをクリックし, 「すべてを送信して終了する」ボタンをクリックしたあとに, 問題をレビューできるが, そのときに, 特定フィードバック欄と**全般に対するフィードバックの欄**が合わせて表示させる. 例として, ケース 1 のレビューの

画面を Figure 9 に挙げておく. **全般に対するフィードバックの欄**で設定された, 「@a@ + @b@ は @k1@ ですね! お疲れ様です.」がレビュー画面では「6 + 8 は 14 ですね! お疲れ様です.」と表示されている. このように, 学生に解答を与えたいときは, この欄に解答を配置しておけばよいだろう.

FIGURE 9. レビュー画面

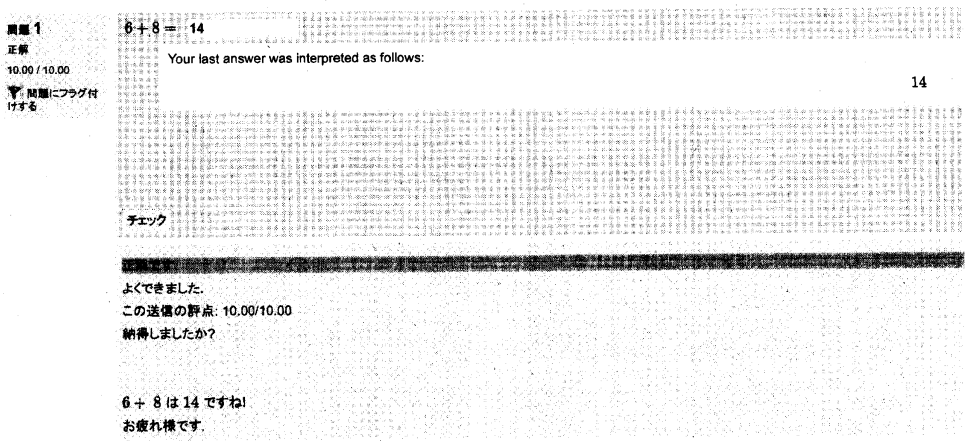
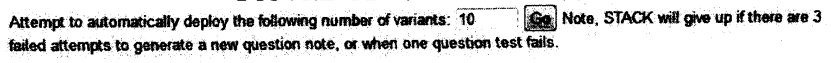
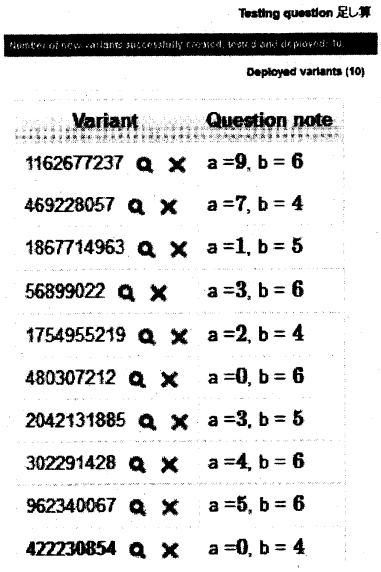


FIGURE 10. デプロイ 手順 1

FIGURE 11. デプロイ 手順 2





Question note の欄は, あらかじめ, a, b にランダムな数を発生させ, 前もって問題をキャッシュに入れ (**デプロイ**) 高速化を図るための設定欄である. 動作が遅く感じられたら実行するとよいだろう. あらかじめ, 10 個作っておきたい場合は, 問題プレビュー画面で, ポインターの差している所をクッリクし (手順 1), 欄に 10 を入力して, Go をクリックすれば (手順 2), 左のように問題がキャッシュに貯められる. あまり数を多くすると重複するが自動的に削除される.

4. mathbank へのアップロード

では, この足し算の問題を http://mathbank.jp/ へアップロードしてみよう.

(1) http://mathbank.jp/ にアクセス. ユーザ名とパスワードを入力してログイン. 緑色の登録ボタンをクリックすると Figure 13 のような画面がでる.

FIGURE 12. mathbank ログイン画面

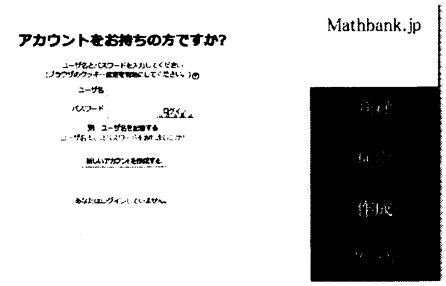
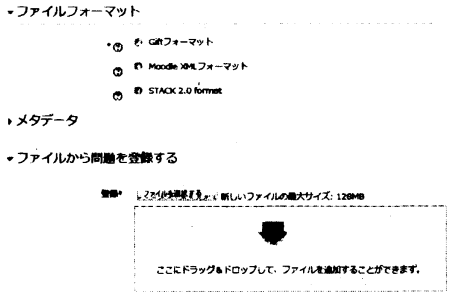


FIGURE 13. メタデータ



(2) ファイルフォーマットを Moodle XML フォーマットにし (STACK 2 の問題のときは STACK2.0 フォーマット), メタデータを適宜設定して, エクスポートしておいた tashizan.xml をドラッグアンドドロップ. 登録ボタンをクリックすると登録完了となる.

FIGURE 14. 登録画面

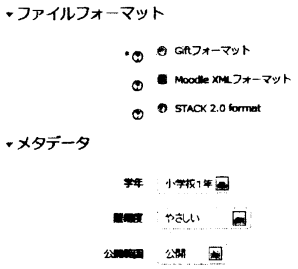
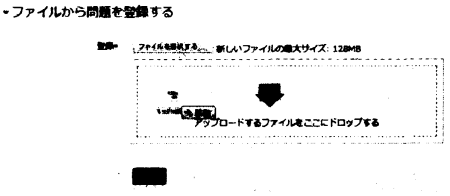


FIGURE 15. メタデータ



(3) ためしに, 検索ボタンをクリックして, 検索してみると, 無事に登録されていることがわかる.

FIGURE 16. 検索画面

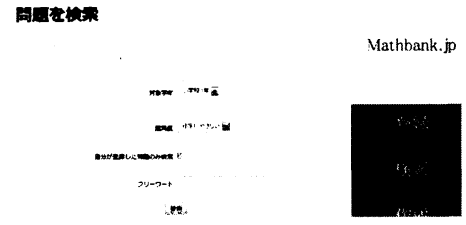
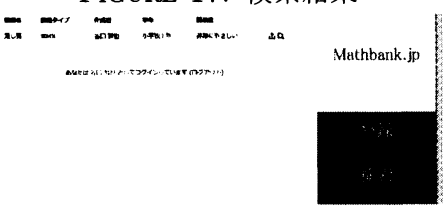


FIGURE 17. 検索結果



5. mathbank からのダウンロード

今度は, http://mathbank.jp/ から何か問題をダウンロードしてみよう.

(1) http://mathbank.jp/ にアクセス. ユーザ名とパスワードを入力してログイン. 青色の検索ボタンをクリックし, 大学の問題で任意の難易度の問題を検索する.

FIGURE 18. mathbank ログイン画面

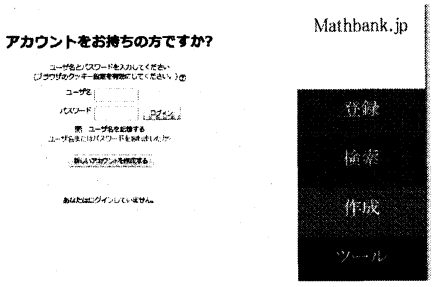
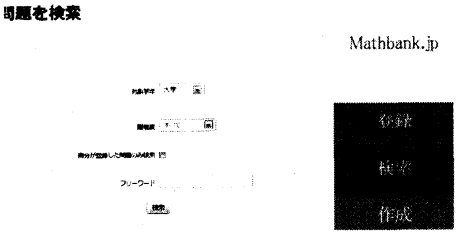


FIGURE 19. 検索画面



(2) 検索結果の連立方程式-1 の問題をダウンロードすることに決めたら, Figure 21 のように対象の問題の所にポインタを合わせクリックすると quiz-mathbank-20131109-1742.xml のようなファイル名でダウンロードされる.

FIGURE 20. 検索結果

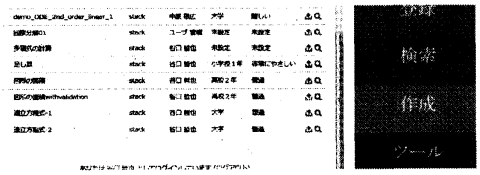
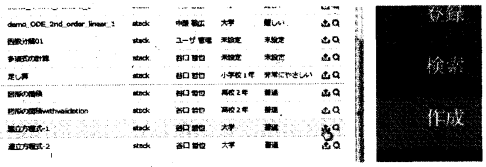


FIGURE 21. 選択



(3) あとは, 各自のサーバの Moodle において, 管理 – コース管理 – 問題バンク – インポートに移動し, Moodle XML フォーマットでインポートする. ためしに, プレビューしてみると, 連立方程式の問題が, 無事出力される.

FIGURE 22. 検索画面

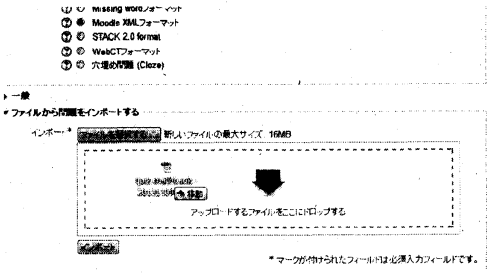


FIGURE 23. 検索結果