Проект технологии извлечения знаний из исходных текстов на языках C++ и C# с использованием общего промежуточного представления

Пустыгин А.Н., Ошнуров Н.А., Ковалевский А.А.
Челябинский Государственный Университет

Исходные тексты C#

AST

Разбор исходных текстов

AST*

Преобразование разобранных текстов

AST*

...

...

ЗНАНИЯ

Исходные тексты C/C++

Исходные тексты C#

ПП-1

Разбор исходных текстов

Разбор исходных текстов

AST*

AST*

Преобр.

Преобр.

ПП

ПП

Анализ 1

Анализ 1

Анализ 2

Анализ 2

Анализ N

Анализ N

...

...

Исходные тексты

Разборщик исходных текстов

AST*

Анализатор N

...

Эквивалентные преобразования программ

...

УПП

- Предложить формат для построения промежуточного представления гетерогенных текстов на C/C++ и C#,
- Спроектировать инструменты для получения такого представления.

1. Универсальность - отображение всех синтаксических конструкций языков, содержащихся в стандарте
2. Расширяемость - возможность введения новых конструкций языка, при переходе на новую версию
3. Расширяемость для подключения синтаксических конструкций новых языков
4. Текстовый формат промежуточного представления, позволяющий использование инструментов для обработки текста

- SrcML - документо-ориентированное XML представление исходного кода, дополненное информацией из AST.
- Поддерживает языки:
  - C/C++ (CppML)
  - Java (JavaML)
  - C# (SrcML.Net)
- Промежуточное представление содержит полный исходный текст, встроенный между тегами

- Зависимость представления пространства имен от языка:

  Разделитель пространства имен в C# и Java обозначен как «.»

  Разделитель пространства имен в C++ обозначен как «::»

- Поскольку для обозначения типа применяется конструкция:

  `<type><name>type_name</name></type>,`

  то пользовательские типы не отличимы встроенных.

- Отсутствует атрибут или тег типа идентификатора, а также атрибут-ссылка на его объявление (в документации id_ref):

  `myVar → <name>myVar</name>`

- Не распознает ключевое слово this

```
this → <name>this</name>
```

- Не распознает модификаторы доступа

```
public → <name>public</name>
```

- Нет раздельного объявления для пользовательских типов и встроенных:

```
class Tool → <decl>class <name>Tool</name></decl>

int len →

<decl><type><name>int</name></type> <name>len</name></decl>
```

В связи с отсутствием удовлетворяющих заданным критериям формата промежуточного представления и инструментов для его извлечения был произведен поиск инструментов для извлечения AST как крайнего из доступных универсальных наборов данных об исходном тексте

Инструмент	Преимущества	Недостатки
ROSE Compiler	Поддержка C++ 11	Не является встроенным компилятором операционной системы
Eclipse CDT	IDE для статического анализа. Поддержка C++	Отсутствие поддержки C++11
Mozilla Dehydra & Treehydra (GCC GIMPLE)	Извлечение информации о некоторых узлах AST	Разработка приостановлена в 2010. Неполное и неэквивалентное исходному тексту AST
Clang и его библиотека libclang	Поддержка C++11, активное сообщество, поддержка Google, Apple, пришел на замену GCC в FreeBSD	-

Инструмент	Преимущества	Недостатки
Mono C# Compiler	Кроссплатформенность решения (Windows, GNU/Linux, MacOS, iOS, Android)	Несовместимость API разных версий компилятора.
ICSharpCode NRefactory	Поддерживает C# 5.0, предоставляет инструментарий для преобразования исходного кода	-

- JSON — текстовый формат обмена данными, основанный на JavaScript и обычно используемый именно с этим языком.
- YAML — человекочитаемый формат сериализации данных, близкий к языкам разметки, но ориентированный на удобство ввода-вывода типичных структур данных многих языков программирования
- XML — рекомендованный Консорциумом Всемирной паутины (W3C) язык разметки.

- XPath [W3C] - язык запросов для выборки и навигации по XML документу, вычисления его метрик
- XQuery [W3C] - функциональный язык запросов и трансформаций XML документа
- XSLT [W3C] - язык для трансформации XML документа в другие его эквивалентные представления (XML, SVG, PDF, PNG)
- XML-базы данных (Sedna [Apache License 2.0], BaseX [BSD]) - как единое хранилище знаний об исходном коде проекта

Текст XML:

```
<Project>
    <Class name="IHuman" kind="interface"></Class>
    <Class name="Doctor" kind="class"></Class>
    <Class name="Programmer" kind="class"></Class>
    <Class name="IAnimal" kind="interface"></Class>
    <Class name="Bird" kind="class"></Class>
</Project>
```

Пример запроса для получения списка интерфейсов в проекте:

```
//Class[@kind="interface"]/@name
```

Результат:

```
name=IHuman, name=IAnimal
```

Текст XML:

```
<Project>
    <Class name="IHuman" kind="interface"></Class>
    <Class name="Doctor" kind="class"></Class>
    <Class name="Programmer" kind="class"></Class>
    <Class name="IAnimal" kind="interface"></Class>
    <Class name="Bird" kind="class"></Class>
</Project>
```

Пример запроса для получения списка интерфейсов в алфавитном порядке:

```
for $x in doc('test')//Class[@kind="interface"]/@name
order by $x
return string($x)
```

Результат: `IAnimal, IHuman`

Исходные тексты

Генерация УПП

УПП

Генерация эквивалентного предложения

Эквивалентное предложение

БД УПП XML-Base

XPath запрос

Генерация эквивалентных предложений

Пользо

- Общие или частично совпадающие
  - Операторы потока управления (for, if, while, switch)
  - Арифметические операторы (+, -, *, ++)
  - Объявления классов, методов, их вызовов и т.д.
  - Директивы препроцессора
- Частные C++
  - Типы наследования
  - Шаблоны как средство метапрограммирования
- Частные C#
  - Linq
  - Делегаты
  - Атрибуты
  - async / await и т.д.

Исходный текст (условный переход):

```
if(a > 2) {   } else { }
```

Представление:

```
<If>
  <Condition>
      <op:Binary kind=”greater”>
          <VarRef ref=”1” name=”a” />
          <lit:Integer value=”2” />
      </op:Binary>
  <Condition>
      <Then />
  <Else />
</If>
```

Исходный текст (вызов метода объекта):

```
obj.Foo(1.5f);
```

Представление:

```
<Call>
  <Target>
      <VarRef ref=”1” name=”obj” />
  </Target>
  <Exec>
      <MethodRef name=”Foo” ref=”511” def=”512” decl=”513” />
  </Exec>
  <Args>
      <Arg><lit:Float value=”1.5f” /></Arg>
  </Args>
</Call>
```

Исходный текст (объявление переменной):

```
MyClass obj(); // C++
MyClass obj = new MyClass(); // C#
```

Промежуточное представление:

```
<VarDecl name=”obj” id=”500”>
 <Type name=”MyClass” kind=”class” ref=”100” />
 <Init>
     <Call>
          <Exec>
            <MethodRef name=”MyClass” ref=”110” def=”111” decl=”111”/>
          </Exec>
      </Call>
 </Init>
</VarDecl>
```

Исходный текст (объявление делегата):

```
public delegate void Foo();
```

Промежуточное представление:

```
<cs:Delegate id=”256” name=”Foo”>
  <Modifiers>
      <Modifier name=”public” />
  </Modifiers>
  <Return>
      <Type name=”void” kind=”void” />
  </Return>
</cs:Delegate>
```

Исходный текст (Linq запрос списка атрибутов Name объектов коллекции):

```
from x in col select x.Name;
```

```
<cs:Query>
  <cs:FromClause>
    <Arg id="128" name="x" />
    <InExpr><VarRef ref="129" name="col" /></InExpr>
    <cs:QueryBody>
      <cs:QueryBodyOperation>
      <cs:SelectClause>
        <Src><Call>
          <Target><VarRef ref="128" name="x" /></Target>
          <Exec><PropertyRef ref="130" name="Name" /></Exec>
        </Call></Src>
      </cs:SelectClause>
      </cs:QueryBodyOperation>
    </cs:QueryBody>
  </cs:FromClause>
</cs:Query>
```

Исходный текст (объявление шаблонного класса):

```
template<class T, int Size = 100>
class Set { };
```

Представление:

```
<GenericParams>
 <GenericParam name="T" kind="class" />
 <GenericParam name="Size" kind="type">
      <Type name="int" kind="internal" />
      <Default><lit:Integer value="100" /></Default>
  </GenericParam>
</GenericParams>
<Class name="Set" />
```

Всего 136 узлов из них

- 98 узлов общих для C++ и C#
- 32 узла специфичных для C#
- 6 узлов специфичных для C++

- Был предложен проект технологии для извлечения знаний из исходных текстов
- Был разработан формат универсального промежуточного представления для языков C/C++ и C#: http://tiny.cc/UIR