


Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

**ГРАФОВАЯ ИНТЕРПРЕТАЦИЯ ПРОГРАММНОГО КОДА
КАК СПОСОБ ОТРАЖЕНИЯ КОНТЕКСТА ЕГО ВЫВОДА**
АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

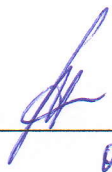
студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Сагунова Данила Георгиевича

Научный руководитель
доцент, к. ф.-м. н.


_____ 05.06.17

А. С. Иванов

Заведующий кафедрой
к. ф.-м. н.


_____ 05.06.17

С. В. Миронов

Саратов 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Основное содержание работы	4
1.1 Постановка проблемы	4
1.2 Графовая интерпретация программного кода	5
1.3 Подробное описание реализации решения	7
1.4 Тестирование системы	10
ЗАКЛЮЧЕНИЕ	12
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	13

ВВЕДЕНИЕ

Рассматриваемая в работе проблема может быть описана на примере некоторой части исходного кода, результатом выполнения которого будут являться, в частности, определенные выходные данные. Эти данные подчинены некоторым правилам, которые поручены только определенным участникам группы разработчиков, которым, с другой стороны, не известны правила разработки непосредственно исходного кода. Без специального решения таким участникам понадобится разбираться в правилах разработки кода или же отслеживать поведение программы уже в процессе ее работы, хотя им необходим лишь определенный контекст вывода программы. Одной из задач данной работы является более подробное описание проблемы, а также описание ее решения.

Решением проблемы предложена система автоматического построения графовой интерпретации программного кода. В данной работе поставленная проблема и решение рассматриваются на примере проекта видеоигры «Evol Online» в жанре ролевой игры с открытым миром. Там описанная проблема возникла в процессе постепенной локализации диалогов персонажей, чьи реплики и являются необходимым контекстом вывода. Таким образом, одной из целей работы поставлена реализация системы автоматического построения графовой интерпретации программ диалогов персонажей с возможностью мгновенного перевода. Такая система призвана быть удобным инструментом переводчика, не имеющего отношения непосредственно к программам поведения персонажей.

Итак, в рамках выпускной квалификационной работы поставлены следующие задачи:

- сформулировать проблему перевода диалогов персонажей;
- предложить и описать решение этой проблемы;
- дать подробное описание реализации предложенного решения;
- протестировать реализованное решение как систему перевода диалогов.

1 Основное содержание работы

1.1 Постановка проблемы

Данный раздел посвящен истории возникновения проблемы и ее описанию в рамках проекта «Evol Online». Тем не менее, представляется возможной и более общая формулировка проблемы.

«Evol Online» представляет собой проект игрового MMORPG сервера. Основные усилия в разработке игрового сервера сводятся к наполнению его внутриигровым содержимым, в том числе графикой, музыкой, планами локаций, а так же так называемыми неигровыми персонажами (NPC). Согласно [1], NPC управляются программой сервера и служат важным средством создания игровой атмосферы, мотивируют игроков совершать те или иные действия и являются основным источником информации об игровом мире и сюжете игры.

Таким образом, NPC являются неотъемлемой частью игрового процесса, а взаимодействие игрока с ними осуществляется путем диалогов. Вообще говоря, реплики самого игрока в процессе диалога ограничиваются несколькими фиксированными программой персонажа вариантами, поэтому окно диалога выглядит как текстовое поле, под которым во время диалога у игрока появляется возможность выбрать один из вариантов ответа.

В «Evol Online» программы NPC создаются с помощью исходного кода на специальном Си-подобном языке. Такие программы содержат вызовы функций, реализующих взаимодействие игрока с персонажем — вывод реплик в диалоговое окно и предоставление выбора ответов игроку. Связи между такими вызовами осуществляются с помощью операторов ветвления, циклов, меток, локальных функций.

Одной из особенностей проекта «Evol Online» является мультиязычность. Другими словами, в проекте реализована возможность перевода внутриигрового содержимого игры на разные языки, причем игроку предоставляется возможность изменить язык окружения прямо во время игры. В игре эта возможность реализована с помощью специальных опций диалогов NPC.

На данный момент процесс перевода диалогов происходит непосредственно через систему Transifex. Реплики персонажей там становятся вырванными из контекста и следуют в произвольном порядке, а реплики различных персонажей оказываются перемешаны, и, вообще-то говоря, могут совпадать,

но иметь разное значение.

Переводчик сталкивается с проблемой интерпретации конкретной реплики, и ему приходится обращаться к исходному коду программы поведения персонажей, возможно, связываясь при этом с ее разработчиком. В этом случае процесс оказывается достаточно трудоемким сразу для нескольких участников проекта.

Данная работа возникла в процессе реализации решения проблемы процесса такого перевода. Дальнейшие разделы посвящены этому конкретному решению — графовым интерпретациям — и его реализации. Помимо этого способа, в работе поверхностно рассмотрены некоторые альтернативные решения.

В работе предлагается некоторое обобщение описанной выше проблемы.

Итак, пусть существует программа, описанная некоторым кодом. На этапе исполнения программа совершает некоторые действия, среди которых есть некие *важные* действия. Будем считать, что эти важные действия приводят к некоторому результату, который будем называть *выводом* программы. Вывод производится только важными действиями, и, таким образом, зависит от того, какие важные действия и в какой последовательности осуществляются программой.

Контекстом вывода можно назвать информацию о всевозможных последовательностях важных действий программы во время ее исполнения. Теперь проблему можно задать так: построить отражение контекста вывода программы, для которой известны важные действия, на основании ее исходного кода.

В проблеме перевода диалогов персонажей программами являются программы персонажей, важными действиями — отображение реплик (нейгрового персонажа или игрока), выводом — последовательность этих реплик. В качестве способа отражения контекста вывода выбраны графовые представления, описанные в следующем разделе.

1.2 Графовая интерпретация программного кода

Данный раздел представляет собой описание теоретической идеи, положенной в основу решения. Основным объектом рассмотрения будет являться ориентированный граф, служащий удобной контекстной моделью представления.

В работе дано формальное определение графовой интерпретации программного кода. *Графовой интерпретацией программы*, для кода которой фиксирован контекст вывода, называется ориентированный граф $G = (V, E)$ такой, что:

1. Каждой команде вывода c в программе соответствует единственная вершина $v_c \in V$;
2. Любая вершина $v \in V$ графа соответствует не более, чем одной команде вывода;
3. Если существует такая последовательность выполнения команд программы, что команда вывода b следует *непосредственно* после команды вывода a , то в графе существует цепь из вершины v_a в вершину v_b , содержащая в качестве промежуточных *только* некоторые вспомогательные вершины u_1, u_2, \dots, u_k : формально, согласно [2], $(v_a, u_1), (u_i, u_{i+1}), (u_k, v_b) \in E$;
4. Существует единственная вершина, имеющая смысл точки входа в программу, $v_s \in V$. Следует считать, что эта вершина соответствует самой первой, но несуществующей, команде вывода, для которой выполняется правило 3.

Правила 1 и 2 являются естественными, однако не исключают существования и других вершин, что видно на примере правила 4.

Правило 4, в свою очередь, задает «стартовую» вершину программы, не имеющую, однако, действительного отражения в выводе: из этой вершины дуги проведены в те команды вывода, которые могут идти первыми при некотором исполнении программы, то есть пользователь может увидеть результат вывода этих команд до выполнения каких-либо других команд вывода.

Любой маршрут графа, начинающийся в вершине точки входа, согласно правилу 3 призван отражать одну из возможных последовательностей выполнения команд вывода. [2]

Правило 3 в работе изначально было сформулировано достаточно жестко (без вспомогательных вершин). Было легко привести пример, когда даже минимальная графическая интерпретация будет содержать $O(n^2)$ дуг (то есть, согласно [3], будет плотным графом), где n соответствует количеству команд вывода в программе. Пример алгоритма такой программы представляет ал-

горитм 1.

Алгоритм 1: Пример алгоритма, соответствующего программе с большим количеством дуг в графовой интерпретации

$color \leftarrow 0;$

до тех пор, пока $color \neq 7$ выполнять

$color \leftarrow$ случайное число от 1 до 7;

 вывести цвет радуги под номером $color$;

конец

Изображения графовой интерпретации программы, описанной алгоритмом 1, представлены на рисунке 1. В случае отсутствия вспомогательных вершин существует возможность вывода любого из шести цветов после любого другого из них, и вершин минимальная графовая интерпретация столь простой программы содержит, согласно [4], в качестве подграфа клику K_6 — полный граф из 6 вершин.

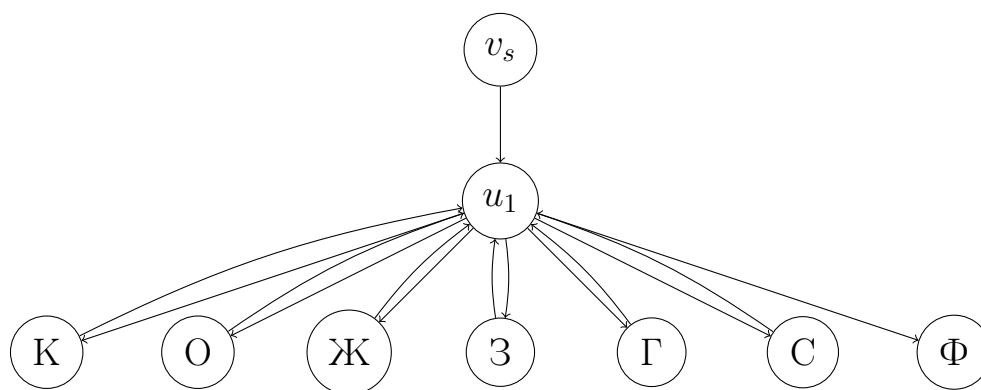


Рисунок 1 – Графовая интерпретация программы, представленной алгоритмом 1, использующая одну вспомогательную вершину

В работе приведены некоторые преимущества и недостатки графовых интерпретаций, сформулированные согласно [5] и [6].

1.3 Подробное описание реализации решения

Данный раздел представляет практическую часть работы. В частности, здесь описано большинство идей, необходимых для построения графовых интерпретаций на примере системы перевода диалогов персонажей проекта «Evol Online».

Данный подраздел описывает программные продукты, используемые в разработке автоматической системы построения графовых интерпретаций. Это инструменты:

- GNU flex [7];
- yacc [8];
- GCC и make [9, 10];
- Cytoscape.js [11];
- Transifex API [12];
- jQuery [13].

Прежде всего, процесс построения графовой интерпретации программы поведения персонажей сводится к лексическому и синтаксическому анализу ее исходного кода. Соответствующие анализаторы реализуются с помощью генераторов flex и yacc на основании исходных файлов в соответствующем генераторам формате. Любая корректно описанная на языке программ персонажей программа должна успешно проходить оба этапа анализа, в результате которого должен быть выведен основной символ.

Сам же процесс построения графовой интерпретации программы имеет уже семантический характер, который, благодаря возможности yacc реализуется с помощью описания семантических зависимостей. Другими словами, создание вершин и дуг графа происходит непосредственно во время вывода семантических правил, которые выводятся «слева-направо» с помощью стека, соответствуя естественной последовательности следования команд.

После завершения синтаксического анализа граф нуждается в постобработке, в том числе исключая некоторые вспомогательные вершины и дуги, а также в выводе в формате, необходимом для построения интерфейсной части.

Интерфейсная часть представляет собой веб-приложение, реализованное с помощью JavaScript. Его основной целью является интеграция системы перевода Transifex вместе с графовым представлением, которая позволяет переводчику работать непосредственно в контексте вывода без приложения дополнительных усилий.

В работе приводится альтернативный способ построения графовых интерпретаций, однако он сочтен нецелесообразным в связи со слишком формальным подходом к построению.

Прежде всего, для реализации семантических связей, задающих построение графа, необходимо было определить тип семантических значений. Для определения такого типа сначала потребовалось введение некоторого атомар-

ного типа.

Таким типом было решено считать пару целых чисел. Смысл значений этой пары заключается в следующем: если атом соответствует константе или имени идентификатора, то первое ее значение принимается равным -1 , а второе — равным идентификатору константы в специальном массиве. Все константы исходно полагаются строковыми.

В противном случае, оба значения пары соответствуют номерам вершин графа. Первое из них соответствует «стартовой» вершине подграфа, второе — «финишной». Ввиду специфики построения графовых представлений, основным типом семантических значений было решено выбрать динамический массив переменных атомарного типа.

Для обработки семантических переменных на языке C++ был реализован функционал, упрощающий описание семантических связей. Вот некоторые возможности этого функционала:

- создание атомов-констант и вершинных атомов;
- создание дуг между вершинными атомами;
- проверка и извлечение строковых и числовых констант из атомов;
- установление флаговых значений вершинам, в том числе с помощью обходов графа [3];
- обработка вершин-меток безусловного перехода;
- обработка вершин, соответствующих операторам преждевременного выхода;
- обработка вариантов выбора ответных реплик.

В процессе выполнения работы были сформулированы и описаны многие семантические связи, необходимые для построения графов параллельно синтаксическому разбору. Это семантические связи:

- последовательностей команд;
- параметров функций;
- условных операторов;
- операторов цикла;
- операторов безусловного перехода;
- меню выбора ответных реплик;
- вызовов локальных функций.

Все они подробно описаны в работе.

На этапе постобработки удаляются вспомогательные вершины. Удаление вершины реализовано образом, сохраняющим отношение следования, то есть информацию о дугах, входящих и исходящих из этой вершины. Для этого было реализовано динамическое удаление и добавление ребер в граф. Вывод графа был реализован в формате, соответствующем описанию констант массивов языка JavaScript.

Для интеграции приложения с системой перевода Transifex были использованы Transifex API (ознакомление с ним происходило с помощью [14]) и jQuery. Было реализовано сопоставление вершинам графа идентификаторов строк в системе Transifex. В том числе, реализовано обновление перевода для каждой вершины, происходящее после успешной загрузки строк. Вершины, имеющие перевод, помечаются специальным идентификатором и отображаются по другому.

Далее были описаны два поля ввода, содержащие оригинал строки и ее перевод на языке переводчика. Первое поле неизменяемо, а второе может редактироваться переводчиком. Для них был реализован функционал, осуществляющий запросы к API для обновления данных о переводе единственной строки. Структура подобных запросов была изучена на сайте документации Transifex [15]. Для выполнения потребовался функционал, позволяющий вычислять контрольные суммы на языке JavaScript [16].

1.4 Тестирование системы

В ходе тестирования реализованного решения были успешно проверены алгоритмы построения графов на целевых данных — программах поведения персонажей проекта.

Были рассмотрены реализованные возможности интерфейса, в частности, загрузки действующего перевода. Для этого на сайте Transifex был получен специальный ключ API.

Кроме того, для тестирования обновления перевода строки была выбрана вершина, строка которой совпадает со строкой другой вершины. Для строки этой вершины перевод в базе данных отсутствовал. Процесс перевода строки для вершины на веб-странице и внешний вид веб-страницы после обновления перевода этой строки представлены на рисунке 2. На рисунке показано, что перевод обновился сразу и для второй вершины.

Результаты перевода через разработанное веб-приложение были успеш-

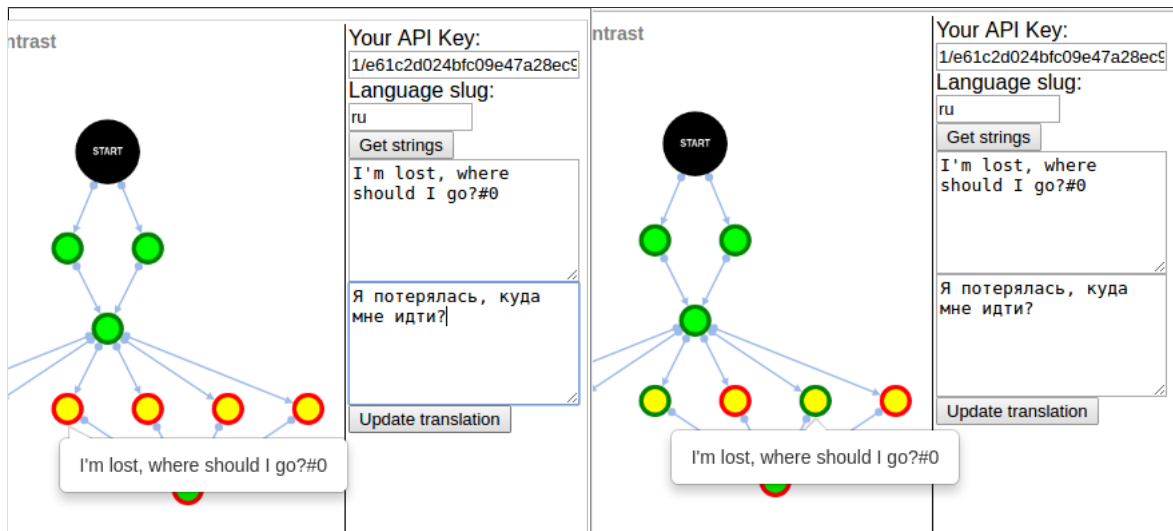


Рисунок 2 – Внешний вид веб-страницы во время (слева) и после (справа) перевода строки

но проверены на сайте Transifex.

ЗАКЛЮЧЕНИЕ

В рамках выполнения выпускной квалификационной работы была сформулирована проблема перевода диалогов в программах поведения персонажей. Приведены альтернативные подходы к ее решению и ее обобщение на более широкий класс задач. Было предложено основное решение проблемы — система автоматического построения графовых интерпретаций.

Графовым интерпретациям было дано подробное формальное описание, приведены возможные допущения из нескольких правил формального определения. Были приведены преимущества и недостатки графовых интерпретаций.

Была реализована система автоматического построения графовых интерпретаций для решения проблемы перевода диалогов персонажей.

Были перечислены и описаны инструменты, используемые для реализации системы автоматического построения графовых интерпретаций для решения проблемы перевода диалогов персонажей. Процессу реализации, как и основным идеям, используемым в этом процессе, так же было дано подробное описание.

Функции реализованной системы — как построение графовых интерпретаций, так и интеграция с системой перевода — были протестированы на целевых данных. Было замечено, что графовые интерпретации являются достаточно хорошим способом отражения контекстов диалогов персонажей, и реализованная система решает поставленную задачу.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 *Ellison, B.* Defining Dialogue Systems [Электронный ресурс] / B. Ellison. — URL: http://www.gamasutra.com/view/feature/3719/defining_dialogue_systems.php (Дата обращения 28.05.2017). Загл. с экр. Яз. англ.
- 2 *Оре, О.* Теория графов / О. Оре. — Москва: Либроком, 2009.
- 3 *Кормен, Т.* Алгоритмы. Построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. — Москва: Вильямс, 2016.
- 4 *Харари, Ф.* Теория графов / Ф. Харари. — Москва: Либроком, 2009.
- 5 *Емеличев, В.* Лекции по теории графов / В. Емеличев, О. Мельников, В. Сарванов, Р. Тышкевич. — Москва: «Наука», Глав. ред. физико-математической лит-ры, 1990.
- 6 *Евстигнеев, В.* Графы в программировании: обработка, визуализация и применение / В. Евстигнеев, В. Касьянов. — Санкт-Петербург: БХВ-Петербург, 2003.
- 7 *Levine, J.* flex & bison: Text Processing Tools / J. Levine. — O'Reilly Media, 2009.
- 8 *Brown, D.* lex & yacc / D. Brown, J. Levine, T. Mason. — O'Reilly Media, 1992.
- 9 GCC, the GNU Compiler Collection [Электронный ресурс]. — URL: <https://gcc.gnu.org/> (Дата обращения 28.05.2017). Загл. с экр. Яз. англ.
- 10 GNU Make Manual [Электронный ресурс]. — URL: <http://www.gnu.org/software/make/manual/> (Дата обращения 28.05.2017). Загл. с экр. Яз. англ.
- 11 *Franz, M.* Cytoscape.js: a graph theory library for visualisation and analysis / M. Franz, C. T. Lopes, G. Huck, Y. Dong, O. Sumer, G. D. Bader // *Bioinformatics*. — 2015. — P. btv557.
- 12 Introduction to the Transifex API — Transifex Documentation [Электронный ресурс]. — URL: <https://docs.transifex.com/api/introduction> (Дата обращения 28.05.2017). Загл. с экр. Яз. англ.

- 13 *Freeman, A. Pro jQuery 2.0 (Expert's Voice in Web Development) / A. Freeman. — Apress, 2013.*
- 14 Translations — Transifex Documentation [Электронный ресурс]. — URL: <https://docs.transifex.com/api/translations> (Дата обращения 28.05.2017). Загл. с экр. Яз. англ.
- 15 Translation Strings — Transifex Documentation [Электронный ресурс]. — URL: <https://docs.transifex.com/api/translation-strings> (Дата обращения 28.05.2017). Загл. с экр. Яз. англ.
- 16 Raj's Home: Cryptography: JavaScript MD5 [Электронный ресурс]. — URL: <http://rajhome.org.uk/сrypt/md5/> (Дата обращения 28.05.2017). Загл. с экр. Яз. англ.

Сделано
05.06.2017