УДК 681.3: 658.56.

М.С. Львов

Математические тесты в системах компьютерной математики учебного назначения

Рассмотрено понятие математического теста, определены его типы и изучены алгоритмические аспекты разработки модулей тестирования процедурных знаний в системах компьютерной математики учебного назначения.

The concept of mathematical test is considered, the types of mathematical tests are defined and the algorithmic aspects of the development of the modules of testing the procedural knowledge's in the systems of computer mathematics of educational purpose are considered.

Розглянуто поняття математичного тесту, визначено його типи та вивчено алгоритмічні аспекти розробки модулів тестування процедурних знань у системах комп'ютерної математики навчального призначення.

Введение. Процесс обучения точным дисциплинам включает в себя не только лекции, но и активные формы обучения: практические занятия, лабораторные работы, производственную практику и т.д. Таким образом, контролировать необходимо не только декларативные знания, но и *процедурные знания – знания методов решения задач*. Технологии контроля процедурных знаний исследованы и разработаны еще недостаточно. Математические тесты, о которых будет идти речь, предназначены для контроля процедурных знаний. Таким образом, проблема исследования актуальна. Предположительно система тестирования, содержащая математические тесты, реализована в модуле *Среда тестирования* системы компьютерной математики учебного назначения (СКМУН) [1–5].

Проблему исследования статьи можно сформулировать как *исследование функциональных требований, математических моделей и алгоритмов построения системы тестирования процедурных знаний в СКМУН.*

Формальное определение предметной области

Предметную область (онтологию) СКМУН представляют структурно-логические схемы (СЛС). Эти онтологии представлены трехуровневой иерархией «*специальность*» – «*дисциплина*» – «*учебный модуль*».

Ключевые слова: компьютерная математика, информационные системы учебного назначения, математические тесты.

Наш подход заключается в том, что система тестовых заданий должна быть единой в рамках учебной дисциплины, а еще лучше – в рамках специальности. Однако ее разработка осуществляется поэтапно. Таким образом, в онтологии определяется:

- Учебный модуль (УМ) – минимальная предметная область, для которой необходима разработка системы тестовых заданий как единого этапа работ построения общей системы тестирования.
- Логически необходимая последовательность расширений системы тестовых заданий как этапов этих работ.

Модель учебного модуля математической дисциплины

Сигнатура УМ. Математические теории, излагаемые в УМ, используют, как правило, новые математические символы. Например, модуль *Тригонометрия* вводит символы тригонометрических и обратных тригонометрических функций, символ константы π. Список этих символов составляет собственную сигнатуру УМ. Предмет изучения – формальные определения и содержательные интерпретации символов сигнатуры. Методы проектирования сигнатур описаны в [6, 7].

Математические модели УМ. Каждый учебный модуль определяется списком математических объектов (моделей), являющихся предметом изучения. В модуле *Тригонометрия* – это формальные определения тригонометрического выражения, тождества, уравнения. В модуле

Электричество изучаются модели электростатического поля, участка цепи постоянного тока, электрической схемы постоянного тока.

Элементарные преобразования моделей УМ. В УМ определены так называемые элементарные операции – преобразования (ЭП) математических объектов этого модуля. ЭП определяется аргументами и результатами. Например, собственные ЭП для модуля *Дифференциальное исчисление* определяется правилами дифференцирования и таблицей производных элементарных функций. Методы проектирования элементарных преобразований описаны в [2, 8].

Модели учебных задач УМ. Основной предмет изучения УМ математической дисциплины – учебные задачи, перечень типов которых определен учебной программой дисциплины. Формулировка учебной задачи использует одну или несколько базовых математических моделей, объединенных соотношениями. Это – модель учебной задачи. Формальные определения стандартных задач включают модель задачи $M(x_1, \ldots, x_n)$, условие задачи $\varphi(x_1, \ldots, x_n)$ и вопрос $Q(x_{j_1}, \ldots, x_{j_m})$:

Дано $M(x_1, \ldots, x_n)$, *причем* $\varphi(x_1, \ldots, x_n)$. *Найти* $Q(x_{j_1}, \ldots, x_{j_m})$.

Например, задача «*Построить касательную L к графику F функции* $y = \dfrac{x+1}{x}$ *в точке A с абсциссой* $x_A = 1$» представлена в виде модели $P = < M, \varphi, Q >$, где

$$M = F(y = f(x) \,\&\, A(x_A, y_A) \,\&\, \&L(y - y_A = f'(x_A)(x - x_A)), \quad (1)$$

$$\varphi = (f(x) = \frac{x+1}{x}) \,\&\, (x_A = 1),$$

$$Q = L.$$

Модель задачи использует базовые модели *графика функции* $y = f(x)$, *точки* $A(x_A, y_A)$ и *касательной к графику функции* $L(y - y_A = f'(x_A) \times (x - x_A))$. Эти модели объединены соотношениями $\left(f(x) = (x+1)/x\right) \,\&\, (x_A = 1)$, определенными в условии задачи.

Точные определения математических объектов и учебных задач задаются в формальных спецификациях каждого конкретного учебного модуля.

Область применения моделей УМ. Анализ СЛС таких математических дисциплин, как *школьная алгебра*, *математический анализ*, *линейная алгебра*, *теория обычных дифференциальных уравнений*, некоторых дисциплин прикладной математики, показал, что все они удовлетворяют этой схеме. Соответствующая СКМУН, а следовательно, и система тестирования, может строиться как единая система, основанная на понятии *алгебраического объекта* (АО) и *элементарных алгебраических преобразований*.

В статье рассматриваются проблемы построения систем тестирования алгебраических учебных модулей. Системы тестирования, основанные на результатах этого исследования, реализованы в СКМУН «Терм» [10]. Специфические проблемы, возникающие при реализации систем тестирования в других предметных областях, – предмет дальнейших исследований.

Тестовые задания в учебных модулях

Классом тестовых заданий называется класс учебных задач $P = < M, \Phi, Q >$, в котором определена модель $M(x_1, \ldots, x_n)$, зависящая от переменных $x_1, \ldots, x_n$, множество условий, заданное конечным набором логических формул $\Phi = < \Phi_1, \ldots, \Phi_k >$ и конечным набором вопросов $Q = < Q_1, \ldots, Q_m >$. Конкретное тестовое задание – элемент класса $P = < M, \Phi, Q >$ имеет вид $P_{ij} = < M, \varphi_i, Q_j >$.

Замечание. Когда речь идет о тестах, вопросы к задачам принято называть ответами.

Пример 1. Квадратные уравнения с целыми коэффициентами

$M(a,b,c,x) = \{ax^2 + bx + c = 0\}$, $\Phi = < \Phi_1, \Phi_2, \Phi_3 >$, где

$\Phi_1 = (a = 1) \& (b \in Int) \& (c \in Int) \& (b^2 - 4ac > 0)$,

$\Phi_2 = (a \in Int) \& (b \in Int) \& (c \in Int) \& (b^2 - 4ac = 0)$,

$\Phi_3 = (a = c) \& (b \in Int) \& (b^2 - 4ac = 0)$,

$Q = < x >$.

Этот класс тестовых заданий определяет задания типа *Решить квадратное уравнение с целыми коэффициентами*. Первая серия класса (с условием Φ_1) задает приведенные квадратные уравнения с двумя корнями, вторая серия – квадратные уравнения с равными корнями, а третья – квадратные уравнения, где коэффициенты *a* и *c* равны.

Модели учебных задач могут содержать не только выражения с буквенными коэффициентами, но и формальные определения элементов моделей. Модель M примера 1 можно обобщить определением $X \in Var$, где $Var = \{a,b,...,z\}$. Тогда

$$M(a,b,c,X) = \{aX^2 + bX + c = 0\} \,\&\, (X \in Var).$$

Пусть $\Psi = (X \in \{u,v,w,x,y,z\})$. Тогда Φ_1, Φ_2, Φ_3 можно доопределить

$$\Phi_1 := \Psi \,\&\, \Phi_1,\ \Phi_2 := \Psi \,\&\, \Phi_2,\ \Phi_3 := \Psi \,\&\, \Phi_3.$$

Класс тестовых заданий $P = <M, \Phi_1, Q_2>$ интерпретируется как *Поиск суммы квадратов корней приведенного квадратного уравнения с целыми коэффициентами*. Конкретное тестовое задание – элемент этого класса формулируется как *квадратное уравнение* $au^2 - 5u + 3$, *причем* $a = 1$, $25 - 12a > 0$. *Найти* $u_1^2 + u_2^2$.

Описания и алгоритмы генерации тестовых заданий

В принципе существует два подхода к решению задачи генерации конкретных тестовых заданий. Во-первых, можно хранить серии однотипных тестовых заданий в базе данных. Во-вторых, реализовать в виде системных процедур алгоритмы автоматической генерации однотипных тестовых заданий. Каждый из этих подходов имеет свои преимущества и недостатки. В первом случае необходимо тратить много времени на наполнение базы данных, и тестовые задания могут повторяться. Во втором случае время тратится на реализацию алгоритмов автоматической генерации, но каждое тестовое задание индивидуально. Взвешенный подход заключается в том, чтобы:

- на основе алгоритмического анализа моделей и задач каждой конкретной предметной области для каждого достаточно широкого класса тестовых заданий разрабатывать единую общую модель этого класса, а также модели и алгоритмы генерации условий и ответов к этой модели;
- разработать общие *CASE*-технологии описания подклассов тестовых заданий на основе единой общей модели пользователями СКМУН;
- разработать общие механизмы хранения и вызова алгоритмов генерации конкретных тестовых заданий.

Описание модели тестового задания содержит описание типа математического объекта и типов его параметров. Авторы ограничились следующими типами алгебраических объектов:

Атомарные АО: числа, логические значения, переменные.

Числа: полукольцо *Nat*, кольцо *Int*, поле *Rat*.

Логические значения: *Bool* = (*False*, *True*).

Переменные: малые и большие латинские буквы и эти буквы с (натуральными) индексами. Множество переменных обозначим через Var.

Выражения: термы от атомарных АО в сигнатуре алгебраических операций модуля Σ. В иерархии алгебраических учебных модулей сигнатуры АО расширяются. В алгебре изучаются: *линейные выражения*, *мономы*, *целые выражения*, *рациональные*, *радикальные и модульные выражения.*

Атомарные предикаты: $A = B$, $A \neq B$, $A < B$, $A > B$, $A \leq B$, $A \geq B$. В иерархии алгебраических учебных модулей сигнатуры предикатов АО расширяются. В алгебре также изучаются: *уравнения*, *неравенства*, *тождества*.

Логические алгебраические объекты – конъюнкции и дизъюнкции атомарных предикатов (в алгебраической терминологии – системы и совокупности). Кроме того, в алгебре изучаются *системы уравнений и неравенств.*

Тестовые задания-шаблоны. Выражение $F(x_1, \ldots, x_m; a_1, \ldots, a_k)$ в сигнатуре $\sum_{SD}$ данной предметной области SD назовем выражением-шаблоном (*Expression Template*).

Переменные $x_1, \ldots, x_m$ назовем метапеременными. Общей областью значений метапеременных является множество Var ($x_j \in Var$). Переменные $a_1, \ldots, a_k$ назовем параметрами. Область значений параметров – числовые множества ($a_j \in Num$).

Пример 2. $a_1x_1 + a_2x_2$ – выражение-шаблон. Конкретные экземпляры этого шаблона: $2a + 3x$, $-2/3u + 5x$, и т.д. Если $F(x_1, \ldots, x_m; a_1, \ldots, a_k)$ – выражение-шаблон, то

$$Sub^{v_1,\ldots,v_m}_{x_1,\ldots x_m} Sub^{c_1,\ldots,c_m}_{a_1,\ldots a_k} F(x_1,\ldots,x_m;a_1,\ldots,a_k) =$$

$$= F(u_1, \ldots, u_m; c_1, \ldots, c_k),\ x_j \in Var,\ a_j \in Num$$

назовем специализацией (или частным случаем, или экземпляром) $F(x_1, \ldots, x_m; a_1, \ldots, a_k)$.

Выражения-шаблоны обозначают как общий вид модели, так и общий вид ответа тестового задания. Таким образом, основа определения тестового задания T – пара $<F_{Task}, F_{Ans}>$, где F_{Task}, F_{Ans} – выражения-шаблоны, определенные над общими списками метапеременных и параметров и обозначающие соответственно модель и ответ тестового задания (вопрос соответствующей задачи). Обозначим эти списки через X_{var}, A_{Coef}. Тогда

$$T(X_{\text{var}}, A_{Coef}) = < F_{Task}(X_{\text{var}}, A_{Coef}), F_{Ans}(X_{\text{var}}, A_{Coef}) > .$$

В тестах на эквивалентные преобразования выражений $F_{Task} =_{SD} F_{Ans}$, где «$=_{SD}$» обозначает семантическое равенство в области SD.

Пример 3. Тест на перемножение степеней.

$$T(x, y; m_1, n_1, m_2, n_2) =$$
$$= < x^{m_1} y^{n_1} \cdot x^{m_2} y^{n_2},\ x^{m_1+m_2} y^{n_1+n_2} > .$$

Кроме сигнатуры предметной области, в определении шаблона тестового задания используется системная функция $Val(F)$, определенная на множестве выражений данной предметной области. Системный интерпретатор функции $Val(F)$ вычисляет специальную каноническую форму выражения F. Функция $Val(F)$, как будет показано ниже, используется в алгоритмах генерации и проверки тестового задания.

Пример 4. Тест на приведение подобных. Шаблон условия тестового задания $F_{Task} = (a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4)$ при специализации метапеременных $x_1, x_2, x_3, x_4 \in \{u, v\}$ задает линейное выражение от u, v. Ответ – приведенное линейное выражение $c_1u + c_2v$. Условие $x_1, x_2, x_3, x_4 \in \{u, v\}$ интерпретируется *подстановкой случайным образом вместо каждой метапеременной x_j одной из переменных u, v*. Поскольку значения c_1, c_2 определить невозможно, ответ описан в виде $F_{Ans} = Val(a_1x_1 + a_2x_2 + a_3x_3 + a_4x_3)$ или $F_{Ans} = Val(F_{Task})$.

Такая ситуация является общей: в тестах на вычисление значений, упрощение или приведение к стандартному виду выражений шаблон имеет вид $<F_{Task}, Val(F_{Task})>$.

Пример 5. Тест на факторизацию. В простом частном случае шаблон тестового задания имеет вид $T = < acxy + adx + bcy + bd,\ (ax + b) \times (cy + d) >$, $a, b, c, d \in Int$, $x, y \in Var$. В общем виде этот тест можно описать шаблоном $T = < Val(F \cdot G), F \cdot G >$.

Описания переменных и коэффициентов. Область значений метапеременных – множество Var. Описание метапеременных удовлетворяет синтаксическим правилам:

$$VarDescription ::= < VarList > \in < VarType > . \mid$$
$$\mid VarDescription; < VarList > \in < VarType > .$$

$VarList$ – список переменных через запятую.

$$VarType ::= Var \mid [VarID \ldots VarID] \mid VarID \mid$$
$$\{VarSet\} \mid VarType \cup VarType .$$

Var – базовое множество переменных;

$VarID$ – одна буква (возможно, с индексом);

$[VarID \ldots VarID]$ – отрезок типа Var. Начальное и конечное значения отрезка – либо разные буквы без индексов, либо разные буквы с одним и тем же индексом, либо одна и та же буква с разными индексами.

$\{VarSet\}$ – множество переменных, перечисляемых через запятую.

$VarType \cup VarType$ – объединение двух описаний типов.

Переменные $VarList$ (т.е. левой части определения) – это метапеременные. Переменные, входящие только в $VarType$, – экземпляры. В шаблоне могут использоваться как метапеременные, так и экземпляры переменных. Например:

$X, Y \in Var$ – метапеременные, определенные в Var.

$A,B,C \in [a \ldots d] \cup [u \ldots z]$ – метапеременные, принимающие значения a,b,c,d,u,v,w,x,y,z.

$A \in [a \ldots d]$; $B \in [u \ldots z]$. Метапеременные A и B имеют разные области определений.

$F = ax + by + c$; $a,b,c \in [A \ldots R]$. a,b,c – метапеременные, x, y – экземпляры переменных.

$a_1, a_2, a_3, a_4, a_5, a_6 \in \{u, v\}$; $u, v \in [a \ldots z]$ – метапеременные, принимающие значения u и v, которые в свою очередь принимают в качестве значений малые латинские буквы.

Общие области значений параметров – алгебры $Nat \subset Int \subset Rat$. Описания параметров типов Nat, Int удовлетворяют тем же синтаксическим правилам с соответствующей интерпретацией. Значения типа Rat определяются через отрезки значений их числителей и знаменателей:

$$r \in Rat,\ Num(r) \in [\text{Min}Num, \text{Max}Num],$$
$$Den(r) \in [\text{Min}Den, \text{Max}Den]$$

с ограничениями по умолчанию:

$\mathrm{Min}Num \le \mathrm{Max}Num$, $\mathrm{Min}Den \le \mathrm{Max}Den$, $Num(r) \in$
$\in Int$, $Den(r) \in Nat$, $GCD(Num(r), Den(r)) = 1$.

Описания переменных шаблонов используются в процедуре генерации экземпляров тестов. В простейшем случае процедура *GetTest* имеет спецификацию

$$TaskTest := GetTest(TemplateTest\ T),$$

где $T = < F_{Task}, F_{Ans} >$.

Описание и использование условий в шаблонах. В общем случае в структуру шаблона тестового задания включается шаблон условия Φ, а шаблон F_{Ans} может содержать несколько дизъюнктивных членов: $F_{Ans} = F_{Ans\_1} \vee \ldots \vee F_{Ans\_k}$. Рассмотрим пример:

Пример 6. Применить формулу сокращенного умножения: $T = <(a+b)^2, a^2 + 2ab + b^2>$.

Пусть пользователь описал следующие шаблоны этого тестового задания:

$T_1 = <(X+Y)^2, a^2 \cdot X^2 + 2 \cdot a \cdot X \cdot Y + y^2>$,
$T_2 = <(a \cdot X + b)^2, a^2 \cdot X^2 + 2 \cdot a \cdot b \cdot X + b^2>$,
$T_3 = <(a \cdot X + b \cdot Y)^2, a^2 \cdot X^2 + 2 \cdot a \cdot b \cdot X \cdot Y + b^2 \cdot Y^2>$,
$T_4 = <(a \cdot X^m + b \cdot Y^n)^2, a^{2 \cdot m} \cdot X^{2 \cdot m} + 2 \cdot a^m \cdot b^n \cdot X^m \cdot$
$\cdot Y^n + b^{2 \cdot n} \cdot Y^{2 \cdot n}>$.

Здесь X, Y– метапеременные, a,b,m,n – параметры. Использование условий приводит и к сокращению числа шаблонов, и к расширению класса тестовых заданий. Так, задание T_4 в сочетании с условиями $a = 1$, $b = 1$, $m = 0 \vee m = 1$, $n = 0 \vee n = 1$ описывает все классы тестовых заданий с шаблонами T_1, T_2, T_3. Описание условий следует синтаксису

$$CondList ::= Cond; | Cond, CondList;$$
$$Cond ::= AtomCond \mid AtomCond\ \& \&(Cond) \mid AtomCond \vee (Cond);$$
$$AtomCond ::= VarID = Const \mid VarID = VarID.$$

В описании условия мы ограничились условиями типа простейших равенств. Однако эти определения можно расширять. В рассматриваемом примере условие имеет вид

$$\Phi_{Cond} = (a = 1, b = 1, m = 0 \vee m = 1, n = 0 \vee n = 1).$$

Семантика описания условия состоит в следующем:

- каждое атомарное условие разбивает множество тестов на два класса: класс случаев, в которых оно выполняется, и класс случаев, в которых оно не выполняется. Соответствующее значение генерируется случайным образом из диапазона значений, определенных описаниями параметров: $VarID := \mathrm{Const} \mid VarID := Random()$;
- через запятую перечисляются независимые условия, всевозможные сочетания которых определяют все частные случаи;
- конъюнкция объединяет независимые условия, описывающие один и тот же частный случай;
- дизъюнкция объединяет зависимые условия, каждое из которых определяет разные частные случаи.

Алгоритм генерации тестов в каждом своем вызове возвращает один частный случай. Поэтому условие приводится к каноническому виду $\Phi_{Cond} = \Phi_1 + \Phi_2 + \ldots + \Phi_N$, где Φ_j определяет лишь один тест. Алгоритм приведения Φ_{Cond} рассмотрим на нашем примере: введем логические переменные, соответствующие атомарным условиям:

$A = (a = 1)$, $B = (b = 1)$, $M_0 = (m = 0)$,
$M_1 = (m = 1)$, $N_0 = (n = 0)$, $N_1 = (n = 1)$,

Тогда всевозможные варианты тестов описываются полиномом Жегалкина

$$G = (A + 1)(B + 1)(M_0 + M_1 + 1)(N_0 + N_1 + 1),$$

каждый моном которого описывает один тест. Поэтому алгоритм приведения Φ_{Cond} к каноническому виду заключается в построении стандартного вида полинома G:

$$G = ABM_0N_0 + ABM_0N_1 + \ldots + A + B + M_0 + M_1 + N_0 + N_1 + 1.$$

Представление множества тестов данного шаблона полиномом G используется в классификации тестов по степени их вычислительной сложности. Сложность данного теста определяется степенью (количеством логических переменных) монома: чем меньше степень, тем сложнее вариант теста. Наиболее простые тесты в нашем примере – тесты ABM_0N_0, ASM_0N_1, ABM_1N_0, ASM_1N_1. Самый сложный тест – это первый, в котором все параметры генерируются случайным образом.

Для управления вычислительной сложностью тестов в процедуре тестирования нужно построить гистограмму распределения тестов данно-

го шаблона по их сложности. С этой целью рассмотрим производящую функцию шаблона $g_T(x)$, построенную по условию Φ_{Cond}: $g(x) = G(x, \dots, x)$ [11–12]. Подставим в G вместо всех логических переменных переменную x: $g(x) = (x+1)(x+1)(x+x+1)(x+x+1) = (x+1)^2 \times (2x+1)^2$. Тогда распределение множества тестов шаблона T равно вектору коэффициентов $g(x)$:

$$g(x) = 4x^4 + 12x^3 + 13x^2 + 6x + 1.$$

Таблица 1. Гистограмма распределения сложностей тестов в тестовом шаблоне T_4

Сложность i	1	2	3	4	5
Кол-во тестов N_i	4	12	13	6	1

Функцию G представим в виде $G = G_1 + G_2 + G_3 + G_4 + G_5$, где полином G_i определяет все тесты одной сложности i. Пусть p_i – заданная вероятность генерации теста сложности i. Тогда выбор одного из полиномов G_1, G_2, G_3, G_4, G_5 проводится с вероятностью p_i, а выбор конкретного теста из G_i – с вероятностью $1/N_i$.

Для упрощения шаблона T при условии $U(M_{ij})$, определенном мономом M_{ij} полинома G_i, используется системная функция Red, возвращающая упрощенный шаблон T_{ij}: $T_{ij} := \mathrm{Re}d(T, M_{ij})$. В нашем примере для частного случая ABM_0 получаем

$$T_4 = < (aX^m + bY^n)^2,\ a^{2\cdot m}X^{2\cdot m} + 2a^m b^n X^m Y^n + b^{2\cdot n}Y^{2\cdot n} >;\quad U(ABM_0) = (a = 1 \,\&\, b = 1 \,\&\, m = 0).$$

Подставим эти значения в T_4:

$$T_4' = < (1\cdot X^0 + 1\cdot Y^n)^2,\ 1^{2\cdot 0}X^{2\cdot 0} + 2\cdot 1^0 \cdot 1^n \cdot X^0 \cdot Y^n + 1^{2\cdot n}\cdot Y^{2\cdot n} > .$$

Функция Red упрощает T_4' до шаблона $T_4'' = < (1+Y^n)^2, 1 + 2\cdot Y^n + Y^{2\cdot n} >$. Осталось применить процедуру подстановки конкретного значения метапеременной Y.

В общем случае функция *CetTest* имеет спецификацию

$$GetTest : (TemplateTest, Condition) \to TaskTest .$$

Семантика процедуры *CetTest* (T, Φ_{Cond}), генерирующей экземпляр тестового задания, определенного шаблоном и условием:

- выбрать случайным образом, в соответствии с заданным распределением вероятностей, частный случай – множество G_i условий шаблона T;
- из G_i выбрать случайным образом моном M_{ij};
- вычислить шаблон $T_{ij} = \mathrm{Re}d(T \,\&\, M_{ij})$, упростив T по условию M_{ij};
- выбрать случайным образом и подставить в T_{ij} вместо каждой метапеременной x_j одну из объектных переменных, перечисленных в описании метапеременных;
- выбрать случайным образом и подставить в полученный шаблон T_{ij} вместо каждого параметра a_j одно из его значений, перечисленных в описании параметров;
- в полученном экземпляре вычислить функцию *Val* во всех ее вхождениях;
- возвратить результат в виде $P = < F_{Task}, F_{Ans} >$.

Типы и алгоритм проверки математических тестовых заданий

В соответствии с определением УМ математические тесты можно классифицировать как тесты:

- на знание элементарных преобразований;
- на знание методов решения учебных задач;
- на составление математических моделей.

Тесты на знания элементарных преобразований. Решая тест, пользователь самостоятельно выполняет предложенное в тесте элементарное преобразование. Дидактическая цель – формирование навыков правильного выполнения ЭП, необходимых для решения учебных задач. Тесты этого класса не должны быть слишком сложными. Ученик должен выполнить все вычисления «в уме», записав сразу правильный ответ.

Большинство ЭП выражений эквивалентны в данной предметной области. Поэтому и тесты часто носят двухсторонний характер.

Пример 7. Тест на знание тригонометрической формулы:

– Применить формулу синуса суммы

$$T = <\mathrm{Sin}(2x+y),\ \mathrm{Sin}(2x)\cdot\mathrm{Cos}(y) + \mathrm{Cos}(2x)\cdot\mathrm{Sin}(y)>.$$

– Применить формулу синуса суммы

$$T = <\mathrm{Sin}(2x)\cdot\mathrm{Cos}(y) + \mathrm{Cos}(2x)\cdot\mathrm{Sin}(y),\ \mathrm{Sin}(2x+y)>.$$

Тесты на знания методов решения учебных задач содержат задания, для решения ко-

торых нужно использовать несколько разных элементарных преобразований.

Пример 8. Упростить выражение

$$T =< Cos^4(x) - Sin^4(x), Cos(2x) > .$$

Если системная функция *Val* решает эту задачу (т.е. правильно упрощает выражения), тестовое задание можно записать в виде

$$T =< Cos^4(x) - Sin^4(x), Val(Task) > .$$

Пример 9. Вычислить значение выражения при данных значениях переменных

$$T =< \begin{cases} E = \dfrac{a^2 + 2ab + b^2}{a + b} \\ b = -4 \end{cases}, Val(Task, E > .$$

Тесты этого класса имеют, как правило, большую вычислительную сложность, чем тесты на элементарные преобразования. Поэтому учету подлежит количество шагов, выполненных пользователем в процессе выполнения теста.

Тесты на составление математических моделей содержат текстовые условия учебных задач, решение которых включает дидактически важный этап составления математической модели задачи.

Пример 10. Составить модель задачи:

Задание	Ответ
Поезд на середине пути между станциями *A* и *B* задержался на 15 мин. Чтобы прибыть на станцию B вовремя, машинист увеличил скорость движения на 10 км/час. Найти скорость поезда *v*, если расстояние между станциями составляет 210 км.	$\frac{105}{V} + \frac{15}{60} + \frac{105}{V+10} = \frac{210}{V}$

Шаблон ответа – уравнение $\frac{S}{2V} + \frac{t}{60} + \frac{S}{2(V+v)} = \frac{S}{V}$, а в шаблоне условия данные обозначены параметрами. Единственная метапеременная V обязательно указывается в условии. Шаблон ответа связывает уравнением переменные S, V, t, v. Это означает, что в тестах типа *Решить задачу*, шаблоном ответа может быть не только *V*, но и любая из остальных переменных.

Задача автоматической проверки тестовых заданий. Пусть $F_{Ans}(x_{j_1},...,x_{j_k})$ – шаблон ответа, $\Phi_{Cond}(x_1, \ldots, x_n)$ – условие. Для проверки правильности ответа система вычисляет ответ $f_{Ans}(u_{j_1},...,u_{j_k})$, являющийся результатом подстановки в $F_{Ans}(x_{j_1},...,x_{j_k})$ значений метапеременных и параметров (процедура *GetTest*). Пусть далее $q_{Ans}(y_1, \ldots, y_l)$ – ответ пользователя. Принципиальное отличие правильности ответа пользователя на математический тест от синтаксической правильности ответа на тест с конструируемым условием заключается в следующем:

- Ответ на тест с конструируемым условием проверяется синтаксическим равенством $f_{Ans}(u_{j_1},...,u_{j_k}) = q_{Ans}(y_1,...,y_l)$.
- Ответ на математический тест проверяется семантическим равенством, т.е. некоторой эквивалентностью $f_{Ans}(u_{j_1},...,u_{j_k}) \sim q_{Ans}(y_1,...,y_l)$. Это означает, что правильный ответ может быть представлен несколькими формулами. Например,

$$\mathrm{Sin}(2x + y) = \mathrm{Sin}(2x) \cdot \mathrm{Cos}(y) + \mathrm{Cos}(2x) \cdot \mathrm{Sin}(y),$$

$$\mathrm{Sin}(2x + y) = \mathrm{Sin}(y) \cdot \mathrm{Cos}(2x) + \mathrm{Cos}(y) \cdot \mathrm{Sin}(2x),$$

$$\mathrm{Sin}(2x + y) = \mathrm{Sin}(x + y) \cdot \mathrm{Sin}(x) + \mathrm{Cos}(x + y) \cdot \mathrm{Cos}(x).$$

Поскольку аргумент $2x + y$ может быть представлен в виде суммы бесконечным числом способов и в ответе можно менять местами сомножители и слагаемые, количество правильных ответов бесконечно.

- При проверке правильности ответа следует отличать неправильный ответ от ответа, представляющего правильный, но промежуточный результат решения. Например: *Составить приведенное квадратное уравнение с корнями* $(x_1 = 4)\&(x_2 = -3)$. Промежуточный результат может иметь вид

$$x^2 - (4 + (-3))x + 4 \cdot (-3) = 0,$$

$$x^2 - (4 - 3)x - 12 = 0.$$

- В тестах на составление моделей правильный ответ пользователя может отличаться от ответа системы несколькими дополнительными переменными. Например, в тесте на построение модели (пример 10).

Ответ системы:

$$f_{Ans} = \frac{105}{V} + \frac{15}{60} + \frac{105}{V+10} = \frac{210}{V}.$$

Ответ пользователя:

$$q_{Ans} = \begin{cases} t_1 = \dfrac{105}{V} \\ t_2 = \dfrac{105}{V+10} \\ \\ t_1 + 0,25 + t_2 = 2 \cdot t_1 \end{cases} .$$

Эти отличия делают задачу проверки правильности ответа нетривиальной. Для ее решения используются несколько системных функций, каждая из которых представляет один из вариантов реализации отношения эквивалентности $f_{Ans} \sim q_{Ans}$.

♦ *EquNum*(*f*, *g*) – выражения *f* и *g* равны с точностью до формы записи числовых коэффициентов.

♦ *EquCom*(*f*, *g*) – выражения *f* и *g* равны с точностью до коммутативности (перестановок сомножителей и слагаемых).

♦ *EquAcc*(*f*, *g*) – выражения *f* и *g* равны с точностью до ассоциативности (расстановок скобок в произведениях и суммах).

♦ *EquVar*(*f*, *g*, *V*) – редукция (проекция) логического выражения *g* на множество переменных {*V*} равна логическому выражению *f*. В последнем пункте (варианты ответа к примеру 10) функция *Val* генерирует решение системы уравнений, представленной пользователем. Полный ответ – логическое выражение $g(t_1, t_2, V) = (t_1 = t_1^0 \ \& \ t_2 = t_2^0 \ \& \ V = V^0)$. Редукция *g* на множество {*V*} есть выражение $V = V^0$.

Можно считать, что f_{Ans} представлено в канонической форме относительно функции *Val*: $Val(f_{Ans}) = f_{Ans}$. Тогда алгоритм проверки ответа заключается в следующем:

– если *Equ*(f_{Ans}, *g*), то возвратить сигнал «Правильно», иначе

– если *Equ*(f_{Ans}, *Val*(*g*)), то возвратить сигнал «Промежуточный результат», иначе

– возвратить сигнал «Неправильно».

Здесь функция *Equ*(*f*, *g*) представляет собой конъюнкцию нескольких функций (все пункты). Например: *Equ*(*f*,*g*)=*EquNum*(*f*,*g*)&*EquCom*(*f*,*g*) проверяет правильность ответа с точностью до форм представления чисел и коммутативности.

Заключение. Системы тестирования, основанные на результатах этого исследования, реализованы в СКМУН «ТерМ» [10]. Эксперимент показал эффективность и перспективность данного подхода при условии, что реализация системы тестирования удовлетворяет требованиям стандарта *IMS*. Специфические проблемы, возникающие при реализации систем тестирования в других предметных областях, – предмет дальнейших исследований.

1. *Львов М.С.* Основные принципы построения педагогических программных средств поддержки практических занятий // УСиМ. – 2006. – № 6. – С. 70–75.
2. *Львов М.С.* Проектирование логического вывода как пошагового решения задач в математических системах учебного назначения // Там же. – 2008. – № 1. – С. 25–32.
3. *Львов М.С.* Концепция информационной поддержки учебного процесса и ее реализация в педагогических программных средах // Там же. – 2009. – № 2. – С. 52–57, 72.
4. *Львов М.С.* Концепция, архитектура и функциональность гибкой распределенной программной среды учебного назначения для средней школы. Рабочее место методиста // Там же. – 2009. – № 6. – С. 71–78.
5. *Львов М.С.* Распределенные программные среды учебного назначения. Подсистема управления учебным процессом // Там же. – 2010. – № 1. – С. 66–71.
6. *Львов М.С.* Синтез інтерпретаторів алгебраїчних операцій в розширеннях багатосортних алгебр // Вісн. Харьк. нац. ун-ту. (Серія «Математичне моделювання. Інформаійні технології. Автоматизовані системи управління»). – 2009. – № 847. – С. 221–238.
7. *Львов М.С.* Тригонометричні обчислення в математичних системах навчального призначення // Зб. наук. пр. Харківського ун-ту Повітряних Сил. – Харків: ХУПС, 2010. – **1**(23). – С. 132–136.
8. *Львов М.С.* Математичні моделі та методи підтримки ходу розв'язання навчальних задач з аналітичної геометрії // Искуственный интеллект. – 2010. – № 1. – С. 86–92.
9. *Львов М.С.* Методы генерации учебных примеров программ с нетривиальными полиномиальными инвариантами // Восточно-Европейский журнал передовых технологий. Математика и кибернетика – фундаментальные и прикладные аспекты. 2010. – № 2/4 (44). – С. 28–30.
10. *Львов М.С.*, *Львова Н.М.* Вивчаємо алгебру з ком.п'ютером: Навч. посібник. – К.: Шкільний світ, 2008. – 127 с.

Поступила 01.05.2011

Тел. для справок: (0552) 32-6781, 43-2315 (Херсон)
E-mail: *lvov@ksu.ks.ua*