УДК 681.5

ОБ ОДНОМ СПОСОБЕ ФОРМАЛИЗАЦИИ СТРУКТУР ТЕСТОВЫХ ЗАДАНИЙ

© 2002 г. *А.Н. Иванченко, П.В. Шлыков*

Анализ многочисленных научно-педагогических и практических публикаций по теории и технологии тестового контроля знаний позволяет предложить следующую обобщенную структуру тестового задания (рис. 1).

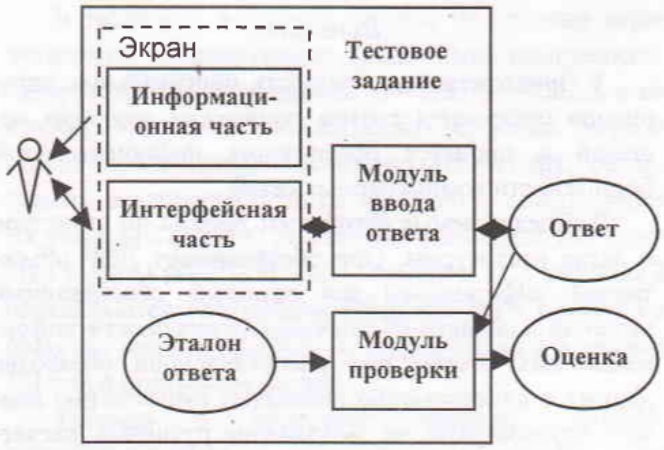

Рис. 1

«Информационная часть» тестового задания (ТЗ) содержит постоянную (неизменяемую) часть задания: постановку проблемы, привила выполнения задания, комментарии, множество вариантов ответов (в текстовой или графической форме), поясняющие рисунки, анимационные и видеосюжеты и т.п.

«Интерфейсная часть» обеспечивает интерактивность ТЗ и содержит поля для ввода ответов тестируемого. В общем случае это могут быть поля ввода числовых или символьных данных, битовые поля (поля типа "CheckBox"), «чувствительные зоны» на графических изображениях и т.п. Функционирование интерфейсной части поддерживается «Модулем ввода ответа», который интерпретирует действия тестируемого и формирует структуру данных специального вида – «Ответ».

Результат выполнения тестового задания в виде оценки формируется «Модулем проверки», который производит сравнение ответа с «Эталоном ответа», хранящимся в составе ТЗ.

Одним из основных требований при разработке систем тестового контроля является требование унификации информационного и программного обеспечения при сохранении в разумных пределах разнообразия типов тестовых заданий. В современной практике тестового контроля используются, в основном, следующие формы тестовых заданий [1]:

- закрытой формы (с множественным выбором), в которых тестируемый выбирает правильный ответ (ответы) из предложенного набора ответов;
- открытой формы, требующие от тестируемого самостоятельного формулирования ответа (в кратком или развернутом виде) и его «ручного» ввода;
- на установление соответствия (с множественным выбором), выполнение которых связано с выявлением связей между элементами двух множеств;
- на установление правильной последовательности, в которых тестируемый должен перечислить (пронумеровать) элементы заданного множества.

Исключая из рассмотрения задания открытой формы, заслуживающие самостоятельного рассмотрения, и опуская несущественные детали, примем, что любое ТЗ содержит в информационной части предикат (явный или неявный) и одно или два множества, элементы которых тестируемый мысленно подставляет в этот предикат и делает вывод об истинности получившегося высказывания. В практике тестирования получили распространение ТЗ, базирующиеся на:

- нульместных предикатах (высказываниях), информационная часть которых имеет форму высказывания, относительно которого тестируемый должен принять решение, истинно оно или ложно;
- одноместных предикатах (предикатах-свойствах), когда условие задания представлено в форме предиката $P(x)$ и задано конечное множество X значений для x. Требуется произвести разбиение этого множества на два – X_1 и X_2, удовлетворяющих условиям $P(x \mid x \in X_1) = true$, $P(x \mid x \in X_2) = false$. Другими словами, на множестве X необходимо указать элементы, обладающие свойством $P(x)$;

• двуместных предикатах (предикатах-отношениях) и содержащих предикат $P(x,y)$ и два конечных множества значений: X для x и Y для y (возможен случай совпадения множеств X и Y). Требуется построить множество пар (т.е. перечислить пары, образующие бинарное отношение $\rho \in X \times Y$), удовлетворяющих условию $P(x, y \mid x \in X, y \in Y) = true$.

Частным случаем предикатов-отношений являются отношения предшествования-следования (т.е. отношения строгого или совершенного строгого порядка) на множестве X. К этому типу, в частности, относятся тестовые задания на выбор правильной последовательности. Заметим, что использование предикатов-отношений предполагает выявление тестируемым определенных структурных характеристик у заданного множества (или у заданных множеств), что объективно сложнее, чем выявление определенных свойств у элементов множества.

Предлагается следующая классификационная схема (рис. 2).

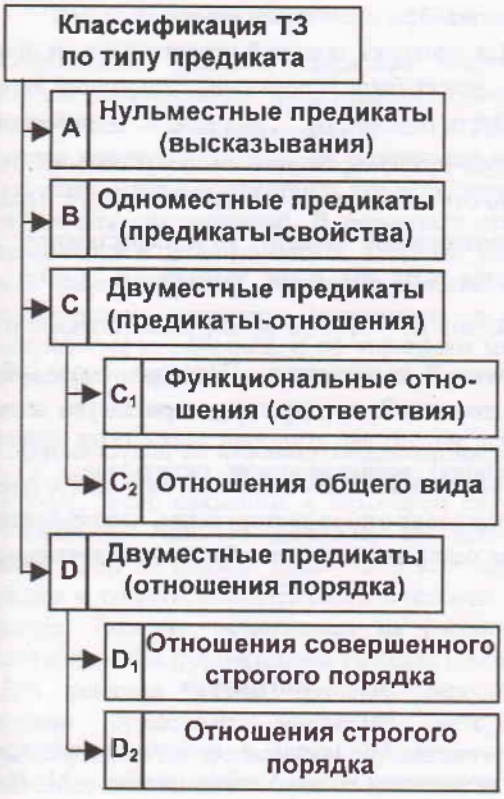

Рис. 2

Рассмотрим, какие структуры данных для хранения эталонов ответов и ввода ответов тестируемых целесообразно применять для различных типов ТЗ. Заметим, что базовой структурой данных для всех типов ТЗ является конечное множество, а наиболее эффективным способом представления n-элементных конечных множеств являются n-разрядные битовые шкалы (последовательности двоичных цифр, каждая из которых определяет, присутствует ли во множестве некоторый элемент). Битовые шкалы естественным образом упаковываются в скалярные данные подходящего размера или, при необходимости, в массивы скалярных данных. Размер скалярных данных в компьютерах современной архитектуры достигает 32 и даже 64 бита и этой емкости вполне достаточно для представления множеств в тестовых заданиях.

Заметим, что на практике в основном используются ТЗ типов A, B, C_1 и D_1. Для ТЗ в форме высказывания (тип A) естественной формой хранения ответа является логическая (булева) переменная, однако в целях унификации с другими типами ТЗ возможно использование одноразрядного битового шаблона. Для ТЗ в форме предиката-свойства (тип B) естественным является использование n-разрядного битового шаблона. Для ТЗ в форме функционального отношения (соответствия) между элементами двух множеств (тип C_1) можно использовать n-элементный массив (n – число элементов первого множества), элементами которого являются m-разрядные битовые шаблоны (m – число элементов второго множества). Наконец, для ТЗ в форме отношения совершенного строгого порядка (тип D_1) можно использовать n-элементный целочисленный массив (по числу элементов множества), элементами которого могут быть числа от 1 до n.

Таким образом, для четырех наиболее распространенных типов ТЗ можно использовать одну унифицированную структуру данных для хранения ответов тестируемого и эталонных ответов – целочисленный массив. Это позволяет использовать унифицированные модули ввода ответа и модули проверки для всех типов ТЗ и при этом предельно упростить алгоритм проверки, т.к. требуется произвести сравнение на равенство двух целочисленных массивов.

Оставшиеся типы – C_2 и D_2, представляют «неиспользованный резерв». Их применение, с одной стороны, представляет определенные трудности в силу существенно более высокой сложности подготовки ТЗ. Но, с другой стороны, ТЗ этих типов позволяют проверить знание тестируемым достаточно сложных структурных характеристик заданных множеств и, тем самым, расширить диапазон применения тестового контроля, приблизив его, отчасти, к возможностям тренажеров (для типа D_2).

Тестовые задания типа C_2 используют отношения общего вида (отношения «многие ко многим»). Примером подобного задания может служить установление отношения между элементами множеств X:«Болезни» и Y:«Лекарственные препараты». Естественной структурой данных для хранения ответа на подобные ТЗ является множество упорядоченных пар

целых чисел $\rho = \{(x, y)\}$. Однако можно предложить и в этом случае использовать n-элементный массив, элементами которого будут m-разрядные битовые шаблоны, представляющих сечения отношения $\rho \mid x$ по элементам множества X. Пусть, например, $n = 4$, $m = 5$ и

$$\rho = \{(1,2), (1,5), (2,2), (2,4), (2,5), (3,3), (4,1), (4,2)\}.$$

Множество сечений

$$\{\{2,5\}, \{2,4,5\}, \{3\}, \{1,2\}\}$$

можно «упаковать» в массив: {18, 26, 4, 3}, в котором первый элемент хранит связи первого элемента множества X со всеми элементами множества Y ($18=2^1+2^4$), второй элемент – связи второго элемента ($26=2^1+2^3+2^4$) и т.д. Простым решением для интерфейсной части тестовых заданий типа C_2 является использование матрицы с ячейками типа "CheckBox".

Тестовые задания типа D_2 эффективны при проверке знаний на воспроизведение сложных разветвленных последовательностей выполнения определенных действий (операций, работ и т.п.) и в этом они сближаются (или даже совпадают) со сценарными тренажерами. Пусть, например, для пуска в работу технологической установки необходимо последовательно выполнить 12 операций, причем существует много допустимых последовательностей, и при производстве пусковых операций необходимо лишь руководствоваться заданными ограничениями непосредственного предшествования-следования, представленными сетевым графиком в форме «вершины – работы» (рис. 3).

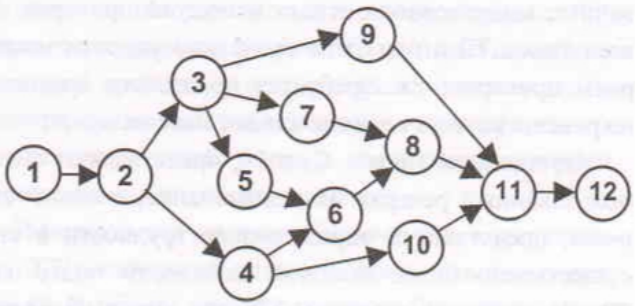

Рис. 3

Другой формой записи этого графика является отношение $\hat{\rho}(x, y)$, задаваемое множеством пар элементов (в паре $x \to y$ элемент x непосредственно предшествует элементу y):

$$\left\{\begin{matrix} 1 \to 2,2 \to 3,2 \to 4,3 \to 5,3 \to 7,3 \to 9,4 \to 6,4 \to 10, \\ 5 \to 6,6 \to 8,7 \to 8,8 \to 11,9 \to 11,10 \to 11,11 \to 12 \end{matrix}\right\}.$$

Очевидно, что, как и в заданиях типа C_2, данное отношение можно «упаковать» с использованием сечений в целочисленный массив (здесь n=m=12): {2, 12, 336, 544, 32, 128, 128, 1024, 1024, 1024, 2048, 0}.

Формулировка тестового задания типа D_2 эквивалентна требованию перечисления элементов заданного множества в каком-либо допустимом порядке. Поэтому модуль проверки должен уметь различать допустимые последовательности (перестановки), исходя из известной структуры отношения. Например, при условиях предыдущего примера перестановки {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12} и {1, 2, 4, 10, 3, 5, 6, 7, 9, 8, 11, 12} являются допустимыми, а перестановка {1, 2, 3, 5, 6, 4, 7, 9, 8, 10, 11, 12} – нет. Выполнить такую проверку с использованием только отношения $\hat{\rho}(x, y)$ невозможно, т.к. $\hat{\rho}(x, y)$ – это редукция отношения строгого порядка, а не само отношение [2]. Поэтому перед проверкой необходимо восстановить полное отношение строгого порядка $\rho(x, y)$, выполняя операцию транзитивного замыкания отношения $\hat{\rho}(x, y)$ с использованием, например, эффективного алгоритма Уоршалла [3].

Для проверки заданной перестановки на допустимость может быть использован следующий алгоритм [2]. Пусть $P = \{p_1, p_2, ..., p_n\}, p_i \in X$ – перестановка, которую нужно проверить на допустимость относительно отношения строгого порядка $\rho(x, y)$. Пусть p_i – произвольный элемент этой перестановки. Если найдется хотя бы один элемент $p_j, j = i+1, ..., n$, такой, что $p_j \to p_i$, то p_i нарушает порядок и перестановка P недопустима. Проверяя таким образом последовательно $p_1, p_2, ..., p_{n-1}$, реализуем алгоритм проверки последовательности на допустимость. Заметим, что этот алгоритм позволяет выполнять проверку по ходу ввода ответа тестируемым, что особенно ценно для реализации режима обучающего тестирования.

Литература

1. *Челышкова М.Б.* Теория и практика конструирования педагогических тестов: Учебное пособие. – М.: Исследовательский центр проблем качества подготовки специалистов, 2001. – 410с.
2. *Танаев В.С., Шкурба В.В.* Введение в теорию расписаний. – М.: Гл. ред. физ.-мат. лит-ры изд-ва «Наука», 1975. – 256с.
3. *Липский В.* Комбинаторика для программистов. – М.: Мир, 1988. – 213с.