РАЗДЕЛ III

ОБУЧАЮЩИЕ СИСТЕМЫ

Использование AICC Script для задания последовательности прохождения учебного курса

О. С. Естехин

Язык AICC Script описывает операторы и правила составления логических выражений, которые могут быть использованы для определения факта прохождения элемента учебного курса в зависимости от состояния его подэлементов (используется в модели AICC CMI001, [1]) и для определения возможности вхождения в элемент учебного курса в зависимости от состояния других элементов (используется в моделях AICC CMI001 [1] и ADL SCORM [3]).

Разработкой языка AICC Script занимается организация Aviation Industry CBT Committee (AICC, [4]).

Синтаксис и правила AICC Script описаны в документе AICC CMI Guidelines for Interoperability [1, параграф 4.2.3, страницы 140–142; параграф 10.6, страница 236] в виде текстового описания, а также BNF правил. ADL SCORM [3] использует AICC Script, но ссылается на более старую версию документа AICC CMI Guidelines for Interoperability [2].

Ниже представлено описание языка AICC Script, в котором исправлены ошибки оригинального описания, данного в документе [1]. Список этих ошибок, а также мест, допускающих неоднозначную трактовку, будет приведен ниже.

Для использования AICC Script для определения возможности прохождения того или иного урока в зависимости от результатов прохождения других уроков учебный курс представляется в виде набора узлов, которыми могут являться как отдельные уроки, так и объединения из нескольких уроков. Каждый узел обладает идентификатором, а так же специальным свойством, которое называется пререквизиты и содержит логическое выражение, результат вычисления которого определяет, можно ли проходить данный узел. В исходном состоянии статус каждого узла в учебном курсе

Таблица 1

Статус	Описание
not attempted	Пользователь не приступал к изучению данного узла. Статус узла принимает это значение по умолчанию.
browsed	Пользователь просмотрел содержимое узла (урока), но не проходил каких-либо тестов.
incomplete	Пользователь частично изучил содержимое узла (урока), и, возможно, приступил к прохождению тестов.
failed	Пользователь полностью изучил содержимое узла (урока), но прошел тестирование с отрицательным результатом.
passed	Пользователь полностью изучил содержимое узла (урока) и прошел тестирование с положительным результатом.
completed	Пользователь полностью изучил содержимое узла (урока), но не проходил тестов.

равен “not attempted”. Пользователь начинает изучение курса с определенного исходного урока. В результате прохождения какого-либо урока статус этого урока принимает значение, соответствующее результатам прохождения, в результате чего пререквизиты каких-либо других (узлов) уроков могут принять значение «истина», что означает, что пользователю становятся доступны другие узлы (уроки).

Логическое выражение на языке AICC Script состоит из идентификаторов узлов, объединенных при помощи операторов. В процессе вычисления выражения идентификаторы узлов заменяются на статус соответствующего узла.

Статус узла может принимать значения, перечисленные в табл. 1.

Логическое выражение на языке AICC Script состоит из термов, разделенных бинарными операторами. В качестве термина может выступать идентификатор курса, оператор сравнения с определенным статусом, оператор отрицания, оператор множества, а так же другое выражение в скобках. Бинарными операторами являются оператор И (логическое И) и оператор ИЛИ (логическое ИЛИ).

В табл. 2 описаны синтаксис и правила вычисления бинарных операторов.

В табл. 3 описаны синтаксис и правила вычисления остальных конструкций языка.

Таблица 2

Оператор	Символ	Синтаксис	Описание
И (and)	&	E1 & E2	Логическое И. Возвращает значение «истина» тогда и только тогда, когда значение выражений E1 и E2 равно «истина».
ИЛИ (or)		E1 E2	Логическое ИЛИ. Возвращает значение «истина», если значение хотя бы одного из выражений E1 и E2 равно «истина».

Таблица 3

Оператор	Символ	Синтаксис	Описание
ID	ID	ID	Идентификатор узла может служить в качестве самостоятельного терма. Возвращает значение «истина», если статус соответствующего узла равен “passed” или “completed”, и значение «ложь» во всех остальных случаях.
равно (equals)	=	ID = Status	Оператор сравнения применяется только для проверки статуса узла на равенство определенному значению. Этот оператор нельзя использовать для сравнения статусов узлов между собой или сравнения каких-либо других величин. Возвращает значение «истина», если статус узла с идентификатором ID в равен указанному статусу. Значение статуса может быть записано как в полной форме (“not attempted”, “browsed”, “incomplete”, “failed”, “passed”, “completed”), так и в сокращенной форме в виде первой буквы (“n”, “b”, “i”, “f”, “p”, “c”). Значение статуса может быть записано в любом регистре, а также заключено в одинарные или двойные кавычки. Если кавычки не используются, то значение “not attempted” должно быть записано ровно с одним пробелом между словами.

Продолжение таблицы 3

Оператор	Символ	Синтаксис	Описание
НЕ (not)	~	~E	Логическое НЕ. Возвращает значение «истина», если значение выражения E «ложь», и значение «ложь», если значение выражения E «истина». В качестве выражения E может выступать идентификатор элемента курса, множество или выражение, заключенное в круглые скобки. Для отрицания оператора сравнения необходимо заключить его в круглые скобки.
скобки (precedence)	()	(E)	Изменяет порядок вычисления выражения. Возвращает значение выражения, заключенного в скобки.
множество (set)	N * { }	N*{E1, E2, ... }	Вычисляет по отдельности значения выражений, перечисленных через запятую внутри фигурных скобок, и возвращает «истина», если значение N или более выражений равно «истина».

При вычислении выражения используется следующий приоритет операторов (см. табл. 4).

Таблица 4

Оператор	Порядок вычисления
=	1
()	2
X * { }	3
~	4
&	5
	6

Ниже приведены EBNF правила, полностью описывающие синтаксис языка.

Правило Expr определяет общий вид выражения на языке AICC Script.

`Expr ::= S? Term S? (binaryOperator S? Term S?)*`

Следующее правило определяет возможные бинарные операторы.

`binaryOperator ::= '&' | '|'`

Следующее правило определяет возможные термы.

`Term ::= ID | equalsTerm | negationTerm | setTerm
| bracketTerm`

Правило `Name` определяет идентификатор узла курса. В качестве начального символа идентификатора могут использоваться символы, относящиеся к Unicode категориям Ll, Lu, Lo, Lt или Nl, а также символы `'_'` и `'.'`. В качестве остальных символов имени могут использоваться символы, относящиеся к Unicode категориям Mc, Me, Mn, Lm или Nd, а также символы `'-'` и `'.'`. Правило `Name`, определенное таким образом, является подмножеством правила XML `Name` с небольшими ограничениями.

`Name ::= startChar nameChar*`
`startChar ::= Ll | Lu | Lo | Lt | Nl | '_' | '.'`
`nameChar ::= Mc | Me | Mn | Lm | Nd | '-' | '.'`

Правило `equalsTerm` определяет оператор сравнения.

`equalsTerm ::= Name S? '=' S? Status`
`Status ::= statusName |
 (" " statusName " ") |
 (' ' statusName ' ')`
`statusName ::= statusP | statusC |
 statusF | statusI | statusB | statusN`
`statusP ::= 'p' | 'passed'`
`statusC ::= 'c' | 'completed'`
`statusF ::= 'f' | 'failed'`
`statusI ::= 'i' | 'incomplete'`
`statusB ::= 'b' | 'browsed'`
`statusN ::= 'n' | 'not attempted'`

Правило `negationTerm` определяет оператор отрицания.

`negationTerm ::= '~' S? (Name | setTerm | bracketTerm)`

Правило `setTerm` определяет оператор множества.

`setTerm ::= Number S? '*' S?
 '{' S? Expr S? (',' S? Expr S?)* '}'`
`Number ::= ('0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9')+`

Правило `bracketTerm` определяет оператор изменения порядка вычисления.

`bracketTerm ::= '(' S? Expr S? ')'`

В табл. 5 даны примеры выражений с описанием их значения.

Таблица 5

Выражение	Описание
A1	Возвращает значение «истина», если статус узла с идентификатором A1 равен “passed” или “completed”, и значение «ложь», если статус равен “not attempted”, “browsed”, “incomplete” или “failed”.
A1 = passed A1 = Passed A1 = PASSED A1 = p A1 = P	Возвращает значение «истина», если статус узла с идентификатором A1 равен “passed”, и значение «ложь» в любом другом случае.
~A1	Возвращает значение «истина», если статус узла с идентификатором A1 равен “not attempted”, “browsed”, “incomplete” или “failed”, и значение «ложь», если статус равен “passed” или “completed”.
~(A1 = passed) ~(A1 = Passed) ~(A1 = PASSED) ~(A1 = P) ~(A1 = p)	Возвращает значение «истина», если статус узла с идентификатором A1 равен “not attempted”, “browsed”, “incomplete”, “failed” или “completed”, и значение «ложь», если статус равен “passed”.
A1 & A2	Возвращает значение «истина», если статусы обоих узлов с идентификаторами A1 и A2 равны “passed” или “completed”.
A1 = passed & A2 = passed	Возвращает значение «истина», если статусы обоих узлов с идентификаторами A1 и A2 равны “passed”.
A1 A2	Возвращает значение «истина», если статус хотя бы одного из узлов с идентификаторами A1 и A2 равен “passed” или “completed”.
A1 = passed A2 = passed	Возвращает значение «истина», если статус хотя бы одного из узлов с идентификаторами A1 и A2 равен “passed”.
A1 & A2 A3	Возвращает значение «истина», если статусы обоих узлов с идентификаторами A1 и A2 равны “passed” или “completed” или статус узла с идентификатором A3 равен “passed” или “completed”.
A1 & (A2 A3)	Возвращает значение «истина», если статус узла с идентификатором A1 равен “passed” или “completed” и статус хотя бы одного из узлов с идентификаторами A2 и A3 равен “passed” или “completed”.

Продолжение таблицы 5

Выражение	Описание
$1 * \{A1, A2, A3\}$	Возвращает значение «истина», если статус хотя бы одного из узлов с идентификаторами A1, A2 или A3 равен “passed” или “completed”.
$2 * \{A1, A2, A3\}$	Возвращает значение «истина», если статус любых двух из узлов с идентификаторами A1, A2 или A3 равен “passed” или “completed”.
$3 * \{A1, A2, A3\}$	Возвращает значение «истина», если статус всех трех узлов с идентификаторами A1, A2 или A3 равен “passed” или “completed”.

Правило S определяет один или более символов пробела.

$S ::= (\#x20)^+$

Ниже перечислены ошибки, а также места, допускающие неоднозначную трактовку, в описании языка AICC Script в том виде, в котором оно дано в документе AICC CMI Guidelines for Interoperability [1].

1. Не указано, следует ли заключать название статуса в операторе сравнения в кавычки. Так как в названии статуса “not attempted” содержится пробел, отсутствие кавычек может привести к неправильному пониманию выражения в целом.
2. Не определен регистр, в котором можно записывать названия статусов, что может привести к несовместимости между различными реализациями языка AICC Script и пользовательскими приложениями.
3. Определен оператор вида $\{A1, A2, \dots\}$, т. е. оператор множества без указания количества элементов, которые должны иметь значение «истина», чтобы значение оператора было «истина». Для этого оператора не указано, как именно вычислять его значение и каков его приоритет по отношению к другим операторам. Также определен оператор «,» (запятая), т. е. разделитель элементов множества. Очевидно, эти две конструкции не являются самостоятельными операторами, а являются частью синтаксиса оператора множества.
4. Нотация, используемая для описания синтаксиса логического выражения, не соответствует какой-либо из общепринятых нотаций (BNF, ABNF или EBNF).
5. Правило для логического выражения в целом допускает выражения вида «A1 A2 A3», т. е. последовательность из нескольких термов без каких-либо бинарных операторов между ними. Подобная строка не может являться логическим выражением.

6. Правило для оператора изменения порядка вычисления (круглых скобок) допускает выражения вида «(A1, A2, A3)», т. е. последовательность из нескольких термов, разделенных запятыми, заключенная в круглые скобки. Подобная строка так же не может являться логическим выражением.
7. Правило для оператора множества допускает выражения вида «{A1, A2, A3}», т. е. оператор множества без указания количества элементов, которые должны принять значение «истина». Поскольку не указано, как вычислять подобное выражение, данное правило должно быть изменено таким образом, чтобы ему удовлетворяли только строки, которые содержат указание требуемого количества элементов.
8. Правило для оператора множества допускает только однозначное число (т. е. число от 0 до 9) для указания количества элементов, которые должны принять значение «истина». Необходимо расширить это правило, чтобы оно допускало использование чисел, больших 9.
9. Не указано, в каких местах логического выражения допустимо использовать пробелы, что может привести к несовместимости между различными реализациями языка AICC Script и пользовательскими приложениями.

Описание этих проблем было передано в AICC, и, возможно, будет учтено в следующей версии языка AICC Script.

Литература

1. AICC CMI Guidelines for Interoperability. CMI001. Revision 4.0. August 16, 2004 (www.aicc.org).
2. AICC CMI Guidelines for Interoperability. CMI001. Revision 3.4. October 23, 2000 (www.aicc.org).
3. ADL SCORM. The SCORM Content Aggregation Model. Version 1.2. October 1, 2001 (www.adlnet.org).
4. Aviation Industry CBT Committee (www.aicc.org).