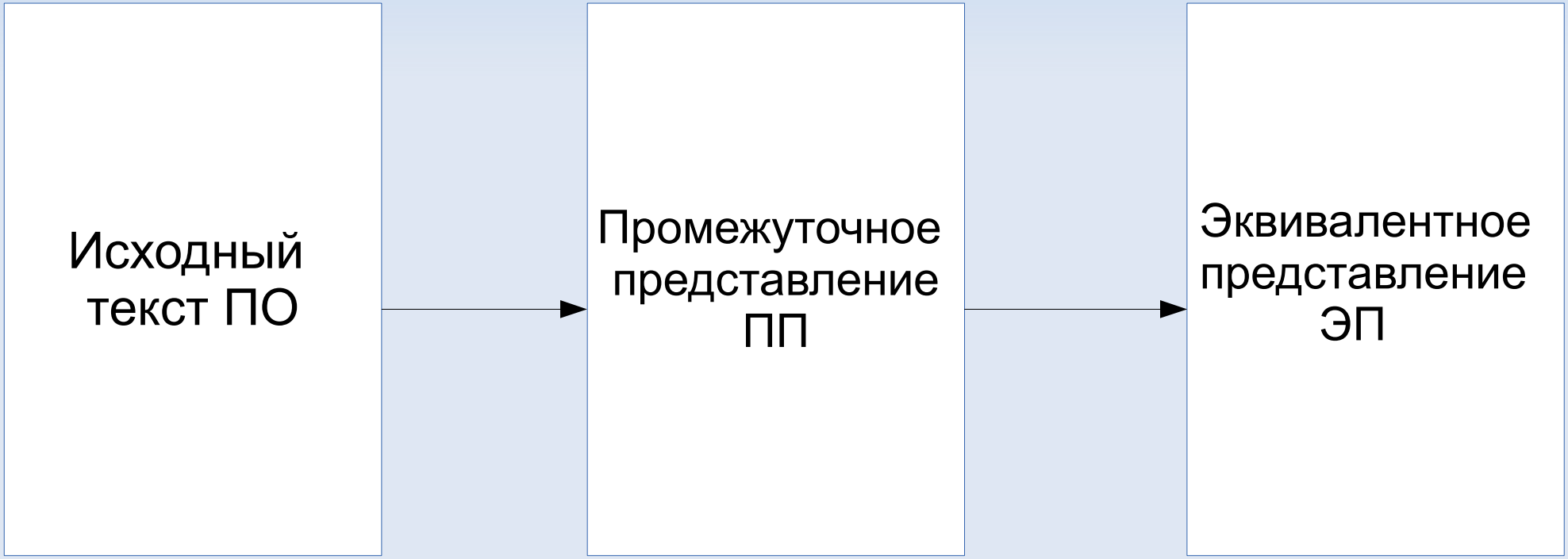
Использование системы выделенных признаков для задач поиска по исходному тесту

Пустыгин А.Н., Ковалевский А.А.
Челябинск
Челябинский государственный университет

Схема извлечения информации из исходных текстов



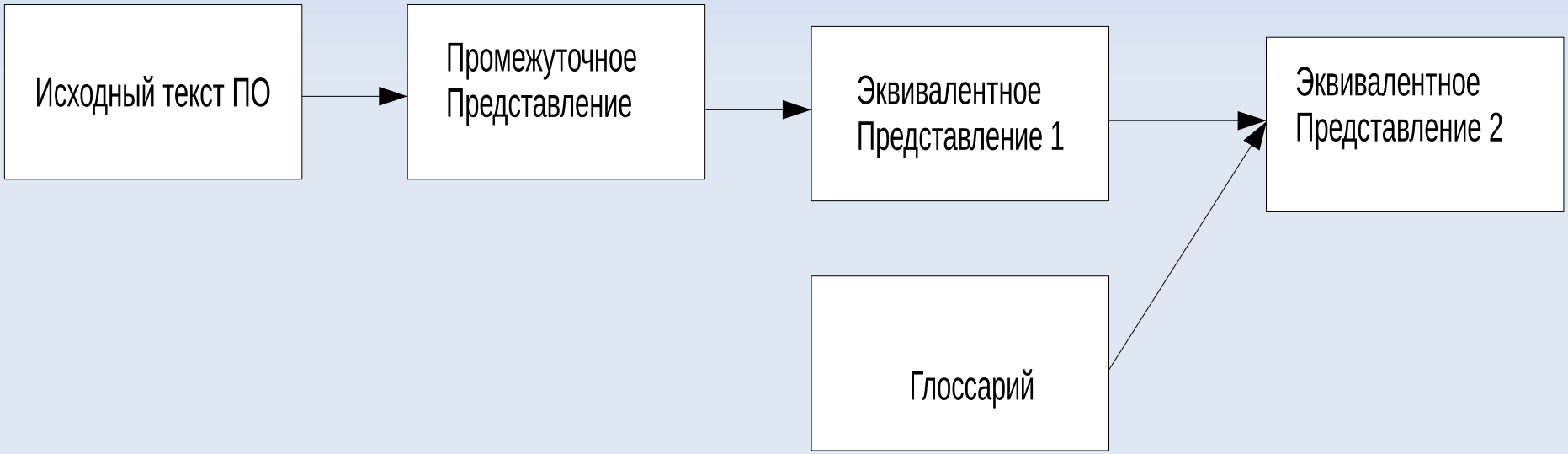
Промежуточное представление

набор данных оговоренного формата, предназначенный для последующего построения эквивалентных представлений исходного текста

Эквивалентное представление

набор данных оговоренного формата,предназначенный для последующего анализа исходного текста по тому или иному критерию

Схема преобразования исходного текста для задач поиска по синтаксическим признакам



Глоссарий признаков квантов исходного текста

Каждая категория содержит набор шаблонов, соответствующих искомым признакам квантов, в виде блока условий, записанных текстом на языке разметки XML

Формат записи условия в файле глоссария на языке разметки

<type name="<имя признака>" title="<описание признака>" [параметры признака] >

<node name="<имя тега>" [параметры условия] >

<attr name="<имя атрибута>" [параметры атрибута] />

…

</node>

…

</type>

Пример записи уловия глоссария в категории «метод» для признака «getter»

```
<node name="MethodDecl">
<attr name="const" vb="true" />  #метод является
                                  константным,
<attr name="argc" vi="0"/>        #метод не имеет
                                    аргументов,
</node>
        #поле объекта this напрямую возвращается
         оператором return
<node name="FieldRef" tags="return" parent="this" />
```

Предложенная система признаков разбивает достаточно крупный проект на некоторое количество уникальных групп квантов, т. е. групп методов или классов. Например, для проекта pugixml с 621 методом количество групп методов составляет 200, а для проекта jsoncpp с 532 методами количество групп методов равно 101

Примеры квантов исходного текста, найденные по выделенным признакам

1. Шаблон признака «функция-конвертер» (func-converter) в тестовом проекте pugixml

Строка	Имя метода
6273	convert_number_to_boolean
6026	tolower_ascii
6332	convert_number_to_string
1386	convert_buffer

Обнаружены методы, имеющие в названии составляющую «convert» (sic!) или характерный предлог «to» (файл pugixml.cpp)

Примеры квантов исходного текста, найденные по выделенным признакам

2. При поиске шаблона признака «проверяющая функция» (func-checker) в тестовом проекте pugixml находится функция «check_string_to_number_ format» (строка 6395, файл pugixml.cpp) с ключевым словом «check» в имени.

Примеры квантов исходного текста, найденные по выделенным признакам

3. При поиске шаблона признака «проверяющий метод» (checker) в тестовом проекте jsoncpp находятся методы, в названиях которых присутствует глагол «is»(файл json_value.cpp)

Строка	Имя метода
248	Value::CZString::isStaticString() const
1255	Value::isNull() const
1262	Value::isBool() const
1269	Value::isInt() const
1276	Value::isUInt() const
1292	Value::isDouble() const
1306	Value::isString() const

Например, метод Value::isBool в строке 1262 файла json_value.cpp сравнивает значение поля «type_» с константой «booleanValue» и возвращает булевый результат сравнения

Стр.	Исходный текст
1261	bool
1262	Value::isBool() const
1263	{
1264	return type_ == booleanValue;
1265	}

Примеры квантов исходного текста, найденные по выделенным признакам

4.При поиске шаблона «сравнивающий метод» (comparator) в тестовом проекте pugixml находятся методы, выполняющие функцию перегруженных операторов сравнения (файл pugixml.cpp)

Строка	Имя метода
3738	bool xml_attribute::operator==(const xml_attribute& r) const
3743	bool xml_attribute::operator!=(const xml_attribute& r) const
3748	bool xml_attribute::operator<(const xml_attribute& r) const
3753	bool xml_attribute::operator>(const xml_attribute& r) const
3758	bool xml_attribute::operator<=(const xml_attribute& r) const
3763	bool xml_attribute::operator>=(const xml_attribute& r) const
3974	bool xml_node::operator==(const xml_node& r) const
3979	bool xml_node::operator!=(const xml_node& r) const
3984	bool xml_node::operator<(const xml_node& r) const
3989	bool xml_node::operator>(const xml_node& r) const
3994	bool xml_node::operator<=(const xml_node& r) const
3999	bool xml_node::operator>=(const xml_node& r) const

Например,метод «operator==» в строке 3738 файла pugixml.cpp сравнивает поле «_attr» передаваемого по ссылке параметра «r» объектного признака «xml_attribute» и поле «_attr» данного объекта

Строка	Исходный текст
3738	bool xml_attribute::operator==(const xml_attribute& r) const
3739	{
3740	return (_attr == r._attr);
3741	}

Использование системы выделенных признаков для поиска

В задаче, когда не вполне формализуются условия поиска, можно отсеивать целые группы квантов, заведомо не соответствующих поисковым критериям

Например:

- при поиске методов, не имеющих определения (тела), можно исключить все группы квантов с признаком «имеет определение» (defined) (исключить 607 вхождений из 621 в pugixml);

- при поиске методов и функций, содержащих системные вызовы, можно исключить те, что не содержат вызовов вообще (!caller) (исключить 213 вхождений из 621 в pugixml), или те, что не содержат вызовов функций (!func-caller) (исключить 359 вхождений из 621 методов проекта pugixml).

- при поиске методов, модифицирующих поля объекта своего класса, можно отсеять все константные методы (исключить 160 вхождений из 621 в pugixml).

Частичные соответствия признаку позволяют выделять участки исходного текста, потенциально содержащие искомый код, если в глоссарии нет полностью соответствующего условиям поиска шаблона.

Поиск квантов исходного текста по заданному набору признаков

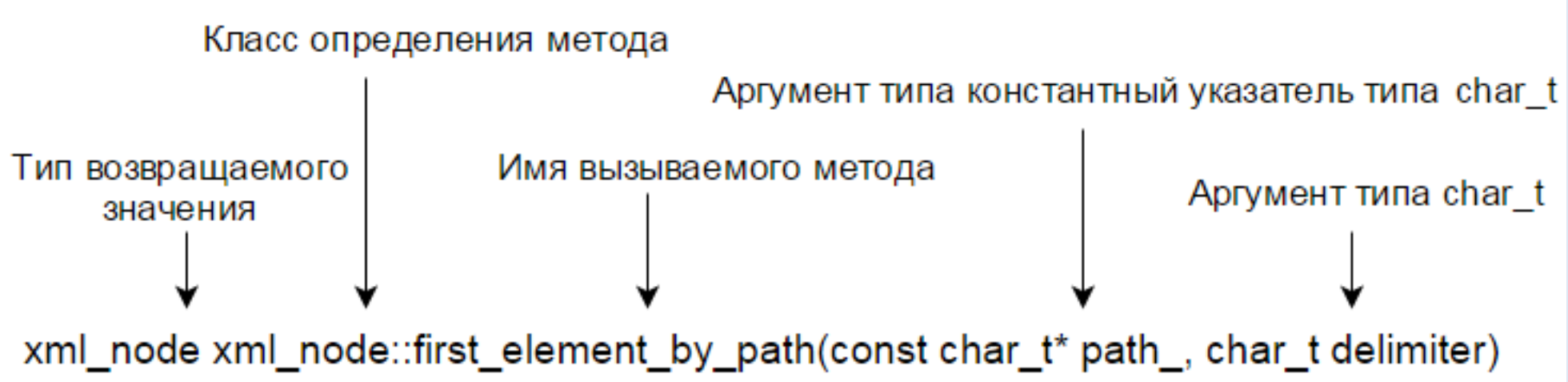
поиск позволяет фильтровать исходный текст, не используя имен идентификаторов и задавая только признаки искомого множества квантов исходного текста (в противовес текстовому поиску)

Для проекта pugixml были сформированы запросы

1. Найти константные методы, обрабатывающие аргументы в цикле.

Синтаксис запроса: const, method, cycled-arg-processor

Результат – строка 4550 в файле pugixml.cpp



2. Найти определения функций, содержащих несколько прямых рекурсивных вызовов.

Синтаксис запроса: function, defined, many-recursive

Строка	Заголовок определения функции
3079	void node_output(xml_buffered_writer& writer, const xml_node& node, const char_t* indent, unsigned int flags, unsigned int depth)
6551	const char_t* namespace_uri(const xpath_node& node)
6682	xpath_variable* new_xpath_variable(xpath_value_type type, const char_t* name)
6709	void delete_xpath_variable(xpath_value_type type, xpath_variable* var)
5619	template <typename I, typename Pred> void sort(I begin, I end, const Pred& pred)

3. Найти методы, которые устанавливают значения нескольких полей, копируя значения аргументов или одного аргумента.

Синтаксис запроса: common-setter, defined, method

Строка	Заголовок определения метода
421	void* xml_allocator::allocate_memory_oob(size_t size, xml_memory_page*&out_page)
1583	void push(char_t*& s, size_t count)
2853	void write(const char_t* data, size_t length)
2899	void write(char_t d0)
2907	void write(char_t d0, char_t d1)
2916	void write(char_t d0, char_t d1, char_t d2)
2926	void write(char_t d0, char_t d1, char_t d2, char_t d3)
2937	void write(char_t d0, char_t d1, char_t d2, char_t d3, char_t d4)
2949	void write(char_t d0, char_t d1, char_t d2, char_t d3, char_t d4, char_t d5)
5682	void* allocate_nothrow(size_t size)

4. Найти методы, которые вызывают методы своего класса, при этом не вызывают методов других классов или функций.

Синтаксис запроса: in-caller, !out-caller, !func-caller

Строка	Заголовок определения метода
4383	xml_node xml_node::append_child(const char_t* name_)
4392	xml_node xml_node::prepend_child(const char_t* name_)
4401	xml_node xml_node::insert_child_after(const char_t* name_, const xml_node& node)
4410	xml_node xml_node::insert_child_before(const char_t* name_, const xml_node& node)
…	и еще 61 метод

Объем исходного текста проекта pugixml составляет 300 Кб, при этом заранее неизвестны имена искомых квантов. Таким образом, поиск по указанным признакам без использования прототипа утилиты представляет собой полный просмотр исходного текста вручную во всех файлах по всем 11500 строкам.

Выводы

Предложенный подход выделения признаков показал свою применимость для задач поиска квантов исходного текста по сложным или плохо формулируемым условиям. Целью такого поиска может быть поиск недокументированных возможностей и потенциально ошибочных участков кода, при введении соответствующих признаков и категорий для искомых частей исходного текста (классов, методов, блоков).