Theory and Applications of Ontologies

CS6852

Study Report on
C ontology for program analysis

Anju MA **CS16D019**

Shikha Singh **CS16D008**

Swarnalakshmi Ravi **MS15D024**

May 4, 2017

Introduction

Program analysis is a fundamental research area which caters to many important tasks like program optimizations and debugging. A few examples of common static analyses which are performed at compile time are pointer analysis, loop dependence analysis, live variable analysis, reaching definitions analysis and shape analysis. The information collected by an analysis serves as the input to many possible optimizations like dead code elimination, register reuse, constant propagation, etc., which eventually will make the program run in less time and/or space.

Even though program analysis is essential to avoid bugs and reduce space/-time requirements of a program, performing various analyses are in general quite costly. In many cases, a pre-processing step needs to be done. Also, if different tools want to reuse the result of an analysis, in general it is difficult since the program representation varies considerably from platform to platform. Another difficulty faced by the current system is that the analyses done are very specific in nature that using the data collected for different optimizations is tough. In an effort to overcome these drawbacks, Zhao et al have come up with a new idea - making use of domain ontology to enhance program analyses.

Any programming language can be modeled as an ontology. The concepts, roles and their relationships can be described by means of the T-box. Every program written in that language can be converted into an A-box. The generated KB is useful in many application scenarios.
As an assignment, we studied the work done by [1] where they focused on the design of ontology for representing programs of C programming language.

Summary of C ontology

To model C program in this ontology, C99 standard has been followed and all the program constructs and concepts are enumerated in a top-down fashion. It contains 178 concepts and 68 properties in total. The top level concept is **CProgram** that contains different constructs such as **Statement**,**Type**, **Expression** and so on(as shown in Figure 1 below).

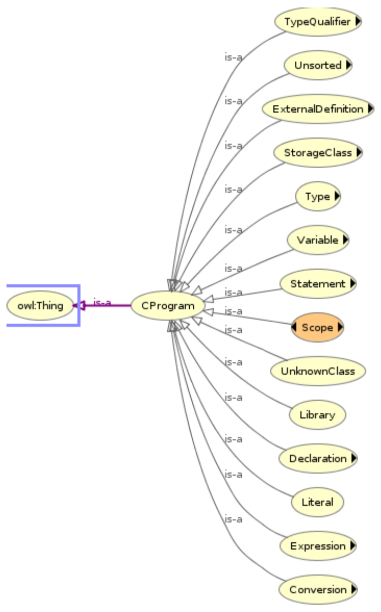
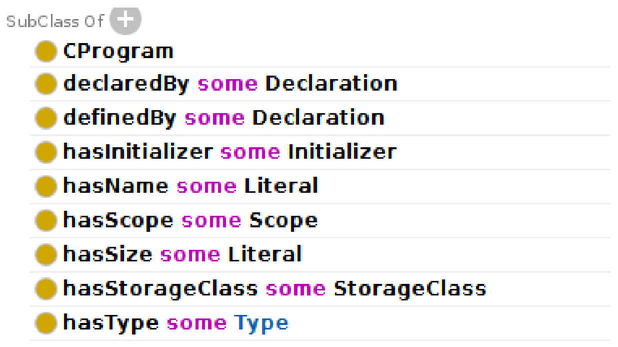


Figure 1: Top-level ontology for C program analysis

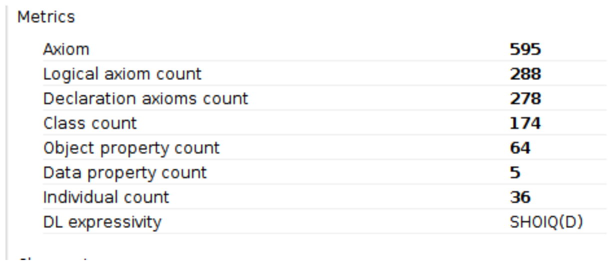
Declaration class contains declaration of different types of constructs as its subclass. Similarly, **Conversion** class contains different data-type conversions - **FloatDoubleConv**, **CharIntConv** etc as its subclass. **Statement** class has **CompoundStatement**,**IterationStatement**, **JumpStatement**, **ExpressionStatement** etc as its subclass. **StorageClass** has **Auto**, **Extern**, **Register**, **Static** as its subclasses as well as **auto**, **extern**, **register**, **static** as instances where auto is an instance of Auto and likewise. Even **Type** class also has predefined instances in its subclasses. For example : **float** is an instance of **FloatType**, **double** and **long_double** are instances of **DoubleType**.
Other classes are also defined according to C99 standard in a similar manner thereby enumerating the required sub-constructs as it subclasses. The definition of **Variable** as is shown as follows:



- CProgram
- declaredBy some Declaration
- definedBy some Declaration
- hasInitializer some Initializer
- hasName some Literal
- hasScope some Scope
- hasSize some Literal
- hasStorageClass some StorageClass
- hasType some Type

Figure 2: definition of variable class

The various properties defined in the ontology are **hasBody**, **declaredBy**, **accessedBy**, **hasType** etc. Some of the data properties are **hasName**, **hasValue** etc. The properties are defined in a generic way, and whenever possible, the semantic meanings are encoded into concepts. Thus a small set of properties can be combined to express a large number of possible properties in a program. The ontology has Annotation properties also. Few of them are : **comment**, **deprecated**, **incompatibleWith** etc.
The total number of the axioms, concepts and properties defined in the ontology is summarised in the figure below:


Metrics

Axiom	595
Logical axiom count	288
Declaration axioms count	278
Class count	174
Object property count	64
Data property count	5
Individual count	36
DL expressivity	SHOIQ(D)

Figure 3: C ontology metrics

Critique on the limitations and possible extensions of C ontology

The given ontology is implemented to represent C programs only, and not the formal definitions of common concepts in program analysis (e.g., dominator, post order, alias), but the purpose of this work [1] was to explore the potential of ontology based program analysis.
The design of the ontology is not complete.
For most of the properties, no range and domain information is provided. After studying it, we felt that not all possible restrictions in the domain, even for

the concepts and properties covered by the ontology, are imposed. for example **hasParameter** property has no range, no domain or any other information. Same is the case with other properties. The restrictions are very loose. While designing the A box for sample C program written below, we faced difficulty deciding classes for each element of code as well assigning object properties due to poor documentation on the restrictions.

- While writing ABox for our C Program, we asserted the comment as a statement as there was no other suitable concept to be assigned to it. But the reasoner finds it consistent. There should be some restriction that all statements must lie inside some block.
- Function definition should have returntype, functionName and parameterList, but in the given ontology has no provision for providing functionName by some property like hasFunctioName. Due to this, the statement which is invoking the function, we had to assert that the calling statement calls function definition only.
- No provision to define main() function as entry point to a program. We had to assert it as **FunctionDefinition** only.

The concepts have overlapping semantics and, the object properties defined in the Ontology do not capture all the possible relations among them.
One possible way of extension is to **first add concepts and relations of program analysis and then look for the required extension and refinement on the part of C ontology**. One of the possible reasons of leaving the ontology loose in terms of property restrictions could be to let it open to different application dependent constraints.

A Box for sample C Program

In the C Program below, we have tried to cover different constructs such as external function definition, Iterative statement, Conditional statements, nested operations etc.

/*C program to print all leap years from 1 to N.*/

```
#include <stdio.h>

//function to check leap year
int checkLeapYear(int year)
{
    if( (year % 400==0)||(year%4==0 && year%100!=0) )
        return 1;
    else
        return 0;
}
```

```
int main()
{

    printf("Leap years from 1 to 2017:\n");
    for(int i=1;i<=2017;i++)
    {
        if(checkLeapYear(i))
            printf("%d\t",i);
    }

    return 0;
}
```

A box Statements

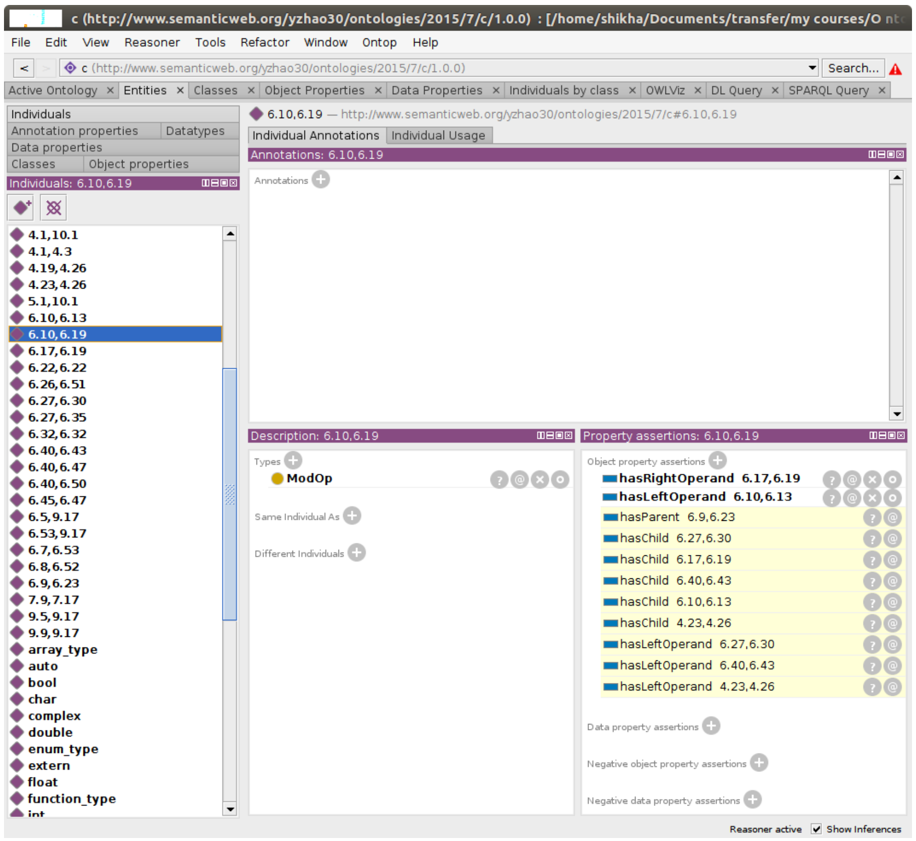
```
### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#0
c:0 rdf:type owl:NamedIndividual ,
             c:IntType .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#1
c:1 rdf:type owl:NamedIndividual ,
             c:IntType .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#1.1,1.18
c:1.1,1.18 rdf:type owl:NamedIndividual ,
                    c:Statement ;
           c:referTo c:1.10,1.18 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#1.10,1.18
c:1.10,1.18 rdf:type owl:NamedIndividual ,
                     c:Library .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#12.1,12.3
c:12.1,12.3 rdf:type owl:NamedIndividual ,
                     c:Type ;
            c:hasType c:int .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#12.1,23.1
c:12.1,23.1 rdf:type owl:NamedIndividual ,
                     c:FunctionDefinition ;
            c:hasReturnType c:12.1,12.3 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#13.1,27.1
c:13.1,27.1 rdf:type owl:NamedIndividual ,
                     c:CompoundStatement .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#15.5,15.43
c:15.5,15.43 rdf:type owl:NamedIndividual ,
                      c:Statement .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#16.17,16.23
c:16.17,16.23 rdf:type owl:NamedIndividual .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#16.25,16.27
c:16.25,16.27 rdf:type owl:NamedIndividual .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#16.5,20.5
c:16.5,20.5 rdf:type owl:NamedIndividual ,
                     c:ForStatement ;
            c:hasBody c:17.5,20.5 ;
            c:hasForIncr c:16.25,16.27 ;
            c:hasForInit c:16.9,16.15 ;
            c:hasForTest c:16.17,16.23 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#16.9,16.13
c:16.9,16.13 rdf:type owl:NamedIndividual ,
                      c:Variable ;
             c:hasScope c:16.5,20.5 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#16.9,16.15
c:16.9,16.15 rdf:type owl:NamedIndividual ,
                      c:VariableDecl .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#17.5,20.5
c:17.5,20.5 rdf:type owl:NamedIndividual ,
                     c:Block .
```

```
### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#18.12,18.17
c:18.12,18.17 rdf:type owl:NamedIndividual ,
                       c:Statement ;
              c:call c:4.1,10.1 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#18.9,19.28
c:18.9,19.28 rdf:type owl:NamedIndividual ,
                      c:IfStatement ;
             c:hasTrueBody c:19.13,19.29 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#19.13,19.29
c:19.13,19.29 rdf:type owl:NamedIndividual ,
                       c:Statement .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#22.5,22.13
c:22.5,22.13 rdf:type owl:NamedIndividual ,
                      c:ReturnStatement ;
             c:hasType c:0 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#4.1,10.1
c:4.1,10.1 rdf:type owl:NamedIndividual ,
                    c:FunctionDefinition ;
           c:hasBody c:5.1,10.1 ;
           c:hasParameterDeclList c:4.19,4.26 ;
           c:hasReturnType c:int ,
                           c:4.1,4.3 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#4.1,4.3
c:4.1,4.3 rdf:type owl:NamedIndividual ,
                   c:Type .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#4.19,4.26
c:4.19,4.26 rdf:type owl:NamedIndividual ,
                     c:FunctionParameterList ;
            c:hasParameter c:4.23,4.26 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#4.23,4.26
c:4.23,4.26 rdf:type owl:NamedIndividual ,
                        c:Variable ;
              owl:sameAs c:6.27,6.30 ,
                         c:6.40,6.43 ;
              c:hasScope c:5.1,10.1 ;
              c:hasType c:int .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#5.1,10.1
c:5.1,10.1 rdf:type owl:NamedIndividual ,
                    c:CompoundStatement .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.10,6.13
c:6.10,6.13 rdf:type owl:NamedIndividual ,
                     c:Variable ;
              owl:sameAs c:6.27,6.30 ,
                         c:6.40,6.43 ;
              c:hasType c:int .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.10,6.19
c:6.10,6.19 rdf:type owl:NamedIndividual ,
                     c:ModOp ;
              c:hasLeftOperand c:6.10,6.13 ;
              c:hasRightOperand c:6.17,6.19 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.17,6.19
c:6.17,6.19 rdf:type owl:NamedIndividual ,
                     c:Literal .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.22,6.22
c:6.22,6.22 rdf:type owl:NamedIndividual ,
                     c:Literal .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.26,6.51
c:6.26,6.51 rdf:type owl:NamedIndividual ,
                     c:LogicalANDOp .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.27,6.30
c:6.27,6.30 rdf:type owl:NamedIndividual ,
                     c:Type ;
              owl:sameAs c:6.40,6.43 ;
```

```
          c:hasType c:int .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.27,6.32
c:6.27,6.32 rdf:type owl:NamedIndividual ,
                     c:ModOp ;
          c:hasLeftOperand c:6.27,6.30 ;
          c:hasRightOperand c:6.32,6.32 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.27,6.35
c:6.27,6.35 rdf:type owl:NamedIndividual ,
                     c:ModOp ;
          c:hasLeftOperand c:6.27,6.32 ;
          c:hasRightOperand c:6.35,6.35 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.32,6.32
c:6.32,6.32 rdf:type owl:NamedIndividual ,
                     c:Literal .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.35,6.35
c:6.35,6.35 rdf:type owl:NamedIndividual ,
                     c:Literal .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.40,6.43
c:6.40,6.43 rdf:type owl:NamedIndividual ,
                     c:Type ;
          c:hasType c:int .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.40,6.47
c:6.40,6.47 rdf:type owl:NamedIndividual ,
                     c:ModOp ;
          c:hasLeftOperand c:6.40,6.43 ;
          c:hasRightOperand c:6.45,6.47 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.40,6.50
c:6.40,6.50 rdf:type owl:NamedIndividual ,
                     c:NeqOp ;
          c:hasLeftOperand c:6.40,6.47 ;
          c:hasRightOperand c:6.50,6.50 .
```

```
### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.45,6.47
c:6.45,6.47 rdf:type owl:NamedIndividual ,
                     c:Literal .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.5,9.17
c:6.5,9.17 rdf:type owl:NamedIndividual ,
                    c:IfElseStatement ;
           c:hasArgumentExpr c:6.7,6.53 ;
           c:hasBody c:6.53,9.17 ;
           c:hasElseBody c:9.5,9.17 ;
           c:hasTrueBody c:7.9,7.17 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.50,6.50
c:6.50,6.50 rdf:type owl:NamedIndividual ,
                     c:Literal .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.53,9.17
c:6.53,9.17 rdf:type owl:NamedIndividual ,
                     c:BasicBlock .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.7,6.53
c:6.7,6.53 rdf:type owl:NamedIndividual ,
                    c:ExpressionStatement ;
           c:hasExpression c:6.8,6.52 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.8,6.52
c:6.8,6.52 rdf:type owl:NamedIndividual ,
                    c:LogicalOROp ;
           c:hasLeftOperand c:6.9,6.23 ;
           c:hasRightOperand c:6.26,6.51 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.9,6.23
c:6.9,6.23 rdf:type owl:NamedIndividual ,
                    c:EqualityOp ;
           c:hasLeftOperand c:6.10,6.19 ;
           c:hasRightOperand c:6.22,6.22 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#7.9,7.17
```

```
c:7.9,7.17 rdf:type owl:NamedIndividual ,
                    c:ReturnStatement ;
           c:hasReturnType c:1 .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#9.5,9.17
c:9.5,9.17 rdf:type owl:NamedIndividual ,
                    c:Statement .


### http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#9.9,9.17
c:9.9,9.17 rdf:type owl:NamedIndividual ,
                    c:ReturnStatement ;
           c:hasReturnType c:0 .
```



Individuals: 6.10,6.19

4.1,10.1
4.1,4.3
4.19,4.26
4.23,4.26
5.1,10.1
6.10,6.13
6.10,6.19
6.17,6.19
6.22,6.22
6.26,6.51
6.27,6.30
6.27,6.35
6.32,6.32
6.40,6.43
6.40,6.47
6.40,6.50
6.45,6.47
6.5,9.17
6.53,9.17
6.7,6.53
6.8,6.52
6.9,6.23
7.9,7.17
9.5,9.17
9.9,9.17
array_type
auto
bool
char
complex
double
enum_type
extern
float
function_type
int

6.10,6.19 — http://www.semanticweb.org/yzhao30/ontologies/2015/7/c#6.10,6.19

Annotations: 6.10,6.19

Description: 6.10,6.19

Types
ModOp

Property assertions: 6.10,6.19

Object property assertions
hasRightOperand 6.17,6.19
hasLeftOperand 6.10,6.13
hasParent 6.9,6.23
hasChild 6.27,6.30
hasChild 6.17,6.19
hasChild 6.40,6.43
hasChild 6.10,6.13
hasChild 4.23,4.26
hasLeftOperand 6.27,6.30
hasLeftOperand 6.40,6.43
hasLeftOperand 4.23,4.26

Data property assertions
Negative object property assertions
Negative data property assertions

Figure 4: C Program A box

In the figure above, we can see that we provided one left operand and one right operand to the **ModOp** but it infers three more individuals as left operand. The reason for such inference is our assertion of those three individuals as **SameAs** the first one as all of them refer to the same variable **year** in the function.

References

[1] Yue Zhao, Guoyang Chen, Chunhua Liao, and Xipeng Shen. Towards ontology-based program analysis. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 56. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.