УДК 004.41

Н.А. Ошнуров, А.Н. Пустыгин, А.А. Ковалевский

Построение универсального промежуточного представления исходных текстов на языках C++ и C#

Описываются существующие промежуточные представления исходных текстов. Описываются инструменты для получения абстрактного синтаксического дерева языков C++ и C#. Описывается новое представление, содержащее семантическую информацию, описано его преобразование к формату, доступному для обработки существующими инструментами.
Ключевые слова: промежуточное представление, исходный код, статический анализ, C++, C#.

Задача статического анализа кода – задача анализа программного обеспечения, производимого без реального выполнения исследуемых программ. Для ее решения нужен набор программных инструментов для каждого типа анализа. В случае анализа гетерогенных исходных текстов, т.е. текстов на нескольких языках программирования, задача становится сложнее. Это связано с тем, что создание таких инструментов – сама по себе трудоёмкая задача. Развитие данных инструментов несет идею создания единого основания для таких анализов. В предыдущих работах [1] предлагалось решение этой проблемы путём введения единого универсального промежуточного представления (УПП) исходных текстов на различных ЯП и его последующего автоматизированного анализа путем построения эквивалентных представлений. Целью данной работы является создание промежуточного представления для исходных текстов на языках C++ и C#, содержащего в себе всю информацию об исходном тексте программы и предоставляющего возможность легкого создания автоматизированных инструментов для анализов на его основе.

Проектирование универсального промежуточного представления. Формат промежуточного представления должен отвечать некоторому набору критериев, среди них можно выделить:

1. Простоту реализации инструментов для автоматической обработки.
2. Расширяемость – простоту добавления новых языковых конструкций.
3. Читабельность – удобство восприятия человеком.

Среди распространённых форматов представления структурированных данных этому набору критериев удовлетворяют JSON [2], YAML [3] и XML [4].

Стек технологий XML предоставляет инструменты, которые можно использовать для решения задач статического анализа исходных текстов программ, путём обработки их промежуточного представления. Из них можно выделить следующие:

1. XPath [5] – язык запросов для выборки и навигации по узлам XML-документа, вычисления некоторых его метрик.
2. XQuery [6] – функциональный язык запросов, разработанный для выполнения запросов и трансформации коллекций структурированных и неструктурированных данных.
3. XSLT [7] – язык для трансформации XML-документов в другие XML-документы, объекты HTML или XSL, которые затем могут быть преобразованы в PDF, PostScript, SVG или PNG. То есть позволяет получить, например, визуальное эквивалентное представление, пригодное для анализа.
4. XML-базы данных, например, Sedna [8] (лицензия Apache License 2.0) или BaseX (лицензия BSD) [9], которые можно использовать в качестве единого хранилища представлений одного или нескольких проектов.

Таким образом, выбор формата XML, как основы для промежуточного представления позволяет сократить затраты на создание инструментов анализа и построения эквивалентных представлений.

Использование JSON и YAML было признано нецелесообразным из-за отсутствия для данных форматов подобного инструментария.

Для получения промежуточного представления исходного кода предполагается использовать абстрактное синтаксическое дерево (AST) [10], дополненное семантической информацией. Среди существующих форматов представления, которые можно использовать в качестве базы для построения анализаторов и эквивалентных представлений, можно выделить следующие:

1. SrcML (Source Markup Language) [11] – открытое представление исходных текстов, основанное на комбинации самого текста и выборочной информации из AST, заключенное в одном XML-документе. Для следующих языков существуют соответствующие подклассы данного представления: C, C++ (CppML), Java (JavaML), C# (SrcML.Net). Формат основан на добавлении информации о структуре исходных текстов к самим исходным текстам. В силу того, что SrcML сохраняет весь исходный код, он хорошо применим как для различных трансформаций исходных текстов, так и для извлечения фактов из них. Однако он не содержит большого количества высокоуровневой информации, что может вызывать затруднения при создании инструментов анализа на его основе. Так, например, в листинге 1 показан фрагмент кода обмена значений численных переменных, в листинге 2 представлено его srcML-представление, по которому невозможно точно установить ни место объявления переменных *a*, *b* и *t*, ни их тип.

```
// swap two numbers
if(a>b)
{
     t = a;
     a = b;
     b = t;
}
```

Листинг 1. Фрагмент кода обмена значений численных переменных C++

```
<unit>
      <comment type=”line”>// swap two numbers</comment>
      <if>if<condition>(<expr><name>a</name>&gt;<name>b</name></expr>)</con
dition><then>
      <block>{

     <expr_stmt><expr><name>t</name>=<name>a</name></expr>;</expt_stmt>
     <expr_stmt><expr><name>a</name>=<name>b</name></expr>;</expt_stmt>
     <expr_stmt><expr><name>b</name>=<name>t</name></expr>;</expt_stmt>
}</block></then></if>
</unit>
```

Листинг 2. SrcML представление исходного кода обмена значений численных переменных

2. JavaML [12] – формат представления исходных текстов на Java. Представляет собой AST Java кода, кроме того, JavaML хранит в себе некоторую семантическую информацию. Например, IDREF-теги используются для того, чтобы ссылаться на объявление переменной. Недостатком JavaML в контексте задачи построения эквивалентных преставлений и в дальнейшем различных анализов является сильное абстрагирование от исходного текста. Так, например, все виды циклов отображаются в один элемент представления <loop>, а комментарии объявлений переменных являются незначимыми и не находят отражения в представлении. Несмотря на заявления авторов, что данный формат может быть использован для других объектно-ориентированных языков, он неприменим, например, для C++, поскольку не имеет механизма описания указателей. Следует отметить, что достоинства JavaML применены в предлагаемом формате промежуточного представления.

3. Представление OOML [13] является обобщенным представлением объектно-ориентированных языков и основывается на общности моделей объектно-ориентированных языков. Изначально он являлся наследником JavaML и унаследовал все недостатки последнего.

4. Специфические для языков представления, такие как GCC-XML [14] (основанное на внутреннем представлении компилятора GCC), cpp2xml [15] не обладают свойством универсаль-ности и служат для выполнения узких задач, например, трансляции кода в машинный код или оптимизации машинного кода.

Главным требованием к формату промежуточного представления является включение в состав атрибутов узлов такого набора тегов, который позволяет удовлетворить требованиям спецификации языков при разборе всех синтаксических конструкций, описанных в стандартах.

В связи с отсутствием альтернативных решений для получения промежуточных представлений, пригодных для поставленных целей, был произведен поиск инструментов для получения AST (табл. 1).

Таблица 1

Обзор открытых инструментов для получения AST

Инструмент	Язык	Возможности	Недостатки
ROSE Compiler [16]	C++	Поддержка C, C++11	Поддержка только C++
Eclipse CDT [17]	C++	IDE для статического анализа. Поддержка C++	Отсутствие поддержки C++11
Mozilla Dehydra & Treehydra (GCC GIMPLE) [18]	C++	Извлечение информации о некоторых узлах AST	Разработка приостановлена в 2010, неполное и неэквивалентное исходному тексту AST
Mono C# Compiler [19]	C#	Кроссплатформенность решения (Windows, GNU/Linux, MacOS, iOS, Android)	Несовместимость API разных версий компилятора
ICSharpCode.NRefactory	C#	Кроссплатформенность решения, активное сообщество	Отсутствие поддержки нового стандарта языка C# 6.0
Roslyn	C#	Поддержка нового стандарта языка C# 6.0, инструментарий для анализа исходного кода	Относительно новое решение, возможны несовместимые изменения API

Источниками для создания формата промежуточного представления послужили следующие документы:

1. Спецификация языка C# [20].

2. Проект стандарта C++ [21].

Для разработки формата промежуточного представления по C++ и C# были выбраны следующие инструменты, удовлетворяющие заданным критериям (полнота, актуальность инструмента, затраты на разработку промежуточного представления):

1. Clang и его библиотека libclang (для языков C/C++), поддержка C++11, Objective-C, активное сообщество, поддержка крупных компаний (Apple, Google), наличие SDK и отдельной библиотеки для получения AST. Библиотека libclang – часть инструментария clang, которая используется для статического анализа и автозавершения кода. Используется в IDE Xcode (Apple).

2. Roslyn [22] (для языка C#) – открытый набор компиляторов и API для анализа исходных кодов для языков C# и Visual Basic .Net, разрабатываемый компанией Microsoft.

Формальное описание представления. В предлагаемом представлении AST исходного текста отображается в XML файл по разработанному шаблону [23]. Так, например, оператору условного перехода if сопоставляется узел XML – <If /> (табл. 2). Поддерживается ссылочная целостность в том смысле, что вне зависимости от того, в каком месте какого файла с исходными текстами объявлен элемент с неким идентификатором ID, данный идентификатор используется при любом использовании данного элемента. Кроме того, представление содержит в себе всю информацию о типах, например, о типе объявляемой переменной, типе возвращаемого значения для метода.

Таблица 2

Пример УПП для C++ и C#

Исходный текст	УПП

`if (a == 10);`	`<If line="5" col="3">`
	`<Condition line="5" col="3">`
	`<op:Binary type="==" line="5" col="6">`
	`<Left>`
	`<VarRef name="a" ref="1" line="5" col="6"/>`
	`</Left>`
	`<Right>`
	`<lit:Int value="10" text="10" line="5" col="11"/>`
	`</Right>`
	`</op:Binary>`
	`</Condition>`
	`<Then line="5" col="3"/>`
	`</If>`

Проектирование и кодирование генераторов этого представления из текстов C++ и C#. При построении УПП исходный текст анализируемого проекта проходит стадии синтаксического и

семантического анализа. Это позволяет выводить всю доступную информацию об исходном коде в УПП. Таким образом, впоследствии из УПП можно целиком восстановить исходный текст с точностью до форматирования. В обоих генераторах для обхода AST используется паттерн «Посетитель». В случае Roslyn используется наследник класса CSharpSyntaxWalker для C#. Библиотека libclang для построения AST исходного текста на C++ предоставляет набор функций для обхода дерева и извлечения из узлов полезной информации.

Выводы. Предложенное представление позволяет производить большой спектр высокоуровневых анализов по обнаружению ошибок в исходных текстах, улучшению их читаемости, производительности, а также выявлению структуры и функциональности программ, документация по которым недоступна или может не соответствовать действительности. Данное свойство достигается за счет использования большого количества семантической информации об исходном коде проектов. Предложенный формат является расширяемым и при необходимости может быть легко доработан для поддержки большего количества языков и их конструкций. Решение использовать с качестве основы формат XML позволяет для выполнения анализов работать со стандартными открытыми технологиями, например, такими как XSLT, XQuery, XML Schema. Все это в совокупности позволяет сказать, что данное представление может быть эффективно использовано для выполнения задач статического анализа исходных текстов.

Направления дальнейшего движения. В качестве дальнейшего развития данного универсального представления и инструментов для его получения можно выделить решение следующей задачи – доработка существующих инструментов до состояния, при котором синтаксические конструкции языков C++ и C# обрабатываются полностью, включая новейшие стандарты и спецификации, несмотря на то, что на данный момент поддерживается большая часть синтаксиса C++ и C#.

Свойства предложенного представления открывают огромные возможности по совместному анализу различных эквивалентных представлений, которые можно получить из него. В перспективе это позволяет извлекать нетривиальные сведения о программах.

Особый интерес представляет собой задача интеграции данных инструментов и представления с существующими решениями для работы с исходными текстами, а также создание набора анализов, основывающихся на данном представлении.

Литература

1. Зубов М.В. Статический анализ ПО с помощью его промежуточных представлений и технологий с открытым исходным кодом / М.В. Зубов, А.Н. Пустыгин, Е.В. Старцев // Матер. 2-й Междунар. конф. «FOSS. Lviv–2012», Львов. – Львів: Сорока, 2012. – С. 165–168.

2. JSON (JavaScript Object Notation) [Электронный ресурс]. – Режим доступа: http://www.json.org, свободный (дата обращения: 13.01.2014).

3. YAML: YAML Ain't Markup Language [Электронный ресурс]. – Режим доступа: http://www.yaml.org, свободный (дата обращения: 13.01.2014).

4. Extensible Markup Language (XML) [Электронный ресурс]. – Режим доступа: http://www.w3.org/XML/, свободный (дата обращения: 13.01.2014).

5. XML Path Language (XPath). W3C Recommendation, 16 November 1999 [Электронный ресурс]. – Режим доступа: http://www.w3.org/TR/xpath/, свободный (дата обращения: 13.01.2014).

6. XQuery 1.0: An XML Query Language (Second Edition). W3C Recommendation, 14 December 2010 [Электронный ресурс]. – Режим доступа: http://www.w3.org/TR/xquery/, свободный (дата обращения: 13.01.2014).

7. XSL Transformations (XSLT). W3C Recommendation, 16 November 1999 [Электронный ресурс]. – Режим доступа: http://www.w3.org/TR/xslt, свободный (дата обращения: 13.01.2014).

8. Fomichev A. Sedna: A Native XML DBMS / M. Grinev, S. Kuznetsov // SOFSEM 2006: Theory and Practice of Computer Science. Lecture Notes in Computer Science. – 2006. – Vol. 3831. – P. 272–281.

9. BaseX. The XML Database [Электронный ресурс]. – Режим доступа: http://www.basex.org/, свободный (дата обращения: 13.01.2014).

10. Компиляторы: принципы, технологии и инструментарий / А. Ахо, М. Лам, Р. Сети, Д. Ульман. – М.: Вильямс, 2010. – 1184 с.

11. Collard M.L. An XML-Based Lightweight C++ Fact Extractor / M.L. Collard, H.H. Kagdi, J.I. Maletic // Program Comprehension, 2003. 11th IEEE International Workshop. – May 10–11, 2003. – P. 134–143.

12. Badros G.J. JavaML: A Markup Language for Java Source Code // The International Journal of Computer and Telecommunications Networking. – 2000. – Vol. 33. – P. 159–177.

13. Mamas E. Towards Portable Source Code Representations Using XML / E. Mamas, K. Kontogiannis // Proceedings of the Seventh Working Conference on Reverse Engineering. – 2000. – P. 172.

14. GCC-XML [Электронный ресурс]. – Режим доступа: https://github.com/gccxml/gccxml, свободный (дата обращения: 13.01.2014).

15. The C++2XML page [Электронный ресурс]. – Режим доступа: http://scl.csd.uwo.ca/Projects/cpp2xml/index.html, свободный (дата обращения: 13.01.2014).

16. ROSE compiler infrastructure [Электронный ресурс]. – Режим доступа: http://www.rosecompiler.org/, свободный (дата обращения: 13.01.2014).

17. Eclipse CDT (C/C++ Development Tooling) ROSE compiler infrastructure [Электронный ресурс]. – Режим доступа: http://www.eclipse.org/cdt, свободный (дата обращения: 13.01.2014).

18. Dehydra [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/en-US/docs/Dehydra, свободный (дата обращения: 13.01.2014).

19. Crossplatform, open source .NET development framework [Электронный ресурс]. – Режим доступа: http://www.mono-project.com, свободный (дата обращения: 13.01.2014).

20. C# Language Specification 5.0 [Электронный ресурс]. – Режим доступа: http://www.microsoft.com/en-us/download/details.aspx?id=7029, свободный (дата обращения: 13.01.2014).

21. Working Draft, Standard for Programming Language C++ [Электронный ресурс]. – Режим доступа: https://www.github.com/google/cxx-std-draft/blob/master/papers/n3337.pdf, свободный (дата обращения: 13.01.2014).

22. .NET Compiler Platform («Roslyn») [Электронный ресурс]. – Режим доступа: http://msdn.microsoft.com/en-us/vstudio/roslyn.aspx, свободный (дата обращения: 06.07.2014).

23. IntermediateRepresentation – Google Docs [Электронный ресурс]. – Режим доступа: https://docs.google.com/document/d/148mS6S-qoJhRfmIxDAytsQhAuTHGnavQDH62HLdc0ic/edit?usp=sharing, свободный (дата обращения: 06.07.2014).

24. Зубов М.В. Применение универсальных промежуточных представлений для статического анализа исходного программного кода / М.В. Зубов, А.Н. Пустыгин, Е.В. Старцев // Доклады Томского государственного университета систем управления и радиоэлектроники. – 2013. – № 1 (27). – С. 64–68.

Ошнуров Николай Андреевич
Аспирант каф. компьютерной безопасности и прикладной алгебры (КБиПА) ЧелГУ, г. Челябинск
Тел.: +7-950-735-79-42
Эл. почта: onlk@yandex.ru

Пустыгин Алексей Николаевич
Канд. техн. наук, доцент каф. КБиПА
Тел.: +7-905-835-98-68
Эл. почта: p2008an@rambler.ru

Ковалевский Алексей Анатольевич
Аспирант каф. компьютерной безопасности и прикладной алгебры ЧелГУ
Тел.: +7-951-814-68-77
Эл. почта: morskoyzmey@gmail.com

Oshnurov N.A., Pustygin A.N., Kovalevskiy A.A.

Universal intermediate representation construction of source code in C++ and C# languages

This article describes existing intermediate representations of source code. We describe the tools for receiving an abstract syntax tree for C++ and C# programming languages. We describe a new representation which contains semantic information and a transformation of this representation to a format available for processing by existing tools.

Keywords: intermediate representation, source code, static analysis, C++, C#.