Построение эквивалентного представления исходных текстов программ для выполнения анализов потока данных в потоке управления

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

Задачи построения эквивалентного представления

- Представление в форме пригодной для выполнения анализов потока данных в потоке управления
- Демонстрация возможности выполнения анализа распространения данных из заданного носителя в потоке управления метода

Критерии для разработки формата

1. Использование универсального промежуточного представления
2. Текстовый формат промежуточного представления, позволяющий использование инструментов для обработки текста

Существующие текстовые форматы представления структурированных данных

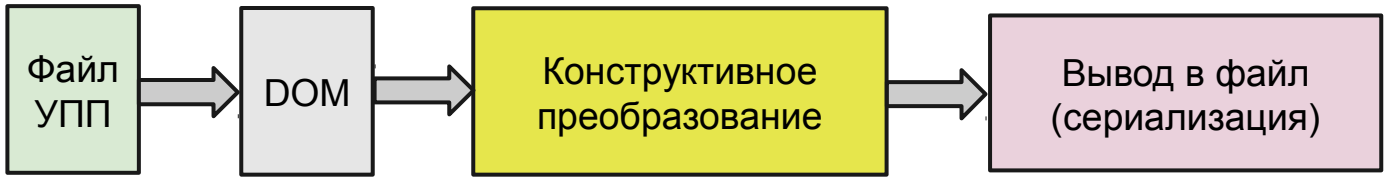
- JSON — текстовый формат обмена данными, основанный на JavaScript и обычно используемый именно с этим языком.
- YAML — человекочитаемый формат сериализации данных, концептуально близкий к языкам разметки, но ориентированный на удобство ввода-вывода типичных структур данных многих языков программирования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

Преимущества XML

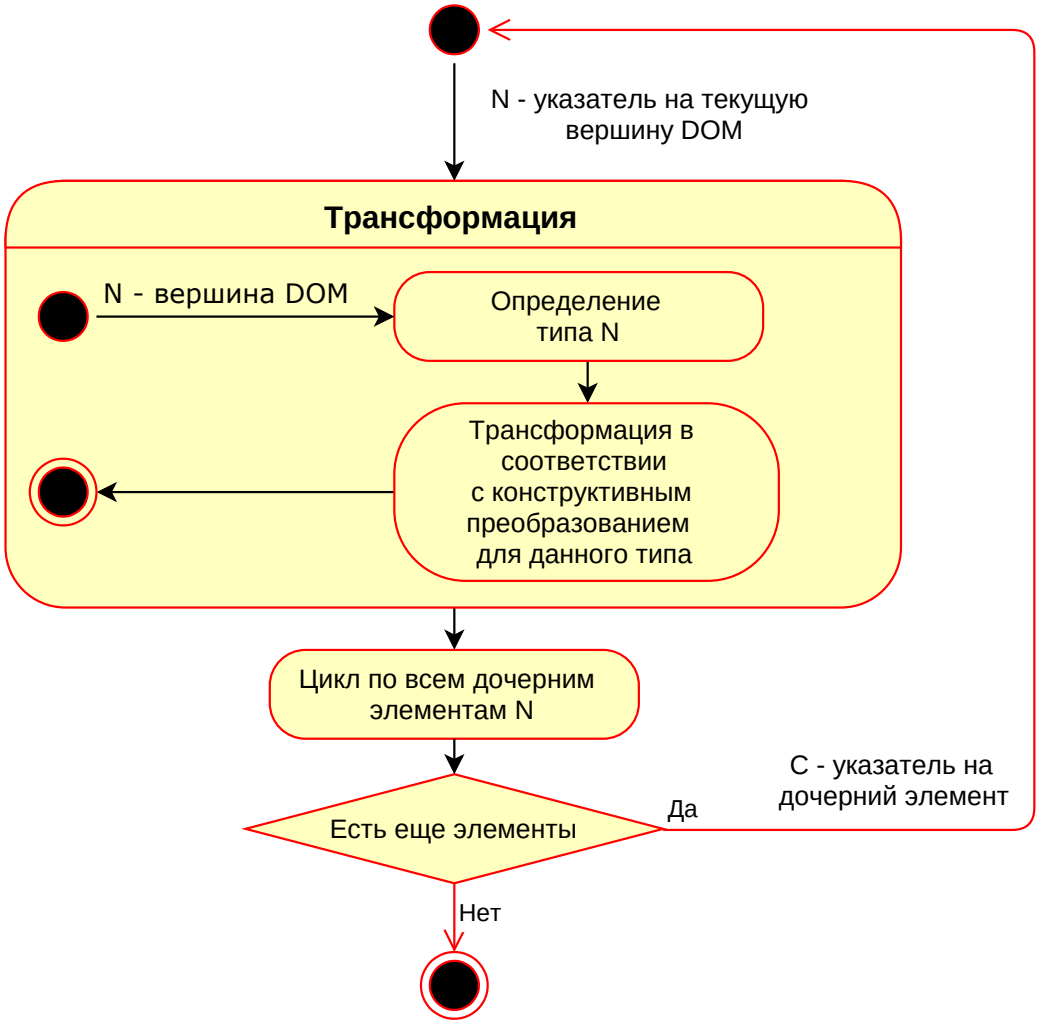
- XPath [W3C] - язык запросов для выборки и навигации по XML документу, вычисления его метрик
- XQuery [W3C] - функциональный язык запросов и трансформаций XML документа
- XSLT [W3C] - язык для трансформации XML документа в другие его эквивалентные представления (XML, SVG, PDF, PNG)
- XML-базы данных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

Схема преобразования УПП по данным и операциям над ними

Алгоритм получения представления однопроходный и выражается в трансформации имеющейся информации в УПП в абстрагированный набор операций над идентификаторами (носителями данных) в обертке операторов потока управления



Алгоритм конструктивного преобразования в UIRDCF



Пример записи в УПП для условного перехода

Исходный текст:

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

Промежуточное представление:

```
<If>
  <Condition>
      <op:Binary type=”>”>
          <Left><VarRef ref=”1” name=”a” /></Left>
          <Right><lit:Integer value=”2” text="2"/></Right>
      </op:Binary>
  <Condition>
  <Then />
  <Else />
</If>
```

Пример записи в УПП для вызова метода объекта

Исходный текст:

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

Промежуточное представление:

```
<Call>
 <Target><VarRef name=”obj” ref=”1” /></Target>
 <Exec>
  <MethodRef name=”Foo” ref=”101” />
 </Exec>
 <Args>
  <Arg type=”internal”>
   <lit:Decimical value=”1.5” text=”1.5f” />
  </Arg>
 </Args>
</Call>
```

Пример записи в УПП для объявления переменной пользовательского типа

Исходный текст:

```
MyClass obj();
```

Промежуточное представление:

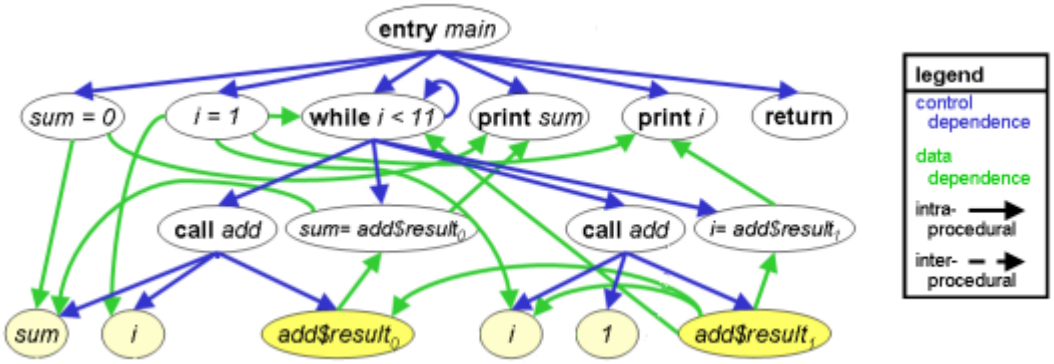
```
<VarDecl name="obj" id="500">
 <Type name="MyClass" kind="class" ref="1001" />
 <Init type="call">
      <Creation constructor="true">
        <Type name="MyClass" kind="class" ref="1001"/>
        <Exec>
          <MethodRef name="MyClass" ref="101" />
        </Exec>
        <Args/>
       </Creation>
 </Init>
</VarDecl>
```

Program Dependence Graph - известное эквивалентное представление, схожее с UIRDCF

Исходный текст

```
void main()
{
    int sum, i;
    sum = 0;
    i = 1;
    while ( i < 11 ) {
        sum = add(sum, i);
        i = add(i, 1);
    }
    System.out.println("sum = " + sum);
    System.out.println("i = " + i);
}
```

Эквивалентное представление PDG

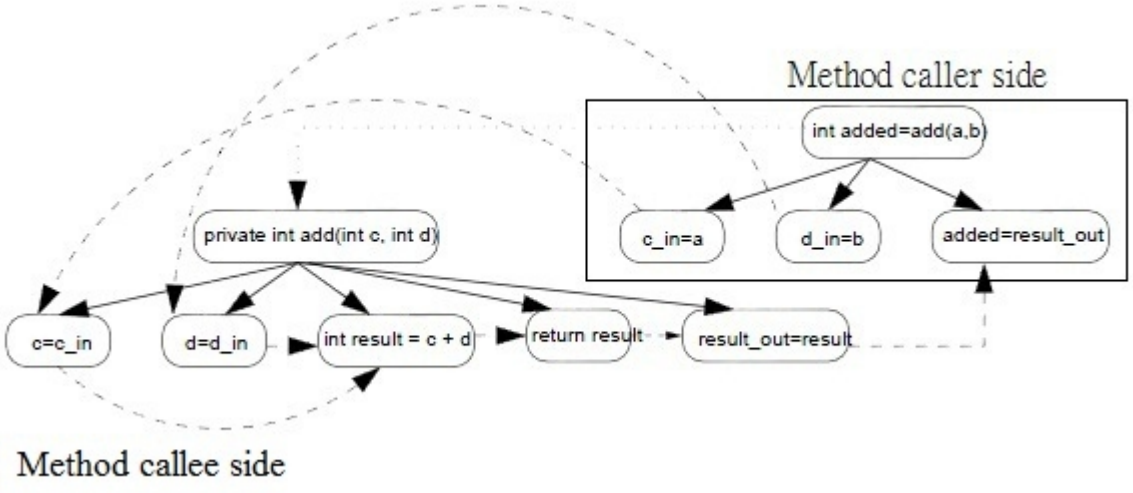


System Dependence Graph- известное эквивалентное представление, схожее с UIRDCF

Исходный текст

```
int added=add(a, b);
```

Эквивалентное представление SDG



Свойства существующих аналогов эквивалентных представлений

- Представляют как зависимости данных, так и зависимости потока управления.
- Узел является единичным оператором выполнения (single execution statement).
- Связи зависимостей данных (data dependence) строятся по факту наличия операций над носителями данных (переменными), которые упоминаются в каждом из узлов, между которыми эта связь устанавливается.
- Направление связи определяется потоком управления и AST.

Недостатки PDG и SDG

- Связи не атрибутируются информацией о типе производимой операции, просто связь содержит узел AST (несериализованного УПП) этой операции.

Пример записи разработанного эквивалентного представления для присваивания значения одной переменной другой

Исходный текст:

```
y = x;
```

Эквивалентное представление:

```
<Modify name="y" id="2"><Copy name="x" id="1"/></Modify>
```

Пример записи разработанного эквивалентного представления для условного перехода и возврата значения переменной

Исходный текст:

```
if(y < x) return x;
```

Эквивалентное представление:

```
<If>
 <Condition>
   <Read name="y" id="2"/>
   <Read name="x" id="1"/>
 </Condition>
 <Then>
   <Return><Copy name="x" id="1"/></Return>
 </Then>
</If>
```

Пример записи разработанного эквивалентного представления для вызова метода с передачей аргументов по ссылке

Исходный текст:

```
Swap(x, y);
```

Эквивалентное представление:

```
<Call name="Swap" id="40fff10">
 <Args>
  <Share name="x" id="1"/>
  <Share name="y" id="2"/>
 </Args>
</Call>
```

DRCF - эквивалентное представление второго уровня для анализа распространения данных в потоке управления

DRCF - Data Redistribution in Control Flow

Данное представление является трансформацией UIRDCF по заданным параметрам:

- Отслеживаемый идентификатор (носитель данных)
- Точка начала отслеживания (позиция в исходном тексте)
- Глубина отслеживания (не ограничена по умолчанию)
- Список имен методов с номерами и уровнями аргументов для отслеживания.

Формат записи в ЭП DRCF для анализа распространения данных в потоке управления

```
<Trace type="<операция>" level="<уровень отслеживания>" id="<ID носителя данных>" name="<имя носителя данных>" index="<номер узла>" line="<номер строки>">
  ...
</Trace>
```

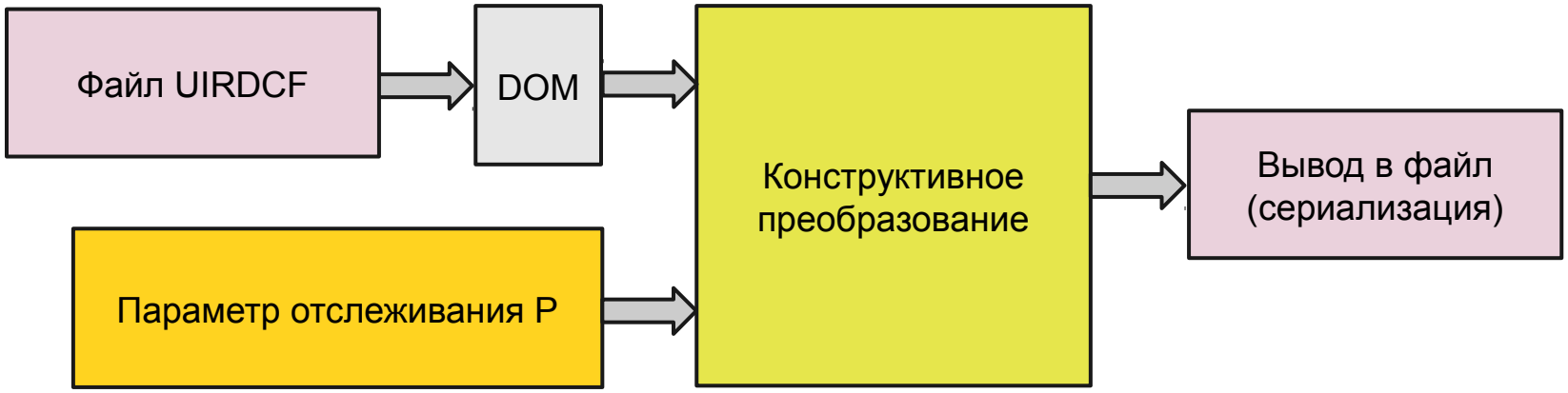
Носитель данных – идентификатор пользователя из списка отслеживаемых переменных, являющийся переменной произвольного типа.

Операция – тип производимой операции над носителем данных: read, copy, share, modify.

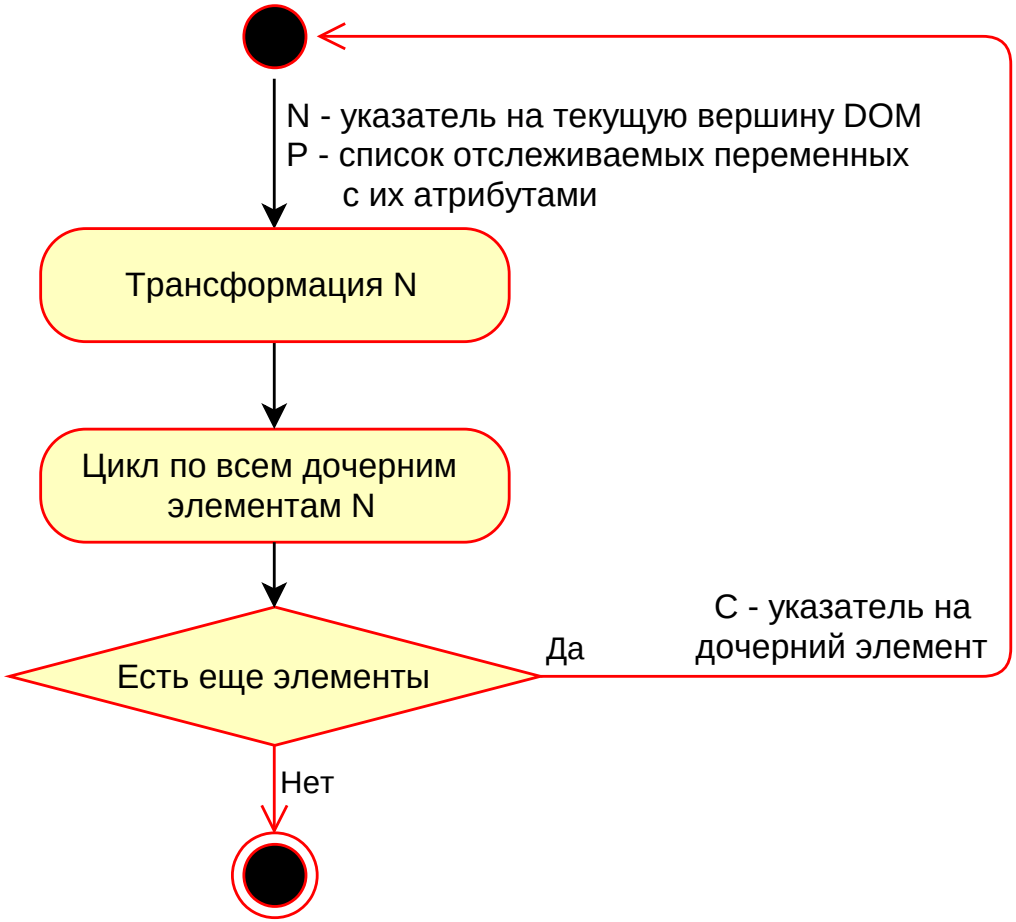
Уровень отслеживания – порядковый номер отслеживаемого носителя данных в списке отслеживаемых переменных

Номер узла – инцидентный номер узла в графе DOM для данного носителя

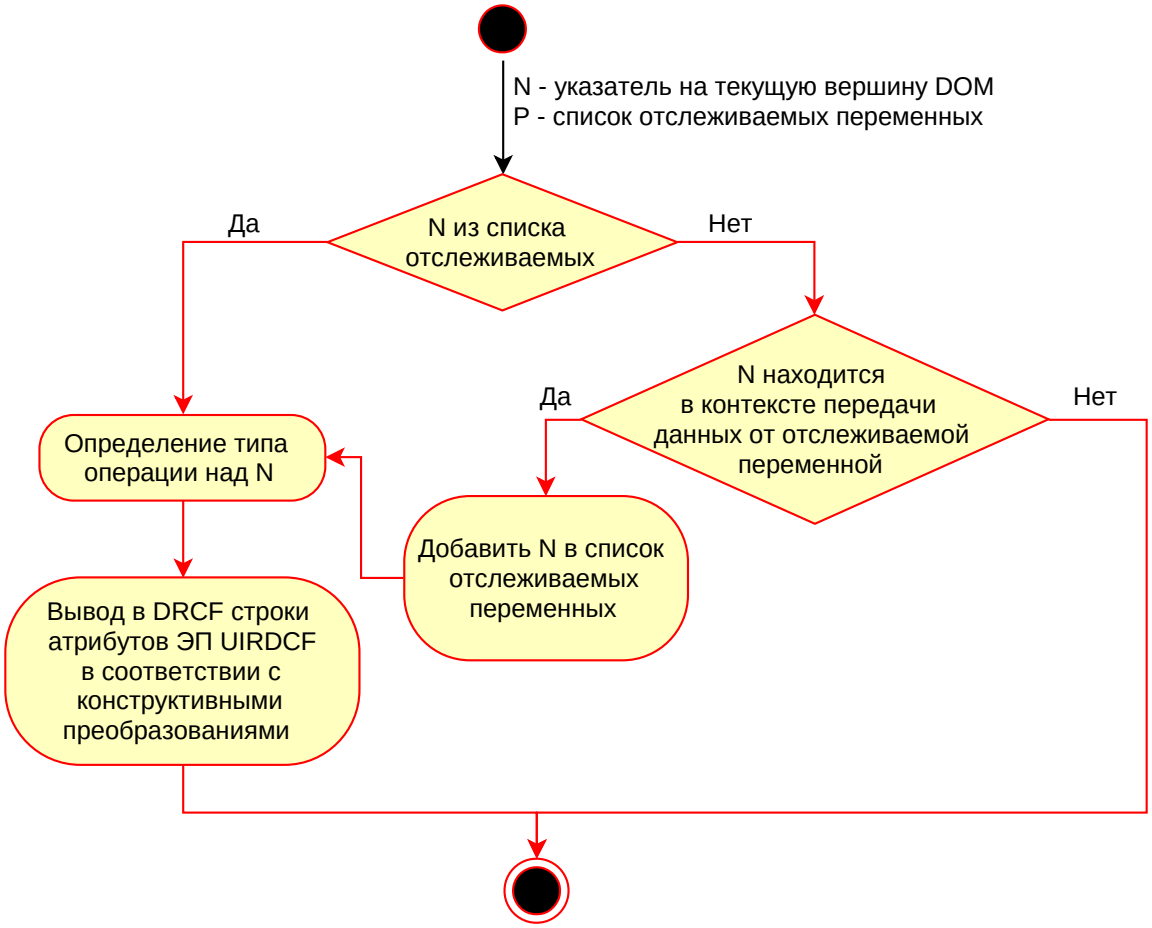
Схема преобразование набора данных UIRDCF в граф распространения данных в потоке управления DRCF



Алгоритм конструктивного преобразования UIRDCF в граф DRCF



Алгоритм трансформации текущей вершины N



Простейший пример анализа распространения данных в потоке управления для отслеживаемой переменной 'x'

Исходный текст:

```
1| int x = rand();
2| int z, y = rand();
3| if(y < x) y = x;
4| z = y;
```

Данные из переменной «x» распространяются не только в «y», но и, возможно, в переменную «z»

Эквивалентное представление:

```
<Trace type="decl" level="0" id="1" name="x" index="1" line="1"/>
<Trace type="read" level="0" id="1" name="x" index="2" line="3"/>
<Trace type="modify" level="1" id="2" name="y" index="1" line="3">
  <Trace type="copy" level="0" id="1" name="x" index="3" line="3"/>
</Trace>
<Trace type="modify" level="2" id="3" name="z" index="1" line="4">
  <Trace type="copy" level="1" id="2" name="y" index="2" line="4"/>
</Trace>
```

Сравнение предложенных представлений UIRDCF и DRCF с аналогами

- UIRDCF позволяет получить расширенную версию PDG и SDG, абстрагированную от AST.
- Различные операторы из AST в UIRDCF представлены классом производимой операции над данными.
- DRCF при этом можно сравнить с графом зависимостей данных из PDG, построенным для одной переменной, но с ответвлениями для побочных носителей.

Результаты

- Получено эквивалентное представление UIRDCF, на основании которого можно выполнять анализы потоков данных и управления
- Представление DRCF демонстрирует возможность анализа распространения данных из заданного носителя в потоке управления метода