Инженерный вестник Дона, №4 (2018)
ivdon.ru/ru/magazine/archive/n1y2018/5302

Особенности создания встроенного контекстно-доопределяемого языка для интеллектуальных обучающихся агентов (предсказание потребления ресурсов в сфере ЖКХ)

В.Н. Подсвиров

Южный федеральный университет, Таганрог

Аннотация: В данной работе рассматриваются вопросы создания и использования контекстно-доопределяемых машинных языков для разработки новых информационных технологий. Рассмотрены вопросы реализации таких языков для интеллектуальных агентов (ИОА), которые используются для решения задач предсказания потребления ресурсов в сфере ЖКХ. Особое внимание уделяется синтезу контекстно-доопределяемого языка, создаваемого для интеллектуального агента, реализующего функции предсказания потребления ресурсов в сфере ЖКХ. Основная идея синтеза встроенного языка состоит в придании ему средств выделения частей алгоритма компонента и организации изменяемого в процессе вычислений соответствия между такой частью и контекстом. Благодаря этой связи исходный алгоритм может изменяться прямо или косвенно в процессе жизнедеятельности ИОА. В результате получается необходимое адаптационное изменение алгоритмов на основе накопленных знаний.

Ключевые слова: алгоритмические языки, встроенные языки, контекстно-доопределяемые языки, интеллектуальные агенты, машинные языки.

Введение

Агентом [1] является все, что может рассматриваться как воспринимающее свою среду с помощью датчиков и воздействующее на эту среду с помощью исполнительных механизмов. Рассмотрим структуру обучающегося агента (Рис.1).

Обучающийся агент может состоять из производительного элемента (ПЭ), определяющего действия, и обучающего элемента (ОЭ), который модифицирует производительный элемент в целях получения лучших решений. На сегодняшний день существуют различные типы ОЭ. На структуру ОЭ оказывают влияние компоненты ПЭ, обратные связи обучения, способы представления компонентов. Предлагается система контроля со стороны ОЭ, которая может быть полезна для решения задач обучения.

Рассмотрим структуру обучающегося агента с точки зрения использования встроенного языка и типичных задач, решаемых этими компонентами (Рис.1).

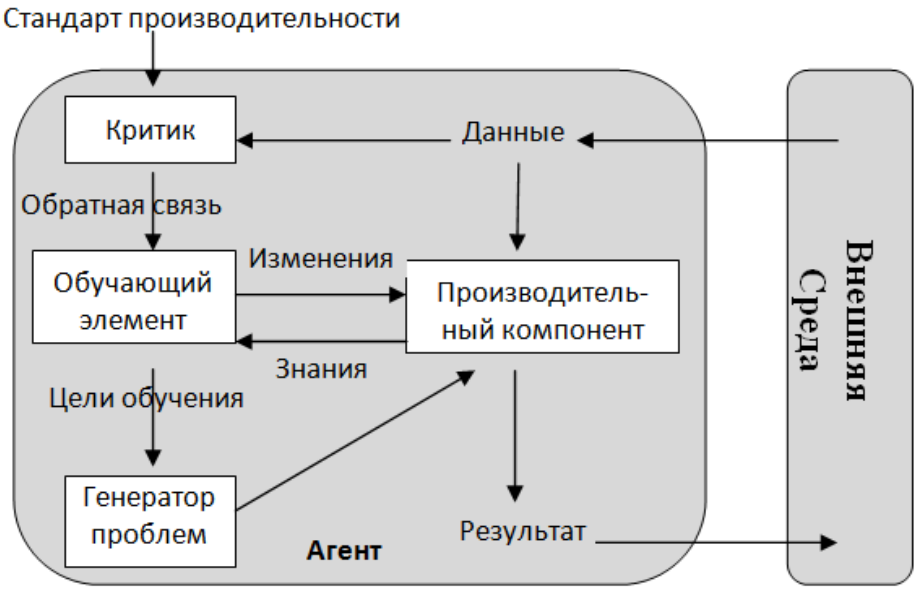

Рис. 1. Обучающийся интеллектуальный агент

Рассмотрим типовые компоненты ПЭ:

1. Средства прямого отображения условий в действия. Между внешними условиями (запрос для производства предсказания, показания приборов учета, изменение температуры и т.д.) и формируемыми действиями существует связь, которая реализуется условно неизменными частями алгоритмов обработки и условно изменяемыми частями.

2. Средства логического вывода релевантных свойств мира из последовательности результатов восприятия. Они фактически определяются входным потоком данных и состоянием ИАО. Как и в предыдущем пункте могут быть описаны условно неизменными частями алгоритмов обработки и условно изменяемыми частями.

3. Информация о том, как развивается мир и какие результаты возможных действий могут быть получены агентом. Для задач предсказания это прежде всего работа с архивом накопленных данных. В простейшем случае

она состоит в статистической обработке уже накопленных данных. На основе этой обработки строится модель предсказаний. Статистическую обработку желательно вынести за рамки непосредственной работы ИАО, чтобы не повторять многократно одни и те же действия и сократить общее время функционирования системы. Результаты, получаемые на основе такой обработки, строятся алгоритмами, состоящими из условно неизменных и условно изменяемых частей.

4. Информация о полезности, которая показывает, насколько желательными являются те или иные состояния мира. Производится алгоритмами, оценивающими полезность. Вместе с тем она может быть внесена извне уже в готовом виде (оценка полезности функционирования ИАО человеком). Структура этих алгоритмом может быть представлена также как и в предыдущих случаях условно неизменными частями алгоритмов обработки и условно изменяемыми частями.

5. Информация о ценности действий, показывающая желательность действий. Производится алгоритмами, оценивающими ценность действий. Вместе с тем она может быть внесена извне уже в готовом виде (оценка ценности действий ИАО человеком). Структура этих алгоритмом может быть представлена также как и в предыдущих случаях условно неизменными частями алгоритмов обработки и условно изменяемыми частями.

6. Цели, описывающие классы состояний, достижение которых максимизирует полезность для агента. Производится алгоритмами, определяющими цели. Вместе с тем информация о целях может быть внесена извне уже в готовом виде (определение целей ИОА человеком). Структура этих алгоритмом может быть представлена также как и в предыдущих случаях условно неизменными частями алгоритмов обработки и условно изменяемыми частями.

На основе вышеприведенных рассуждений [2-11] мы можем сделать выводы:

1. Каждая компонента ИОА может быть условно представлена в виде трех составляющих, реализуемых в виде:

- жестко заданного алгоритма (пишется на обычном языке программирования и дальнейшему изменению не подлежит);
- условно постоянной части алгоритма (реализуется средствами встроенного языка, редко изменяется или не изменяется вовсе);
- условно изменяемой части алгоритма (реализуется средствами встроенного языка, часто изменяется, может изменяться извне или составляющими ИОА);

2. Перед запуском ИОА необходимо предусмотреть автономно работающие программы статистической обработки. Предпочтение следует отдавать жестким алгоритмам (не изменяются в процессе жизнедеятельности ИОА, реализуются обычным языком программирования), которые обеспечивают максимальную скорость обработки. Для нормального функционирования системы предсказаний необходимо периодически (раз в сутки) подновлять базу данных ЖКХ и запускать соответствующие алгоритмы статистической обработки.

3. Условно изменяемые алгоритмы статистической обработки (реализуются средствами встроенного языка) следует, по возможности, минимизировать и создавать прежде всего для моделирования жестких алгоритмов с последующим переводом на обычный язык программирования и включения в перечень общих алгоритмов статистической обработки.

4. С учетом предыдущих пунктов, каждая составляющая ИОА может быть снабжена условно изменяемым алгоритмом, реализованным средствами встроенного языка. Возможности изменения такого алгоритма определяются используемым языком.

Для реализации встроенного языка предлагается использовать язык, принадлежащий классу контекстно-доопределяемых языков (КДЯ).

Под контекстно-доопределяемым языком понимаем язык, способный изменяться под воздействием добавляемого контекста. При этом меняется не только трактовка некоторых предложений языка, но и появляются новые синтаксические и семантические конструкции.

В общем случае каждая из структурных составляющих ИОА может воздействовать на другие составляющие. То есть с точки зрения структурного воздействия можно говорить о полном графе связей между элементами ИОА. Надо обеспечить такую возможность, которая на практике может быть реализована частично. Это одно из требований к языку и реализующей структуре ИОА.

Предлагается реализовать это требование введением в структуру каждого элемента ИОА механизма, который в чем-то напоминает делегаты языка C#. Отличие состоит в том, что внедренные скрипты будут реализоваться интерпретатором встроенного в ИОА языка (ВЯ). В общем случае таких скриптов в каждом элементе ИОА может быть по числу оставшихся элементов ИОА за вычетом текущего. Порядок их срабатывания определяется жестко заданным алгоритмом, реализованным обычным образом. Конкретный состав и порядок срабатывания определяется на этапе создания ИОА и непосредственно зависит от конкретной задачи, стоящей перед разработчиком программного обеспечения (ПО).

Гибкость поведения ИОА в этом случае будет определяться, прежде всего, алгоритмами, реализующими такие скрипты.

Каждый скрипт должен обеспечивать поддержку условно постоянной части (УПЧ) алгоритма и условно изменяемой части (УИЧ) алгоритма.

Для ускорения процесса интерпретации можно использовать компилирующий интерпретатор. Проблема в его реализации состоит в том, что неко-

торые части откомпилированного ВЯ могут заменяться на новые в процессе исполнения скрипта, и могут заменяться самим скриптом в процессе жизнедеятельности ИОА.

Если мы работаем с исходным текстом, которые представляет собой совокупность строк, состоящих их символов, то все более или менее понятно. Задача состоит в изменении исходного текста скрипта.

Если мы изменяем код, созданный компилирующей частью, то главным предъявляемым требованием становится наличие механизма оперативной замены одних частей кода на другие.

Предлагается в качестве такого кода использовать граф типа дерево. Исполнение такого графа начинается с корня. Имеется прямой путь графа (ППГ) от одной вершины к другой. Этот путь определяется последовательностью операторов исходного текста. Кроме этого, имеется возможность циклического или условного перехода. То есть фактически существует два графа. Один граф содержит все вершины и связи между ними и является графом типа дерева. Он является основным структурным элементом исполняемого скрипта. Другой граф содержит те же вершины, что и первый, но определяет условные и циклические переходы. Это соответствует условным и циклическим операторам ВЯ. Описание второго графа содержится в узлах первого, также как и описание связей первого графа.

Принципиальное отличие состоит в том, что наличие или отсутствие связей второго графа определяется во время выполнения скрипта. Кроме прямого пути графа могут иметься ответвления, которые рассматриваются как подграфы. Для каждой вершины имеется информация о прямом пути и об ответвлениях. Порядок обработки ответвлений определяется типом вершины.

Пусть ответвление представляет собой выполнение некоторых арифметических или вычислений любого другого типа. Тогда при достижении данной вершины по ППГ происходит выполнение ответвлений в соответствии с

типом оператора, который реализует данная вершина. В случае арифметических вычислений это будет поддерево разбора строки ВЯ, где в качестве вершин будут фигурировать операции и операнды исходного арифметического выражения. После выполнения ответвления процесс продолжится в соответствии с ППГ.

Таким образом, каждая вершина ППГ имеет свой тип и алгоритм обработки подграфа (подграфов), непосредственно связанного с ней.

Условные и циклические вершины имеют возможность изменить последовательность выполнения процесса обработки по ППГ и передать управление на предшествующую и последующую область ППГ в рамках текущего скрипта или другому скрипту, который может быть скомпилирован отдельно от текущего.

Мы рассмотрели реализацию обычных алгоритмов средствами поддержки ВЯ. Если бы наши задачи исчерпывались такими изобразительными средствами, то нам не надо было изобретать что-то новое. Достаточно было бы воспользоваться традиционными языками, а не КДЯ.

КДЯ предоставляют нам возможность заменить или изменить любую часть графа, начиная с вершины, которая прилежит ППГ или в рамках ответвления. Такое действие соответствует изменению части скрипта на ВЯ.

Если мы заменим некоторую вершину, принадлежащую ППГ, на новую вершину, с примыкающими к ней ответвлениями, то получим принципиальные изменения реализуемого алгоритма.

Пусть нам необходимо заменить исполнение арифметического выражения на другое. Арифметическое выражение определяется оператором арифметических вычислений. Для этого достаточно иметь скомпилированное отдельно дерево, соответствующее новому арифметическому выражению. Значение ссылки предыдущей вершины ППГ (указывает на текущую, заменяемую вершину) изменяем на новое, которое теперь указывает на новую

вершину. Значение ссылки добавленной вершины заменяем значением, указывающим на следующую вершину ППГ. Граф принимает новый вид.

Декомпиляция такого скрипта покажет, что одно арифметическое выражение было заменено на другое.

Каждая вершина графа представляет собой объект, который связан с объектами такого же типа. Это похоже на однонаправленные списки, реализуемые объектами с соответствующими ссылками. Принципиальное отличие состоит в том, что имеется не только последующий элемент, который соответствует ППГ, но и некоторое количество ответвлений, которые обрабатываются в соответствии с типом вершины, до перехода на следующую вершину ППГ.

В общем случае вместо любой вершины, принадлежащей ППГ, мы можем вставить ссылку на полноценный граф, который содержит свой ППГ. По окончанию добавленного подграфа управление должно быть передано следующей вершине ППГ основного графа. Таким образом, можно заменить одну вершину на целый алгоритм, содержащий условные и циклические элементы. Внутри внедренного алгоритма могут быть свои внедрения и так далее. В результате общая сложность простого с виду алгоритма может увеличиваться по мере производства вычислений.

В обычном языке общий процесс вычислений можно описать с помощью двух типов действий: последовательное выполнение команд и изменение последовательности команд. КДЯ предоставляет помимо этого еще и возможность изменять содержание этих действий, то есть можно изменять сами выполняемые команды. Изменения возможны самые разнообразные: от замены одной команды на другую до замены групп команд на другие группы команд.

При замене последовательно выполняемых команд обычно проблем не возникает, так как совместимость последовательно выполняемых команд

контролировать довольно просто. Наибольшие проблемы возникают с командами, которые изменяют последовательность выполнения текущего набора команд. Проблема состоит в том, что могут быть получены ссылки (переходы) на несуществующие места в памяти.

На практике эту проблему можно решить, использовав отсроченную компиляцию ссылок условных операторов и операторов цикла.

Основные операторы реализованного ВЯ приведены в Таблице 1.

Таблица №1

Основные операторы встроенного языка

Имя оператора	Параметры	Описание
DCL	<тип>, <переменная>	Объявляет локальную переменную указанного типа
SET	<арифметическое выражение>	Исполняет арифметическое выражение
IF IFELSE IFEND	<логическое выражение>	В случае истинности выражения выполняет последовательность операторов от IF до IFEND (или до IFELSE при наличии) , в противном случае от IFELSE до IFEND. Конструкция всегда заканчивается IFEND.
FOR FOREND	<арифметическое выражение1>; <логическое выражение>; <арифметическое выражение2>	Выполняет тело цикла в случае истинности логического выражения. Тело цикла располагается между FOR и FOREND.
CHANGE	<метка заменяемого оператора> := <имя вставляемой группы>	Подменяет оператор, помеченный указанной меткой на ранее откомпилированный скрипт.
STOPGROUP	<номер группы>	Запрещает исполнение операторов указанной

		группы
STARTGROUP	<номер группы>	Разрешает исполнение операторов указанной группы
SLEEPSTATE-MENT	<имя оператора>	Предупреждающий символ заменяется символом комментария. Оператор не выполняется до его активации оператором.
UPSTATE-MENT	<имя оператора>	Предупреждающий символ временно замененный на символ комментария восстанавливается.
OUT	<арифметическое выражение>	Выводит рассчитанное предсказание

Для рассматриваемого языка используется правило: один оператор – одна строка. Каждый оператор может быть помечен меткой, которые могут быть использованы операторами CHANGE и другими операторами ВЯ, которые в данной таблице не приведены по техническим причинам.

Упрощенный синтаксис строки выглядит следующим образом:

[<метка оператора>:] %[<номер группы>]<имя оператора> <параметры оператора>

В конце строки опущена конструкция, которая реагирует на нештатные ситуации типа деления на ноль, недопустимая ссылка и т.д. Она напоминает оператор catch языка C# и позволяет запустить скрипт- обработчик непредвиденных ситуаций.

Метка помечает оператор. Далее используется предупреждающий символ % и десятичный номер группы. Предупреждающий символ обеспечивает с одной стороны четкое различие между операторами и другими конструкциями. Он может быть заменен динамически на другой предупреждающий символ прямо в процессе выполнения скрипта. Обычно такая замена проис-

ходит на символ комментария. Получается как бы спящий оператор, который может быть активирован в любой момент. Позволяет наглядно представлять действия, которые сейчас активны или неактивны.

Номер группы может отсутствовать, тогда такой оператор выполняется всегда. Если указан номер группы, то активность всей группы можно менять операторами STOPGROUP и STARTGROUP. При этом операторы целой группы становятся спящими или пробуждаются.

В зависимости от целей компилятора операторы группы могут быть исключены из исходного текста скрипта или включены в него. Если операторы включены компилятором в исполняемый код, то операторы STOPGROUP и STARTGROUP управляют условием их срабатывания.

Задачи предсказания потребления требуют корректного и быстрого взаимодействия с базой данных, в которой содержится необходимая для предсказаний информация. При этом желательно упростить сам процесс до минимально возможного уровня.

Предлагается использовать встроенные переменные, которые обеспечивают неявное обращение к базе данных, используя жесткие алгоритмы доступа. Основной набор таких переменных приведен в таблице 2.

Таблица №2

Основной набор встроенных переменных

Имя переменной	Назначение
OLD_YEAR	Определяет среднее потребление за год в течение исторического периода. Предполагается, что исторический период определяет потребителем при формировании запроса.
OLD_YEAR.MAX	Максимальное среднегодовое значение за исторический период
OLD_YEAR[]	Определяет информацию потребления для текущего счетчика ресурса по годам. Нулевой индекс соответствует последнему году исторического

	периода. Используется как объект или как среднегодовое значение потребления за указанный индексом год.
OLD_YEAR.LENGTH	Количество доступных исторических лет.
OLD_YEAR[].MONTH.MAX	Максимальное месячное потребление в указанном году.
OLD_YEAR[].MONTH[]	Определяет информацию для выбранного года и месяца. Используется как объект или как среднемесячное значение.
OLD_YEAR[].MONTH[].DAY[]	Определяет информацию для выбранного года, месяца и дня.
OLD_YEAR[].MONTH[].LENGTH	Число дней в месяце.

Имена встроенных переменных предваряются предупреждающим символом $,- это обеспечивает ускоренную компиляцию. В таблице приведен неполный список. Отсутствующие переменные образуются аналогичным образом и представляют собой результат статистической обработки и иной предварительной обработки. Как правило, соответствующие им значения уже содержатся в базе данных и вычисляются на этапе формирования базы данных один раз. Такой подход сокращает общее время вычислений.

Подводя итог, отметим, что предлагаемый подход подразумевает типовое внедрение в функцию обратной связи компонентов алгоритмов обучения. Технически это реализуется средствами КДЯ путем выделения контекста, определяющего вариации как средство обучения, так и алгоритмов варьирующих эти средства. То есть речь идет об адаптации изменяющих (обучающих) функций к постоянно изменяющимся внешним условиям, в которых существует интеллектуальный агент. В принципе процесс внедрения изменяющихся составляющих в ранее написанные изменяющиеся составляющие может быть повторен неопределенное количество раз. Естественно очередное внедрение такого рода определяется целесообразностью и ограничивает-

ся сложностью получаемого общего алгоритма функционирования интеллектуального агента.

Существует возможность простого вычленения контекста из функций, варьирующих обучающие составляющие, что и позволяет производить гибкую вариацию алгоритмов таких функций. Использование такой возможности может быть не всегда оправданно, но для интеллектуальных агентов со сложным поведением могут оказаться эффективным решением возникающих проблем управления.

Поскольку предлагаются технические приемы внедрения таких средств, а не сами средства, то все ранее полученные разработки в области интеллектуальных агентов остаются в силе и могут быть применены как неотъемлемая часть формируемой средствами КДЯ структуры агента.

Результаты исследований, изложенные в данной статье, получены при финансовой поддержке Минобрнауки РФ в рамках реализации проекта «Разработка и создание высокотехнологичного производства инновационной системы комплексного учета, регистрации и анализа потребления энергоресурсов и воды промышленными предприятиями и объектами ЖКХ» по постановлению правительства №218 от 09.04.2010г. Исследования проводились в ФГАОУ ВО ЮФУ.

Литература

1. Рассел, Стюарт, Норвиг, Питер. Искусственный интеллект: современный подход, 2-е изд.: Пер. с англ. — М.: Издательский дом "Вильямс", 2006. — 1408 с.: ил. — Парал. тит. англ.

2. Подсвиров В.Н., Семинистая Е.С., Подопригора В.Б. Аналитическое программное обеспечения прогнозирования потребления ресурсов в системе комплексного учета, регистрации и анализа потребления энергоресурсов и воды промышленными предприятиями и объектами ЖКХ // Инженерный вестник Дона, 2017, №4 URL: ivdon.ru/ru/magazine/archive/n4y2017/4604.

3. Корецкий А.А., Подопригора В.Б., Мирошниченко Е.П. Особенности разработки и внедрения системы учета энергоресурсов // Инженерный вестник Дона, 2017, №3 URL: ivdon.ru/ru/magazine/archive/N3y2017/4365.

4. Семенистая Е.С., Анацкий А.Г., Бойко Ю.А. Разработка программного обеспечения автоматизированной системы контроля и учета энергоресурсов и воды // Инженерный вестник Дона, 2016, №4 URL: ivdon.ru/ru/magazine/archive/n4y2016/3897.

5. Podsvirov V.N. Intelligent Learning Agents Construction by Context-Redefined Language Technology (Resource Consumption Behavior Prediction Tasks). International Journal of Applied Engineerining Research ISSN 0973-4562 Volume 12, Number 16 (2017) pp. 5984-5989.

6. Podsvirov V.N. Context-Redefined Language Application for the Tasks of Intelligent Learning Agents (Resource Consumption Behavior Prediction Tasks). International Journal of Applied Engineering Research ISSN 0973-4562 Volume 11, Number 15 (2016) pp. 8471-8484.

7. Подсвиров В.Н. Использование контекстно-доопределяемых языков для решения задачи обучения интеллектуальных агентов. Актуальные проблемы современной науки: Международная научно-практическая конференция. Алушта, 2016.-С.229-232.

8. Подсвиров В.Н. Контекстно-доопределяемые языки и интеллектуальные агенты. Актуальные проблемы современной науки: Международная научно-практическая конференция. Алушта, 2015.-С.192-194.

9. Подсвиров В.Н. Контекстно-доопределяемые языки и продукции. Инфокоммуникационные технологии в науке, производстве и образовании: Шестая международная научно-техническая конференция. Ставрополь, 2014.- С. 391-393.

10. Подсвиров В.Н. Продукции и контекстно-доопределяемые языки. Актуальные проблемы современной науки: Вторая международная научно-практическая конференция. Ставрополь, 2013.- С. 190-192.

11. Подсвиров В.Н. Один подход к реализации контекстно-доопределяемых языков. Инфокоммуникационные технологии в науке, производстве и образовании: Пятая международная научно-техническая конференция. Ставрополь, 2012. – С. 54 – 57.

References

1. Rassel, Styuart, Norvig, Piter. Iskusstvennyy intellekt: sovremennyy podkhod [Artificial Intelligence: a modern approach]. 2 izd.: Per. s angl. M.: Izdatel'skiy dom "Vil'yamc", 2006. 1408 p.

2. Podsvirov V.N., Semenistaya Ye.S., Podoprigora V.B. Inženernyj vestnik Dona (Rus), 2017, №4. URL: ivdon.ru/ru/magazine/archive/n4y2017/4604.

3. Koretskiy A.A., Podoprigora V.B., Miroshnichenko Ye.P. Inženernyj vestnik Dona (Rus), 2017, №3. URL: ivdon.ru/ru/magazine/archive/N3y2017/4365.

4. Ye.S.Semenistaya, I.G. Anatskiy, YU.A. Boyko. Inženernyj vestnik Dona (Rus), 2016, №4 URL: ivdon.ru/ru/magazine/archive/n4y2016/3897.

5. Podsvirov V.N. Intelligent Learning Agents Construction by Context-Redefined Language Technology (Resource Consumption Behavior Prediction Tasks). International Journal of Applied Engineerining Research ISSN 0973-4562 Volume 12, Number 16 (2017) pp. 5984-5989.

6. Podsvirov V.N. Context-Redefined Language Application for the Tasks of Intelligent Learning Agents (Resource Consumption Behavior Prediction Tasks). International Journal of Applied Engineering Research ISSN 0973-4562 Volume 11, Number 15 (2016) pp. 8471-8484.

7. Podsvirov V.N. Ispol'zovaniye kontekstno-doopredelyayemykh yazykov dlya resheniya zadachi obucheniya intellektual'nykh agentov [The use of context-defined languages for solving the problem of training intelligent agents]. Ak-

tual'nyye problemy sovremennoy nauki: Mezhdunarodnaya nauchno-prakticheskaya konferentsiya. Alushta, 2016, pp. 229-232.

8. Podsvirov V.N. Kontekstno-doopredelyayemyye yazyki i intellektual'nye agenty [Context-definable languages and intellectual agents]. Aktual'nyye problemy sovremennoy nauki: Mezhdunarodnaya nauchno-prakticheskaya konferentsiya. Alushta, 2015, pp. 192-194.

9. Podsvirov V.N. Kontekstno-doopredelyayemyye yazyki i produktsii [Context-redefined languages and productions]. Infokommunikatsionnye tekhnologii v nauke, proizvodstve i obrazovanii: Shestaya mezhdunarodnaya nauchno-tekhnicheskaya konferentsiya. Stavropol', 2014, pp. 391-393.

10. Podsvirov V.N. Produktsii i kontekstno-doopredelyayemyye yazyki [Productions and context-redefined languages]. Aktual'nyye problemy sovremennoy nauki: Vtoraya mezhdunarodnaya nauchno-prakticheskaya konferentsiya. Stavropol', 2013, pp. 190-192.

11. Podsvirov V.N. Odin podkhod k realizatsii kontekstno-doopredelyayemykh yazykov [Context-redefined languages implementation, one approach]. Infokommunikatsionnye tekhnologii v nauke, proizvodstve i obrazovanii: Pyataya mezhdunarodnaya nauchno-tekhnicheskaya konferentsiya. Stavropol', 2012, pp. 54 – 57.