ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
2021 Управление, вычислительная техника и информатика № 57

ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ

УДК 004.434
DOI: 10.17223/19988605/57/11

И.А. Жуков, Ю.Л. Костюк

РАСШИРЕННЫЙ АЛГОРИТМ КОНТРОЛЯ ЗНАНИЙ ДЛЯ МОДЕЛИ ПРЕДСТАВЛЕНИЯ МНОГОВАРИАНТНЫХ ЗАДАНИЙ

Для модели представления многовариантных заданий расширено понятия эксперта. Описаны его составные части: считыватель и анализатор. Предложен расширенный алгоритм оценки соответствия, проводящий развернутый анализ ответа обучающегося с предоставлением максимально детализированной информации для преподавателя.
Ключевые слова: автоматизированный контроль знаний; проверка формализованных ответов; сопоставление с последовательностью образцов.

Решение проблемы автоматизированного контроля знаний для многовариантных заданий по программированию можно разделить на два этапа: создание модели представления заданий и разработка алгоритма контроля. Модель представления задания приведена в [1]. Эта модель описывает формальное представление множества правильных ответов с помощью компонентов, элементов и шаблонов. В проверке формализованных ответов существует подход, основанный на копировании простейших контролирующих действий человека, т.е. в процессе контроля ответ обучающегося сравнивается с некоторым эталоном (правильным ответом, образцом) [2]. Задачи сопоставления с образцом встречаются в различных областях информатики. В частности, они используются в компиляторах [3], системах верификации [4], поисковых системах [5] и анализе естественных языков [6].

Одним из способов задания образцов служат регулярные выражения. Они применяются, например, в текстовых редакторах [7]. Авторы [8] использовали регулярные выражения для анализа баз данных электронных медицинских библиотек.

Характерной чертой заданий по программированию является существование множества правильных ответов. Для формализации в [1] предложено группировать правильные ответы по степени сходства. Средствами группировки в модели являются элементы, которые подобны регулярным выражениям, но в силу особенностей предметной области имеют свою специфику. В модели определены три типа элементов: «Один компонент», «Один из компонентов» и «Перестановка компонентов». Последовательность элементов (шаблон) задается на формальном языке. В результате правильный ответ на задание представлен множеством шаблонов. Поэтому алгоритм контроля должен обеспечивать сопоставление с множеством образцов. В модели вводится понятие эксперта. Экспертом назван автомат, оценивающий соответствие варианта решения (входной последовательности) одному шаблону. Следовательно, для одного задания ответ обучающегося оценивает множество экспертов.

В данной работе понятие эксперта модифицировано и представлен расширенный алгоритм оценки соответствия, проводящий развернутый анализ ответа обучающегося с предоставлением максимально детализированной информации для преподавателя.

1. Модель эксперта

Ответ обучающегося эксперт получает в виде последовательности номеров компонентов, которые являются натуральными числами. Для создания эксперта необходимо задать шаблон; множество

всех рубежных элементов из шаблона и количество считываемых компонентов *n*, по умолчанию равное 2.

В процессе оценки вспомогательными средствами для эксперта служат считыватель и анализатор. Считыватель на основе информации о текущем и следующем элементе шаблона читает подпоследовательность из ответа по следующим правилам:

– если текущий элемент не является «перестановкой компонентов» или не помечен как рубежный, считать *n* компонентов;

– для рубежного элемента считать только один компонент;

– для элемента «перестановка компонентов» считыватель просматривает все значения до ближайшего из компонентов следующего рубежного элемента, если ни один из компонентов следующего рубежного элемента не представлен, считывается пустая последовательность.

Затем анализатор оценивает соотношение текущего элемента шаблона и прочитанной подпоследовательности. Для оценки соответствия подпоследовательности и элементов «Один компонент» и «Один из компонентов» достаточно регулярных выражений. Существует две спецификации регулярных выражений: POSIX [9] и Perl [10]. Задача оценки для приведенных двух типов элементов решается средствами любой из этих спецификаций. Элемент «Перестановка компонентов» не может быть представлен средствами спецификаций регулярных выражений. Существуют расширения, позволяющие определять перестановки, например описанное в [11]. Для нашей задачи достаточно определить, какие из компонентов перестановки присутствуют в подпоследовательности, при этом не уточняя, какая именно перестановка представлена.

Для определения соответствия необходимы индекс элемента в подпоследовательности компонентов (далее – индекс элемента) и мощность пересечения множеств всех компонентов элемента и компонентов подпоследовательности (далее – мощность). Для элементов типа «Один компонент» или «Один из компонентов» мощность всегда равна 0. Мощность указывает количество компонентов перестановки в подпоследовательности. Индекс принимает целые значения от –1 до $n - 1$. Значение индекса равно –1, если элемент имеет тип «Перестановка компонентов» или в считанной подпоследовательности не найдено ни одного компонента данного элемента. Индекс равен 0, если подпоследовательность начинается с одного из компонентов текущего элемента. Индекс больше 0 означает, что какой-либо из компонентов элемента присутствует в подпоследовательности, но не первым. В таком случае индекс показывает позицию первого встретившегося компонента данного элемента.

Оценкой соотношения элемента и подпоследовательности является одно из трех значений: 0 – подпоследовательность полностью удовлетворяет элементу, 1 – подпоследовательность частично удовлетворяет элементу, 2 – подпоследовательность не удовлетворяет элементу. Алгоритм анализа представлен деревом принятия решения, изображенным на рис. 1.

В процессе анализа также вычисляется коэффициент ошибки, выявляются отсутствующие и «лишние» компоненты. Коэффициент ошибки определяется следующим образом:

– если подпоследовательность полностью удовлетворяет элементу, то коэффициент равен 0;

– если подпоследовательность не удовлетворяет, то коэффициент равен 1;

– для частично удовлетворяющих: если элемент имеет тип «Перестановка компонентов», то коэффициент ошибки равен разности длины элемента и мощности, делннной на длину элемента; для остальных типов элементов – индекс элемента, деленный на длину подпоследовательности.

Отсутствующими компонентами будем считать компоненты, которые входят в элемент, но не входят в подпоследовательность. Для элемента типа «Один из компонентов», если любой из компонентов присутствует в подпоследовательности, то отсутствующие компоненты не вычисляются. «Лишними» компонентами будем считать компоненты, которые входят в подпоследовательность, но не входят в элемент.

В результате анализа получается запись, имеющая следующую структуру: элемент, считанная из ответа подпоследовательность, индекс элемента, мощность, оценка соответствия, коэффициент ошибки, отсутствующие компоненты, «лишние» компоненты.

По результатам анализа считыватель выставляет позицию начала следующего считывания согласно алгоритму:

– если индекс элемента и мощность меньше 1, сместить позицию в ответе обучающегося вправо на 1;

– если элемент не помечен как опциональный, сместить позицию в ответе обучающегося вправо на длину считанной подпоследовательности;

– если элемент помечен как опциональный, позиция не меняется.

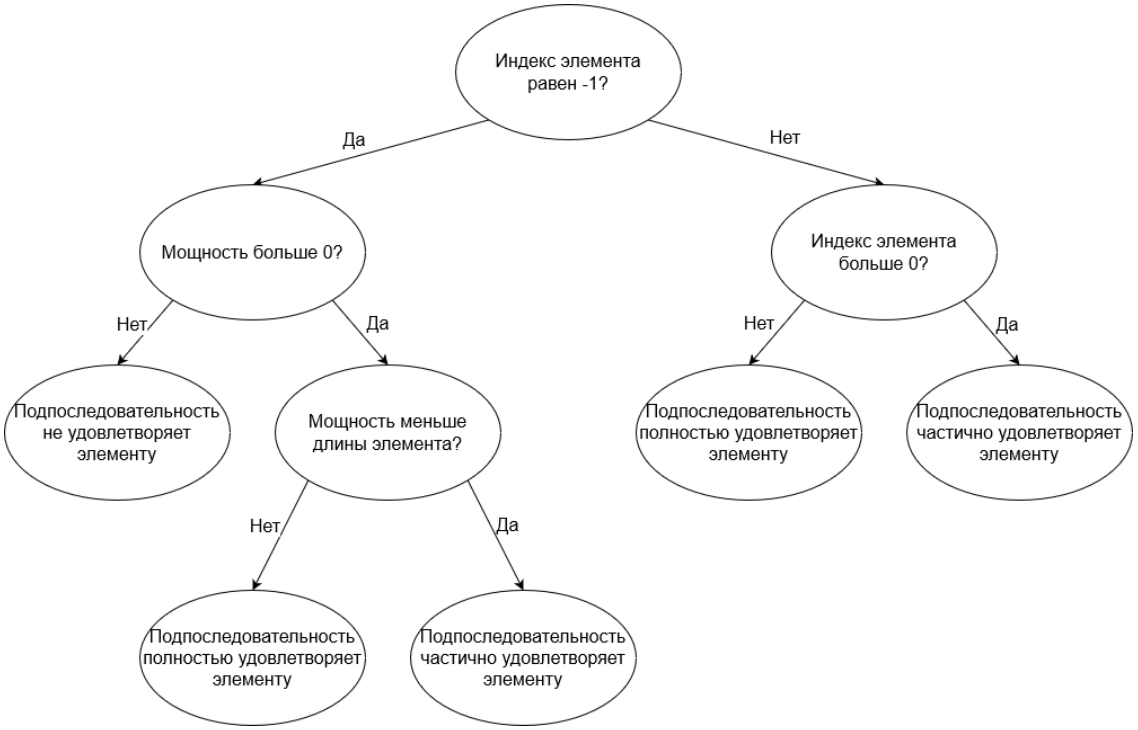

Рис. 1. Дерево принятия решений для оценки соответствия элемента считанной последовательности из ответа обучающегося
Fig. 1. Analysis algorithm represented by decision tree

В ходе работы эксперт сохраняет результаты анализа. В итоге записи о процессе работы эксперта состоят из трех частей: записей о результатах анализа, списка «лишних» компонентов и причины завершения анализа. Записи о результатах анализа делаются последовательно для каждого элемента шаблона, т.е. одна запись соответствует одному элементу. Список «лишних» компонентов относится к компонентам, которые остались несчитанными после завершения работы с шаблоном. Причина завершения классификации может иметь одно из трех значений: закончился шаблон, закончился ответ, закончился и шаблон, и ответ.

Результатом работы является мнение эксперта, которое содержит оценку в виде обыкновенной дроби, таблицу ошибок и записи о процессе работы эксперта. Записи в таблице ошибок имеют вид: позиция элемента в шаблоне, элемент, характеристика ошибки, отсутствующие компоненты, «лишние» компоненты. Характеристика ошибки может иметь одно из четырех значений: 0 – элемент отсутствует, 1 – элемент частично присутствует (часть компонентов отсутствует), 2 – элемент присутствует с лишними компонентами, 3 – элемент присутствует, но часть компонентов отсутствует, и имеются лишние компоненты.

2. Алгоритм работы эксперта

На рис. 2 приведена диаграмма деятельности, описывающая алгоритм работы эксперта.

Остановимся на некоторых шагах алгоритма. В начале работы алгоритма вычисляется пересечение множеств всех компонентов рубежных элементов и всех компонентов ответа. Если множество

всех рубежных элементов непустое, а пересечение множеств – пустое, то эксперт оценит соответствие на 0, в таблицу ошибок заносятся все рубежные элементы, все ошибки буду иметь характеристику «элемент отсутствует».

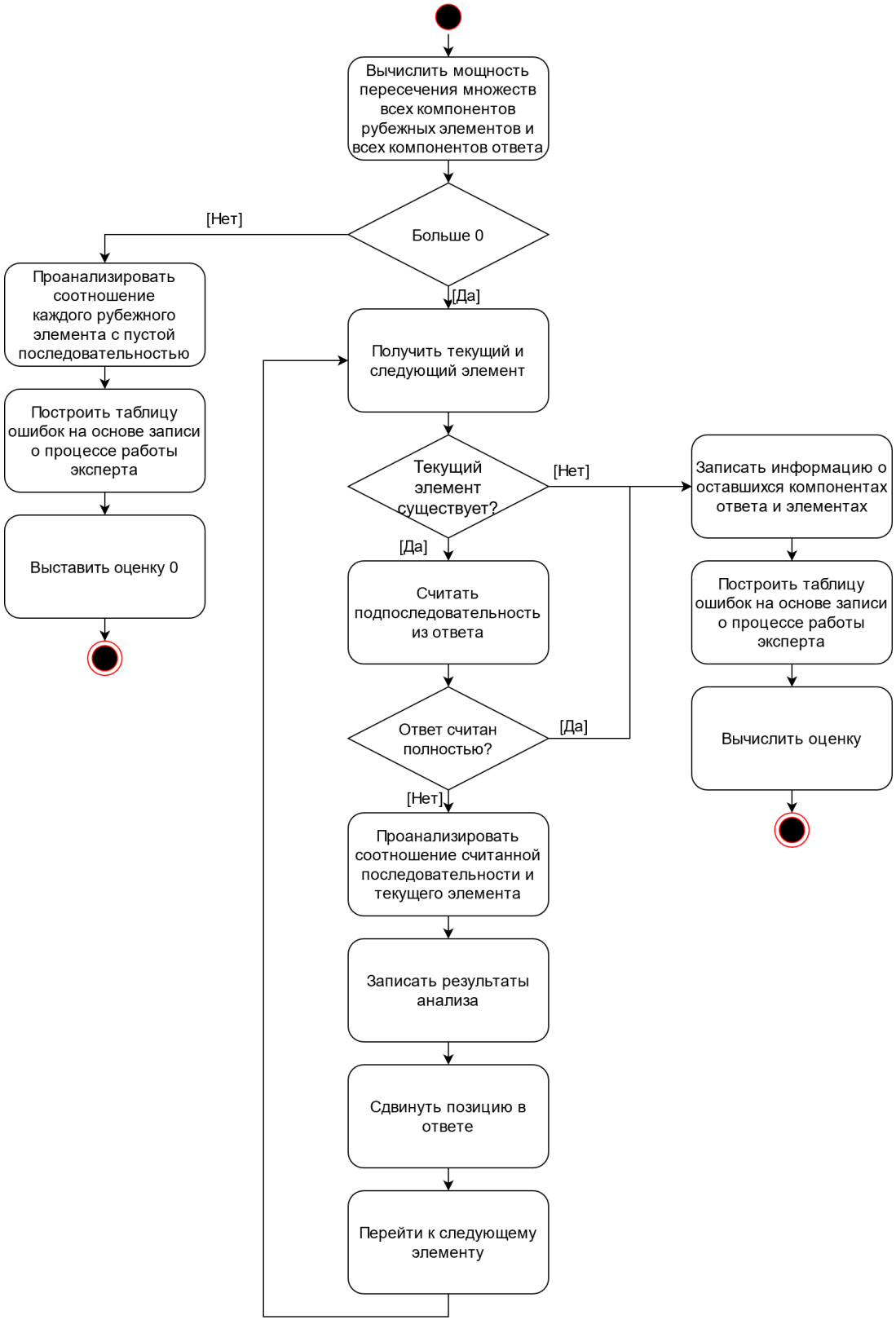

Рис. 2. Диаграмма деятельности для алгоритма оценки
Fig. 2. Activity diagram for assessment algorithm

Если причина завершения анализа – «закончился ответ», проанализировать соотношение каждого оставшегося элемента с пустой последовательностью. То есть в записи о процессе работы эксперта это будет отражено как элемент, не удовлетворяющий подпоследовательности, при этом отсутствуют все компоненты элемента и не присутствует «лишних».

Таблица ошибок строится по следующему алгоритму:

– если соответствие элемента и подпоследовательности оценено как «подпоследовательность не удовлетворяет элементу», соответствующий элемент характеризовать как «элемент отсутствует»;

– если соответствие элемента и подпоследовательности оценено как «подпоследовательность частично удовлетворяет элементу» и количество отсутствующих компонентов не равно 0, характеризовать как «элемент частично присутствует»;

– если количество «лишних» компонентов не равно 0, характеризовать как «присутствует с лишними компонентами»;

– если и количество отсутствующих компонентов, и количество «лишних» компонентов не равно 0, характеризовать как «присутствует, но часть компонентов отсутствует, и имеются лишние компоненты». Список отсутствующих компонентов и список «лишних» компонентов скопировать.

Оценка *M* вычисляется по формуле

$$M = 1 - \sum_{i=1}^{n} w_i e_i p(f_i) - w_d p_{extra}(a_{extra} + a_{ee}), \quad (1)$$

где w_i – вес *i*-го элемента, e_i – коэффициент ошибки *i*-го элемента, $p(f_i)$ – множитель штрафа для метки *i*-го элемента, w_d – вес элемента, по умолчанию равный единице, деленной на длину шаблона, p_{extra} – множитель штрафа для «лишних» компонентов, a_{extra} – количество «лишних» компонентов, не привязанных к элементам, a_{ee} – количество «лишних»" компонентов в элементах. Если вычисленная оценка получилась меньше 0, оценка будет приравнена к 0.

3. Пример работы алгоритма

Разберем работу эксперта на примере. Для примера возьмем один шаблон из задания, приведенного в [1]. Рассмотрим работу эксперта, который оценивает соответствие ответа шаблону {(1;4);5;6;3*;7|8;9}. Пусть ответ обучающегося будет «2;1;5;10;6;3;8;11».

Все программы, соответствующие полностью правильным ответам:

```
S:=0;
i:=0;
while i <= n do
begin
S:=S + m[i];
i:=i + 1;
end;
```

```
S:=0;
i:=0;
while i <= n do
begin
S:=S + m[i];
inc(i);
end;
```

```
i:=0;
S:=0;
while i <= n do
begin
S:=S + m[i];
i:=i + 1;
end;
```

```
i:=0;
S:=0;
while i <= n do
begin
S:=S + m[i];
inc(i);
end;
```

Программа, соответствующая ответу обучающегося:

```
for i:=1 to n do
S:=0;
while i <= n do
repeat
begin
S:=S + m[i];
inc(i);
until i > n;
```

Рассмотрим работу эксперта пошагово. Количество считываемых компонентов $n = 2$. Множество всех компонентов рубежных элементов из шаблона равно {3, 5}, множество всех компонентов ответа равно {2, 1, 5, 10, 6, 3, 8, 11, 9}. Пересечение этих множеств равно {3, 5}, |{3, 5}| = 2 > 0, сле-

довательно, эксперт продолжит оценивание. Позиция считывателя находится в начале ответа. Пройдем шаги алгоритма.

1. Текущий элемент (1;4) имеет тип «Перестановка компонентов». Индекс элемента будет равен –1. За перестановкой находится рубежный элемент с компонентом 5. Согласно алгоритму считывателя из ответа будет прочитана подпоследовательность, состоящая из двух компонентов: 2, 1. Для этого элемента мощность пересечения $|\{1, 4\} \cap \{2, 1\}| = 1$, что меньше длины элемента. Следовательно, подпоследовательность частично удовлетворяет элементу, коэффициент ошибки будет равен 0,5, отсутствует компонент 4, «лишним» компонентом будет 2. Текущая позиция считывателя сдвинется на 2 вправо.

2. Текущий элемент 5 имеет тип «Один компонент». Для этого элемента мощность равна 0. Так как элемент помечен как рубежный, из ответа будет считана подпоследовательность, состоящая из одного компонента 5. Индекс элемента равен 0, следовательно, подпоследовательность полностью удовлетворяет элементу. Текущая позиция считывателя сдвинется на 1 вправо.

3. Текущий элемент 6 имеет тип «Один компонент». Для этого элемента мощность равна 0. Из ответа будет считана подпоследовательность, состоящая из двух компонентов: 10, 6. Индекс равен 1, следовательно, подпоследовательность частично удовлетворяет элементу, «лишним» компонентом будет 10. Текущая позиция считывателя сдвинется на 2 вправо.

4. Текущий элемент 3* имеет тип «Один компонент». Для этого элемента мощность равна 0. Элемент помечен как рубежный. Из ответа будет считана подпоследовательность, состоящая из одного компонента 3. Индекс равен 0, следовательно, подпоследовательность полностью удовлетворяет элементу. Текущая позиция считывателя сдвинется на 1 вправо.

5. Текущий элемент 7|8 имеет тип «Один из компонентов». Для этого элемента мощность равна 0. Из ответа будет считана последовательность, состоящая из двух компонентов: 8, 11. Индекс равен 0, следовательно, один из компонентов входит в считанную подпоследовательность, т.е. подпоследовательность полностью удовлетворяет элементу. Текущая позиция считывателя сдвинется на 1 вправо.

6. Текущий элемент 9 имеет тип «Один компонент». Для этого элемента мощность равна 0. Из ответа должна быть считана подпоследовательность, состоящая из двух компонентов. Но в силу того, что текущее смещение считывателя равно 5, а длина ответа – 6, будет считана подпоследовательность, состоящая из одного компонента 11. Индекс равен –1, следовательно, подпоследовательность не удовлетворяет элементу. Текущая позиция считывателя сдвинется на 1 вправо.

7. Текущий элемент является последним, позиция в ответе равна длине ответа, следовательно, анализ завершен.

Причиной завершения анализа будет «закончился и шаблон, и ответ». Результаты анализа всех элементов приведены в табл. 1.

Таблица 1

Пример результатов анализа

Элемент	Подпоследовательность	Индекс элемента	Мощность	Соответствие	Коэффициент ошибки e_i	Отсутствующие компоненты	«Лишние» компоненты
(1;4)	2, 1	–1	1	Частично удовлетворяет	0,5	4	2
5*	5	0	0	Полностью удовлетворяет	0		
6	10, 6	1	0	Частично удовлетворяет	0,5		10
3*	3	0	0	Полностью удовлетворяет	0		
7|8	8, 11	0	0	Полностью удовлетворяет	0		
9	11	–1	0	Не удовлетворяет	1	9	11

Следующим шагом алгоритма будет построение таблицы ошибок. Таблица ошибок для данного примера приведена в табл. 2.

Таблица 2

Пример таблицы ошибок

Позиция	Элемент	Характеристика ошибки	Отсутствующие компоненты	«Лишние» компоненты
0	(1;4)	Присутствует, но часть компонентов отсутствует, и имеются лишние компоненты	4	2
2	6	Присутствует с лишними компонентами		10
5	9	Элемент отсутствует	9	11

Вычислим оценку по формуле (1). Веса элементов w_i не заданы. В этом случае, вес каждого элемента будет принят равным w_d, т.е. единице, деленной на длину шаблона, которая равна 6. Анализируя табл. 1, видим, что элементы с номерами 1, 3, 6 имеют ненулевой коэффициент ошибки e_i, а именно: $e_1 = 0{,}5$, $e_3 = 0{,}5$, $e_6 = 1$. Среди элементов с ненулевым коэффициентом ошибки нет помеченных, поэтому согласно [1] $p(f_1) = p(f_3) = p(f_6) = 0{,}25$. Множитель штрафа для «лишних» компонентов p_{extra} не задан, поэтому будет использовано значение по умолчанию, равное 0,75. Количество «лишних» компонентов, не привязанных к элементам, a_{extra} определяется экспертом после завершения анализа. В данном примере $a_{extra} = 0$, так как анализ завершился по причине «закончился и шаблон, и ответ». Количество «лишних» компонентов в элементах a_{ee} можно определить по табл. 1. В последнем столбце указаны номера лишних компонентов. В данном примере $a_{ee} = 3$. Подставляя указанные значения в формулу (1), получим общую оценку $M = 0{,}54$.

В данном примере задание затрагивает темы «Массив» и «Циклы». Оценка 0,54 свидетельствует о наличии серьезных проблем с освоением этих тем обучающимся. Приведенный ответ некорректен с точки зрения синтаксиса языка программирования Pascal, т.е. программа не может быть исполнена. Таблица 2 показывает, что обучающийся имеет проблемы с определением сопутствующих циклу while элементов, при этом сам заголовок цикла и суммирование элементов массива реализованы верно. Основываясь на таблице ошибок, можно подготовить рекомендации для обучающегося по изучению указанных тем.

Полученная оценка также может использоваться в системе автоматизированного контроля знаний. При достаточно большом количестве заданий такого типа по изучаемой дисциплине сумма оценок отражает уровень освоения материала обучающимся.

Заключение

В результате развития модели представления многовариантных заданий представлен алгоритм контроля, проводящий развернутый анализ ответа обучающегося с предоставлением максимально детализированной информации для преподавателя о местах, в которых обучающийся допустил ошибки, с указанием данных об отсутствующих и «лишних» компонентах. Применение алгоритма возможно как в виде отдельного инструмента автоматизированного контроля, так и в виде «помощника» преподавателя.

ЛИТЕРАТУРА

1. Жуков И.А., Костюк Ю.Л. Модель представления многовариантных заданий для автоматизированного контроля знаний по программированию // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2020. № 53. С. 110–117.
2. Шевелев М.Ю., Вишнякова Л.А., Шевелев Ю.П. Контроль знаний в компьютерном обучении. Неантропоморфный подход. Saarbrücken : LAP Lambert Academic Publishing, 2014. 395 с.
3. Васенин В.А., Кривчиков М.А. Промежуточное представление программ для описания типов в терминах сопоставления значений с образцом // Программирование. 2020. № 1. С. 63–74.
4. Гаранина Н.О., Ануреев И.С., Боровикова О.И., Зюбин В.Е., Методы специализации онтологии процессов, ориентированной на верификацию // Моделирование и анализ информационных систем. 2019. Т. 26, № 4. С. 534–549.

5. Шокин Ю.И., Федотов А.М., Барахнин В.Б. Проблемы поиска информации. Новосибирск : Наука, 2010. 220 с.
6. Тимофеев П.С., Сидорова Е.А. Лексико-семантические шаблоны как инструмент декларативного описания языковых конструкций и лингвистического анализа текста // Системная информатика. 2018. № 13. С. 35–48.
7. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов : пер. с англ. М. : Мир, 1979. 536 с.
8. Рослик Д.А., Лучинин Е.А., Арькова Е.С., Корнилова Е.Б., Толкушин А.Г., Холовня-Волоскова М.Э. Применение языка «регулярных выражений» (regular expressions) для изучения баз данных электронных медицинских библиотек в рамках оценки медицинских технологий // Московская медицина. 2020. № 4 (38). С. 73–81.
9. The Open Group Base Specifications Issue 7, 2018 edition IEEE Std 1003.1-2017 (Revision of IEEE Std 1003.1–2008). URL: https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html (accessed: 12.05.2021).
10. Perlre-Perl regular expressions. URL: https://perldoc.perl.org/perlre (accessed: 12.05.2021).
11. Lovett A.M., Shallit J. Optimal Regular Expressions for Permutations // Leibniz International Proceedings in Informatics, LIPIcs. 2019. V. 132. Art. 121.

Поступила в редакцию 3 июня 2021 г.

Zhukov I.A., Kostyuk Yu.L. (2021) AN EXTENDED ASSESSMENT ALGORITHM FOR MULTIVARIATE TASK PRESENTATION MODEL. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitelnaja tehnika i informatika* [Tomsk State University Journal of Control and Computer Science]. 57. pp. 101–109

DOI: 10.17223/19988605/57/11

The solution to the problem of automated knowledge assessment for multivariate programming tasks can be divided into two stages: the creation of a model for the presentation of tasks and the development of assessment algorithm. The model describes the formal representation of a set of correct answers using components, elements and patterns. There is an approach in formalized answers assessment based on copying the simplest actions of a human, i.e. in an assessment process a student's answer is compared with some sample (correct answer, pattern). Pattern matching problems are found in various fields of computer science. Regular expressions are one way to specify patterns.

A characteristic feature of programming assignments is that there are many correct answers. For formalization in the task presentation model, it is proposed to group the correct answers according to the degree of similarity. The means of grouping in the model are elements that are similar to regular expressions, but due to the peculiarities of the domain, they have their own specifics. Three types of elements are defined in the model: “One component”, “One of the components” and “Permutation of components”. A sequence of elements (pattern) is specified in a formal language. As a result, the correct answer to the task is represented by many patterns. The concept of an expert is introduced into the model. An expert called an automaton that evaluates the correspondence of a solution option (input sequence) to one pattern. Therefore, for one task, the student's answer is assessed by many experts.

The expert receives the student's answer in the form of a sequence of component numbers, which are natural numbers. To create an expert, it is necessary to specify a pattern, the set of all important elements from the pattern, and the number of readable components n, by default equal to 2. In the process of evaluation, the reader and the analyzer serve as auxiliary means for the expert. The reader, based on information about the current and next pattern element, reads a subsequence from the answer.

The analyzer evaluates the ratio of the current pattern element to the read subsequence. Regular expressions are sufficient to evaluate the match between a subsequence and the elements “One component” and “One of the components”. The “Permutation of components” element cannot be represented by regular expressions. For our problem, it is enough to determine which of the components of the permutation are present in the subsequence, without specifying which particular permutation is represented.

To determine the match, the index of the element in the subsequence of components (element index) and the cardinality of the intersection of the sets of all components of the element and the components of the subsequence are required. For elements of type "One component" or "One of components" the cardinality is always 0. Cardinality indicates the number of components of the permutation in the subsequence. The index takes integer values from –1 to (n – 1). The index value is –1 if the element is of type “Permutation of components” or no component of this element was found in the read subsequence. The index is 0 if the subsequence starts with one of the components of the current element. An index greater than 0 means that any of the element's components are present in the subsequence, but not on first place. In this case, the index indicates the position of the first encountered component of this element.

The estimate of the ratio of an element and a subsequence is one of three values: 0 is the subsequence fully satisfies the element, 1 is the subsequence partially satisfies the element, 2 is the subsequence does not satisfy the element. During the analysis, the error rate is also calculated, and the missing and “extra” components are identified.

During the assessment, the expert saves the results of the analysis. As a result, the records about the experts work process consist of three parts: records about the analysis results, the list of “extra” components, and the reason for the completion of the analysis. The analysis results are recorded sequentially for each element of the pattern, i.e. one record corresponds to one element. The list of "extra" components refers to the components that were left unread after finishing work with the pattern. The reason for the completion of the analysis can have one of three meanings: the pattern ended, the answer ended, both the pattern and the answer ended.

The result of the work is an expert's opinion, which contains an assessment in the form of an ordinary fraction, a table of errors and records about the expert's work process. The entries in the error table are of the form: position of an element in the pattern,

element, error characteristics, missing components, “extra” components. The error characteristic can have one of four values: 0 is an element missing, 1 is an element partially present (some components are missing), 2 an element present with extra components, 3 an element present, but some components are missing and there are extra components. The score *M* is calculated by the formula:

$$M = 1 - \sum_{i=1}^{n} w_i e_i p(f_i) - w_d p_{extra}(a_{extra} + a_{ee}),$$

where w_i is the weight of the *i*-th element, e_i is the error coefficient for the *i*-th element, $p(f_i)$ is the penalty multiplier for the flag of the *i*-th element, w_d is the default weight of the element equal to one divided by the length of the pattern, p_{extra} is the penalty multiplier for "extra" components, a_{extra} is the number of "extra" components that are not tied to elements, a_{ee} is the number of "extra" components in the elements. If the calculated score is less than 0, the score will be set to 0.

Keywords: automated knowledge assessment; assessment of formalized answers; matching with sequence of patterns.

ZHUKOV Igor Andreevich (Post-graduate Student, Institute of Applied Mathematics and Computer Science, National Research Tomsk State University, Tomsk, Russian Federation).
E-mail: Ig.Zhukov963@yandex.ru

KOSTYUK Yuriy Leonidovich (Doctor of Technical Sciences, Professor, Institute of Applied Mathematics and Computer Science, National Research Tomsk State University, Tomsk, Russian Federation).
E-mail: kostyuk_y_l@sibmail.com

REFERENCES

1. Zhukov, I.A. & Kostyuk, Yu.L. (2020) Model of representation of multivariate tasks for automatedcontrol of programming knowledge. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitel'naya tekhnika i informatika – Tomsk State University Journal of Control and Computer Science*. 53. pp. 110–117. DOI: 10.17223/19988605/53/11
2. Shevelev, M.Yu., Vishnyakova, L.A. & Shevelev, Yu.P. (2014) *Kontrol' znaniy v komp'yuternom obuchenii. Neantropomorfnyy podkhod* [Knowledge control in computer learning. The Non-anthropomorphic approach]. Saarbrücken: LAP LAMBERT Academic Publishing.
3. Vasenin, V.A. & Krivchikov, M.A. (2020) Intermediate representation of programs with type specification based on pattern matching. *Programmirovanie – Programming and Computer Software.* 46(1). pp 57–66. DOI: 10.1134/S0361768820010077
4. Garanina, N.O., Anureev, I.S., Borovikova, O.I. & Zyubin, E.V. (2019) Methods for Domain Specification of Verification-Oriented Process Ontology. *Modelirovanie i analiz informatsionnykh sistem – Modeling and Analysisof Information Systems*. 26(4). pp. 534–549. DOI: 10.18255/1818-1015-2019-4-534-549
5. Shokin, Yu.I., Fedotov, A.M., Barakhnin, V.B. (2010) *Problemy poiska informatsii* [Problems of Finding Information]. Novosibirsk: Nauka.
6. Timofeev, P.S. & Sidorova, E.A. (2018) A lexico-semantic templates as a tool for declarative description language constructs linguistic text analysis. *Sistemnaya informatika – System Informatics*. 13. pp. 35–48. DOI: 10.31144/si.2307-6410.2018.n13.p35-48
7. Aho, A., Hopcroft, J. & Ullman, J. (1974) *Postroenie i analiz vychislitel'nykh algoritmov* [Design and Analysis of Computer Algorithms]. Translated from English. Moscow: Mir.
8. Roslik, D.A., Luchinin, E.A., Arkova, E.S., Kornilova, E.B., Tolkushin, A.G. & Kholovnya-Voloskova, M.E. (2020) Use of regular expressions language for processing text data sets when extracted from databases of electronic medical libraries in the framework of medical technology assessment. *Moskovskaya meditsina – Moscow Medicine Journal*. 4(38). pp. 73–81.
9. Opengroup.org. (2018) *The Open Group Base Specifications*. Issue 7. IEEE Std 1003.1-2017 [Online] Available from: https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html (Accessed: 12th May 2021).
10. Perldoc Browser. (n.d.) *Perlre-Perl regular expressions*. [Online] Available from: https://perldoc.perl.org/perlre (Accessed: 12th May 2021).
11. Lovett, A.M. & Shallit, J. (2019) Optimal Regular Expressions for Permutations. *Leibniz International Proceedings in Informatics.* 132. Art. 121. DOI: 10.4230/LIPIcs.ICALP.2019.121