ISSN 0868–5886 НАУЧНОЕ ПРИБОРОСТРОЕНИЕ, 2006, том 16, № 1, с. 116–120

ОРИГИНАЛЬНЫЕ СТАТЬИ

УДК 519.6: 007.3

© В. Я. Мамаев, К. К. Петров

КОНТРОЛЬ ЗНАНИЙ ОБУЧАЕМОГО ПРИ ВЫПОЛНЕНИИ ИМ СИМВОЛЬНЫХ ПРЕОБРАЗОВАНИЙ В ИНТЕЛЛЕКТУАЛЬНОЙ ОБУЧАЮЩЕЙ СИСТЕМЕ

В процессе обучения на тренажере оператора летательного аппарата решению навигационных задач необходимо осуществлять контроль его знаний. Предлагается алгоритм формульных преобразований, позволяющий контролировать правильность выполнения обучаемым тестового задания, выданного ему в максимально свободной форме.

ВВЕДЕНИЕ

Одной из важных задач при создании интеллектуальных обучающих систем (ИОС) является реализация тестового контроля знаний, при этом желательно реализовать тестовые задания открытого типа с максимально свободной формой представлений вопроса и ответа. Выполнение данного требования сдерживается сложностью решения проблемы машинного понимания естественного языка и проблемы оценки полученных ответов. Целью работы является реализация средств сопоставления и преобразования формул, обеспечивающих контроль знаний обучаемого.

Поскольку при решении навигационных задач символьные преобразования составляют значительную часть процесса решения, возникает необходимость разработки моделей и алгоритмов, позволяющих моделировать процесс символьных преобразований. Причем при разработке моделей и алгоритмов необходимо учитывать то, что они будут реализованы в инструментальных оболочках, используемых для обучения. Таким образом, необходимо разработать достаточно общие методы, осуществляющие символьные преобразования и допускающие возможность параметризации их наборами правил преобразований. При осуществлении преобразований происходит моделирование процесса решения задания.

Символьные преобразования необходимы также при сравнении ответа обучаемого и ответа, полученного системой. Действительно, одно и то же математическое выражение может быть представлено различными семантически эквивалентными формулами. Поэтому прежде чем сравнивать свой ответ и ответ обучаемого, система осуществляет преобразование обоих ответов к некоторой стандартной форме. Что понимать под стандартной формой определяет сам разработчик обучающего курса, который задает формулы преобразования в модуле символьных преобразований.

В процессе проектирования модуля символьных преобразований возникает необходимость решения следующих задач:

1) выбор способа представления математических формул в памяти компьютера и разработка алгоритма преобразования формул к выбранному представлению;

2) разработка стандартного вида формул и алгоритма приведения формул к стандартному виду;

3) разработка метода символьных преобразований на основе задаваемых шаблонов.

ВНУТРЕННЕЕ ПРЕДСТАВЛЕНИЕ МАТЕМАТИЧЕСКИХ ВЫРАЖЕНИЙ

Одним из первых вопросов, возникающих при реализации модуля символьных преобразований, является вопрос представления математических формул в памяти компьютера. Очевидно, что от пользователя формулы могут поступать в систему только в виде строк символов. Однако преобразование формул, представленных строками, сложно и неэффективно. Поэтому используем ставшее уже классическим представление в виде бинарных деревьев, которое было адаптировано для объектно-ориентированной парадигмы программирования. Для построения бинарных деревьев реализуется иерархия классов. Ее так называемая UML (Unified Modeling Language) диаграмма показана на рис. 1.

Видно, что корнем иерархии является абстрактный класс FormulaItem, от него происходят все остальные классы. Пользуясь общепринятой терминологией, экземпляры класса FormulaItem, содержащие ссылки на другие экземпляры класса FormulaItem, будем называть ветвями, не содержащие — листьями, узлами будем называть любые экземпляры класса FormulaItem. Таким образом, экземпляры классов Number и Variable,

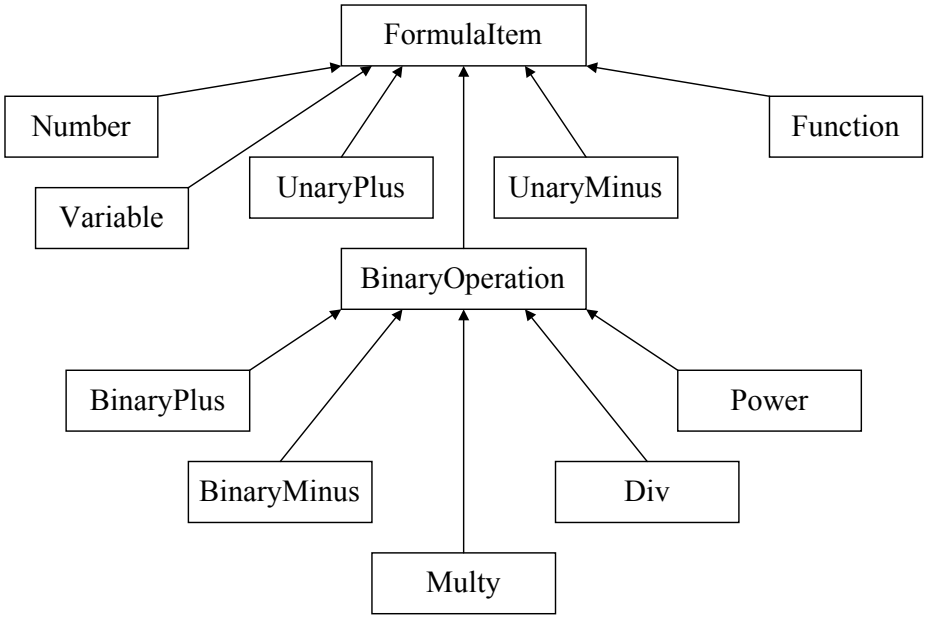

Рис. 1. Диаграмма классов для построения бинарных деревьев

представляющие числа и переменные соответственно, являются листьями, остальные — ветвями. Классы UnaryPlus и UnaryMinus (унарный плюс и унарный минус) содержат по одной ссылке на экземпляры класса FormulaItem. Виртуальный класс BinaryOperation является базовым классом для группы классов, представляющих бинарные операции: BinaryPlus (бинарный плюс), BinaryMinus (бинарный минус), Multy (умножение), Div (деление), Power (степень). Этот класс ссылается на два экземпляра класса FormulaItem, это же свойство наследуют и все производные классы. Что касается класса Function, то он содержит произвольное количество ссылок на экземпляры класса FormulaItem в зависимости от числа параметров функции. Так например, экземпляр класса Function, представляющий функцию sin(ugol) ссылается на экземпляр класса Variable, представляющий переменную ugol. На рис. 2 в качестве примера показано, как в памяти компьютера будет представляться формула расчета направления ветра sigma [1]:

```
sigma = arcsin(ske/tke*sin(fupu-uk)/(v^2+
    ske^2/tke^2-2*v*ske/tke*cos(fupu-uk))^1/2)+uk,
```

где ske — длина контрольного этапа; tke — время пролета контрольного этапа; ske/tke = w — путевая скорость; fupu — фактический условный путевой угол; uk — угол курса; fupu-uk = us — угол сноса; v — воздушная скорость; * — знак умножения; ^ — знак возведения в степень.

Узлы, помеченные символами +, -, *, / и ^, представляют бинарные операции, и являются экземплярами классов BinaryPlus, BinaryMinus, Multy, Div и Power соответственно. Узлы, помеченные символами ske, tke, fupu, uk, v — экземпляры класса Variable; 2 — класса Number; узлы arcsin, sin, cos — функции (Function).

Для преобразования формул из строкового представления в бинарное дерево необходимо использовать методы теории трансляции [2]. В модуле, выполняющем эти преобразования, можно выделить два основных компонента: лексический и синтаксический анализаторы.

Лексический анализатор

Лексический анализатор выделяет из входной строки отдельные лексемы. К лексемам относятся:

- числа — последовательности цифр, содержащие или не содержащие десятичную точку;
- переменные, представленные идентификаторами (последовательностями букв и цифр, начинающимися с буквы);
- знаки арифметических операций: +, -, *, /. ^;
- имена функций — идентификаторы, за которыми непосредственно следует левая круглая скобка;
- левая и правая круглые скобки;
- запятая.

Последовательности символов, которые не соответствуют ни одному из приведенных выше пунктов, считаются недопустимыми и отвергаются лексическим анализатором. Выделенные лексемы передаются на обработку синтаксическому анализатору.

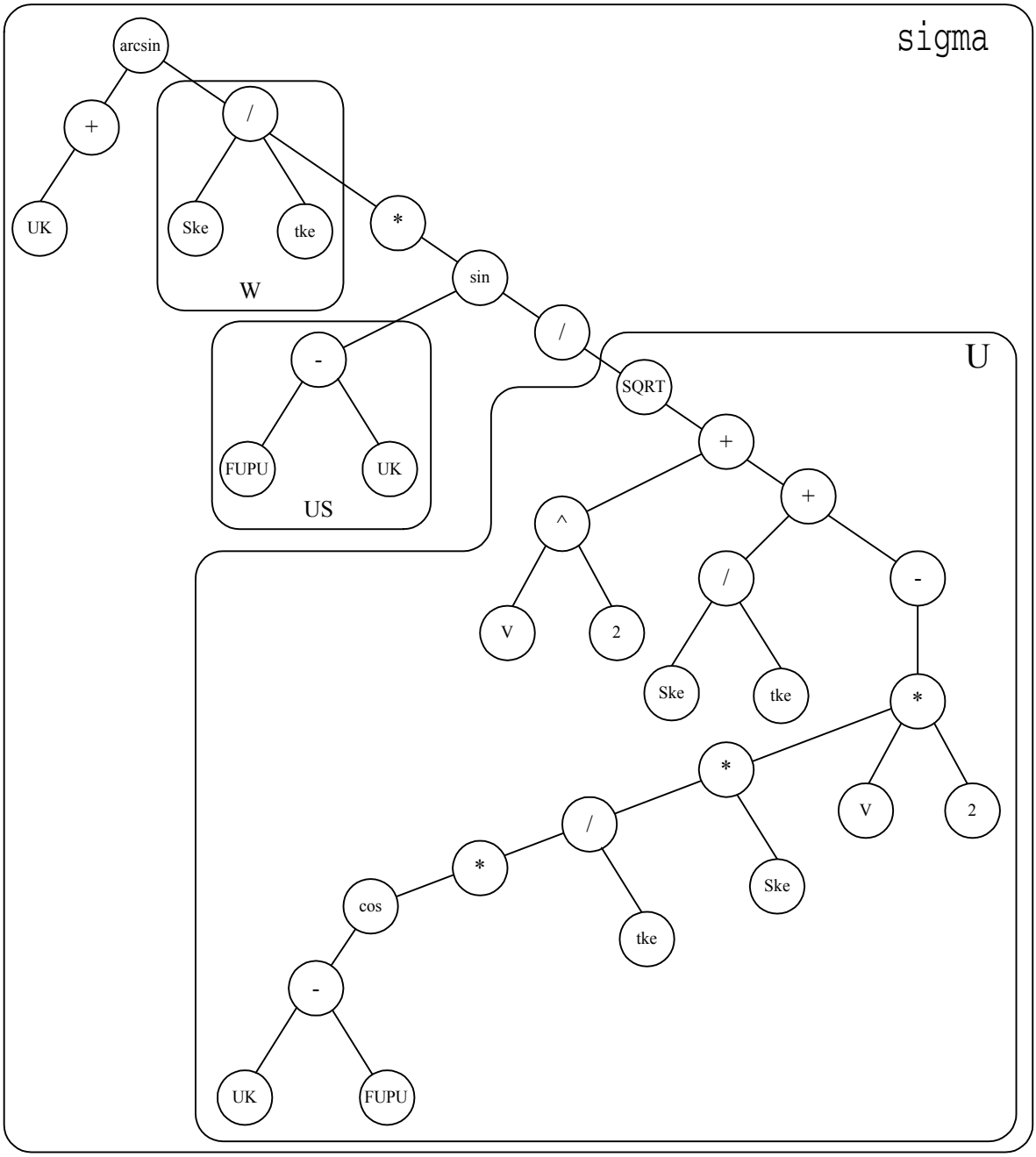

Рис. 2. Представление формулы расчета направления ветра sigma

Синтаксический анализатор

Синтаксический анализатор формирует из полученных лексем бинарное дерево, при этом он проверяет, является ли передаваемая ему последовательность лексем допустимой. Так например, строка символов 2+*v с точки зрения лексического анализатора является вполне допустимой, однако с точки зрения синтаксиса языка математических формул — ошибочной. Такие ошибки должен отслеживать синтаксический анализатор. Кроме того, он отвечает за интерпретацию отдельных лексем. Действительно, тип узла, в который будет преобразована лексема, определяется не только ее типом, но и типом предшествующей лексемы. Так например, минус в строке символов fupu-uk будет преобразован в бинарный минус, в то время как плюс в строке v^+2 в унарный плюс.

Для преобразования последовательности лек-

сем в бинарное дерево используется алгоритм, аналогичный алгоритму преобразования арифметического выражения в польскую инверсную запись. Алгоритм использует два стека: в первом стеке формируются поддеревья строящегося бинарного дерева; второй стек служит буфером для вре́менного помещения лексем, представляющих арифметические операции и левые скобки. На вход алгоритма подается математическая формула в виде строки символов. В случае если строка корректна, на выходе получаем бинарное дерево, представляющее введенную математическую формулу. Неформальное описание работы синтаксического анализатора выглядит следующим образом.

1. Получить лексему от лексического анализатора, сделать ее текущей. Если лексический анализатор обработал всю входную строку и больше не возвращает лексем — перейти к п. 6.

2. Если текущая лексема является числом или переменной — поместить ее в качестве узла в первый стек, перейти к п. 1, если же функцией — вызвать данный алгоритм для каждой подстроки, представляющей собой параметр функции, поместить поддерево, представляющее полученную функцию, в первый стек, перейти к п. 1.

3. Если текущая лексема является арифметической операцией, но не степенью, — сравнить ее приоритет с приоритетом лексемы на вершине второго стека (приоритеты лексем указаны в таблице); если приоритет больше или стек пуст — поместить текущую лексему во второй стек, перейти к п. 1. В противном случае снять лексему с вершины второго стека, выполнить над ней п. 7, после чего вернуться к началу п. 3.

4. Если текущая лексема левая скобка или степень — поместить ее во второй стек, перейти к п. 1.

5. Если текущая лексема правая скобка — последовательно снимать с вершины второго стека лексемы и выполнять над ней п. 7, пока на вершине второго стека не окажется левая скобка. Снять с вершины второго стека левую скобку, перейти к п. 1.

6. Пока второй стек не пуст, снимать с его вершины лексемы и выполнять над ними п. 7. Когда второй стек окажется пустым — завершить алгоритм.

7. Снять с вершины первого стека один или два узла (в зависимости от того, над каким узлом выполняется пункт: унарным или бинарным), сделать эти узлы дочерними узла, образованного из лексемы, над которой выполняется пункт, и поместить этот узел в первый стек.

Необходимо пояснить, почему лексема степени обрабатывается не так, как остальные лексемы арифметических операций (см. п. 3, 4 алгоритма). Дело в том, что все арифметические операции

Приоритеты лексем

Узел	Приоритет
+ (бинарный), - (бинарный)	0
*, /, + (унарный), - (унарный)	1
^	2

с равными приоритетами выполняются слева направо и только несколько последовательных возведений в степень — справа налево. Поэтому лексема, представляющая степень, не вытесняет из стека другие лексемы степени. Отдельно должны обрабатываться и арифметические операции в скобках, поскольку скобки повышают их приоритет. В процессе работы алгоритма в первом стеке формируются поддеревья бинарного дерева. Когда алгоритм заканчивает свою работу, на вершине первого стека оказывается корень бинарного дерева.

ПРИМЕР

Обучаемому предлагается выполнить тестовое задание: определить направление и скорость ветра по углу сноса и путевой скорости, измеренным на контрольном этапе.

Исходные данные:

УК [град] — uk; V [км/ч] — v; $S_{КЭ}$[км] — ske; ФУПУ [град] — fupu; $t_{КЭ}$[мин] — tke.

Реализуемая программа должна произвести диагностику действий обучаемого при решении тестового задания и продемонстрировать те из них, в которых были допущены ошибки.

1. УС — us, $\text{УС} = \text{ФУПУ} - \text{УК}$;

2. W — w, $W = \dfrac{S_{КЭ}}{t_{КЭ}}$;

3. $U = \pm\sqrt{V^2 + W^2 - 2 \cdot V \cdot W \cdot \cos \text{УС}}$, + при $W > V$, – при $W < V$;

4. $\text{sigma} = \arcsin\left(\dfrac{W \cdot \sin \text{УС}}{U}\right) + \text{УК}$;

5. $\text{sigma} = \begin{cases} \text{sigma} - 360^\circ & \text{при sigma} > 0, \\ \text{sigma} + 360^\circ & \text{при sigma} < 0. \end{cases}$

Коллизия 1.

Действия (1) и (2) можно выполнять в любой последовательности, и при этом в каждом из них возможны ошибки.

Коллизия 2.

Ошибки в определении U могут быть допущены как из-за ошибок в действиях (1) и (2), так и при неправильном определении знака перед корнем.

Коллизия 3.

Погрешность в определении параметра sigma может быть допущена как из-за ошибок в действиях (1), (2) и (3), так и при неправильном определении знака sigma.

СПИСОК ЛИТЕРАТУРЫ

1. *Мамаев В.Я., Синяков А.Н., Петров К.К., Горбунов Д.А.* Воздушная навигация и элементы самолетовождения. Учеб. пособие. СПбГУАП, 2002. 256 с.
2. *Ахо А.-В., Сети Р., Ульман Дж.-Д.* Компиляторы: принципы, технологии и инструменты. Пер. с англ. М.: Издательский дом "Вильямс", 2003. 768 с.

Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП)

Материал поступил в редакцию 9.11.2005.

KNOWLEDGE CONTROL OF TRAINEES PERFORMING SYMBOLIC TRANSFORMATIONS IN AN INTELLIGENT TRAINING SYSTEM

V. Mamaev, K. Petrov

State University of Aerospace Instrumentation, Saint-Petersburg

During training on a simulator of the operator of an aircraft it in solving navigational tasks, it is necessary to control his knowledge. An algorithm of formula transformation is proposed, allowing one to supervise correctness of answers of the trainee when he fulfils a test task given to him in maximum free form.