Towards an XML-based Bytecode Level Transformation Framework

Paper review

Vladimir Zakharov 01.08.2023

Meta Data

Conference: ENTCS (Electronic Notes in Theoretical Computer Science)

Keywords: Virtual machine, Java VM, Microsoft CLR, XML

Year: 2009

Number of Authors: 5

Cited By: 5

Pages (PDF): 15

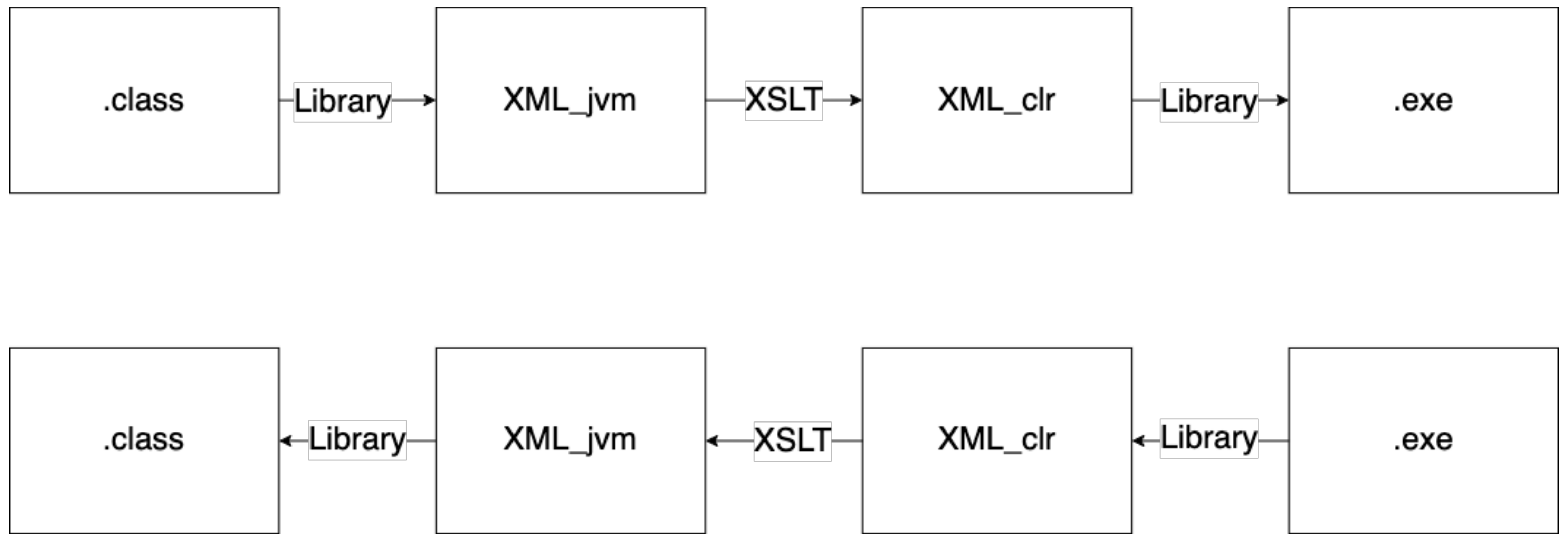
References: 11

Authors: Arno Pruder, Jessica Lee

San Francisco State University, Computer Science Department, 1600 Holloway Avenue,
San Francisco, CA 94132

What is this study about?

Bytecode Transpilation



XMLVM

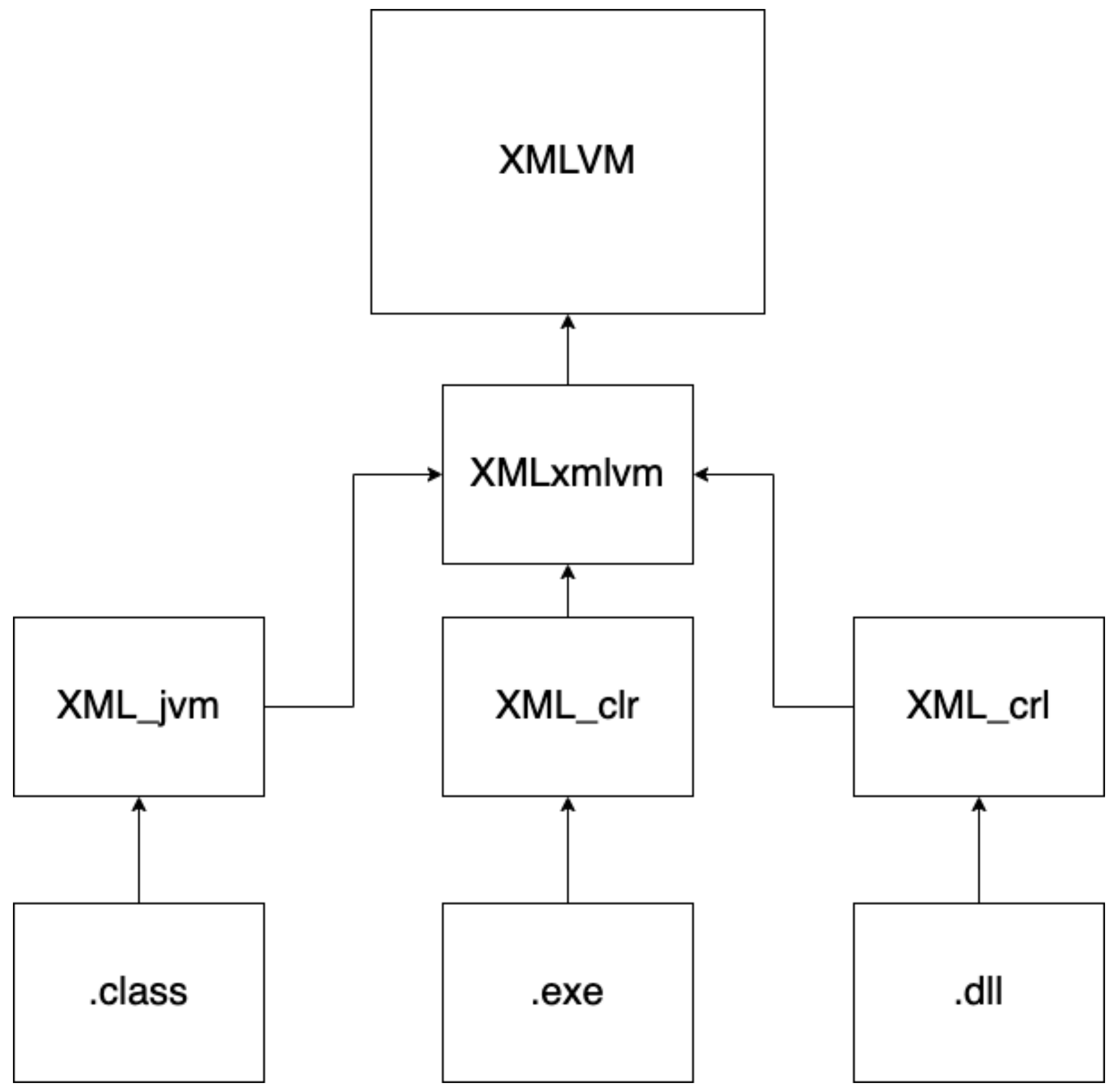
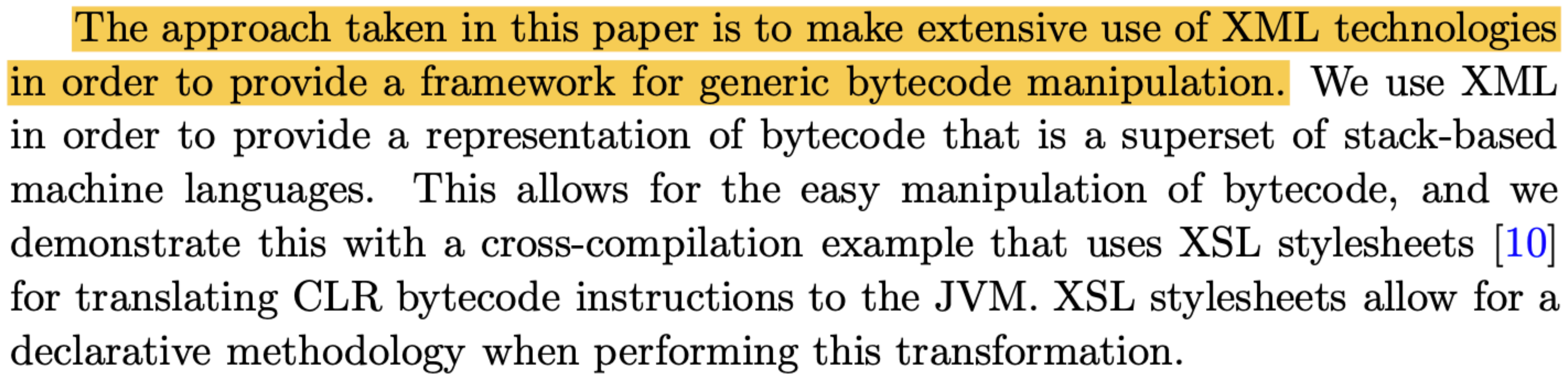
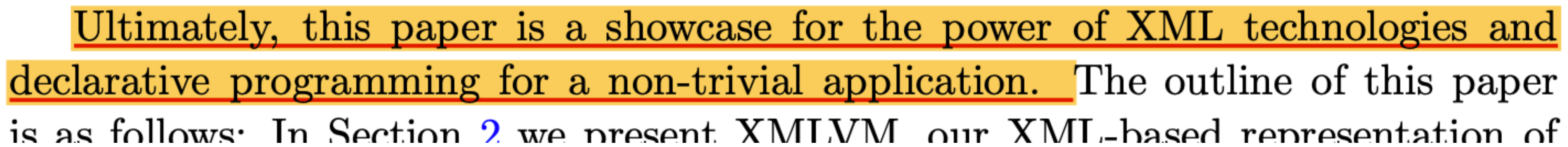


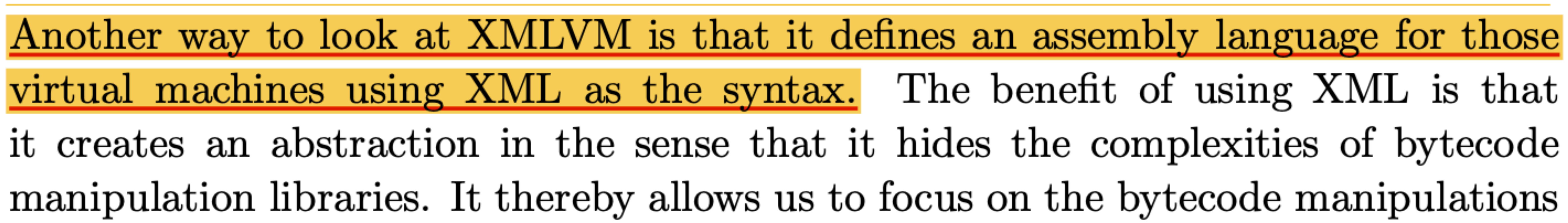
Table of Contents

Problem statement 👎

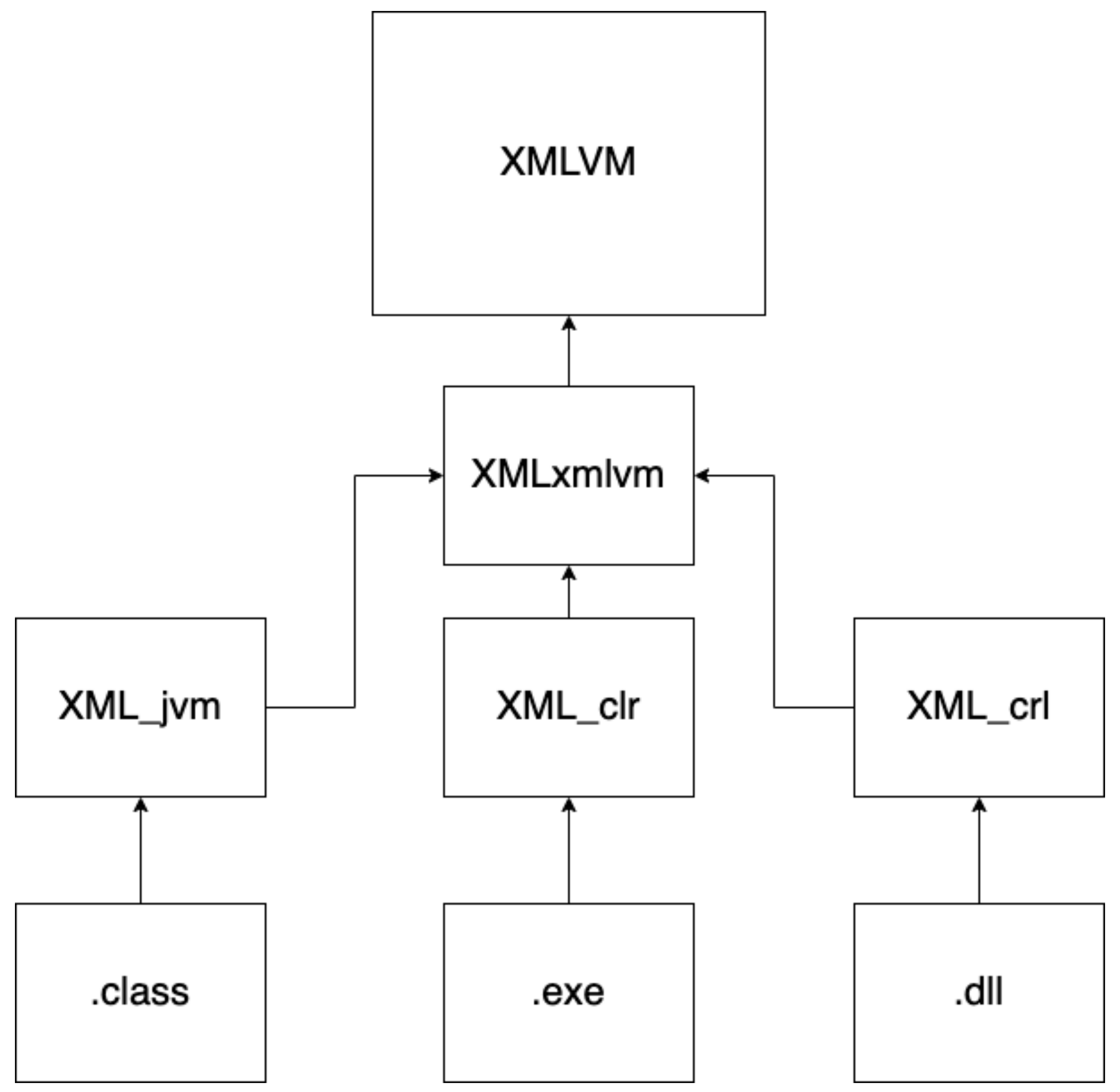

The approach taken in this paper is to make extensive use of XML technologies in order to provide a framework for generic bytecode manipulation. We use XML in order to provide a representation of bytecode that is a superset of stack-based machine languages. This allows for the easy manipulation of bytecode, and we demonstrate this with a cross-compilation example that uses XSL stylesheets [10] for translating CLR bytecode instructions to the JVM. XSL stylesheets allow for a declarative methodology when performing this transformation.


Ultimately, this paper is a showcase for the power of XML technologies and declarative programming for a non-trivial application. The outline of this paper is as follows: In Section 2 we present XMLVM, our XML-based representation of

Innovation 👎

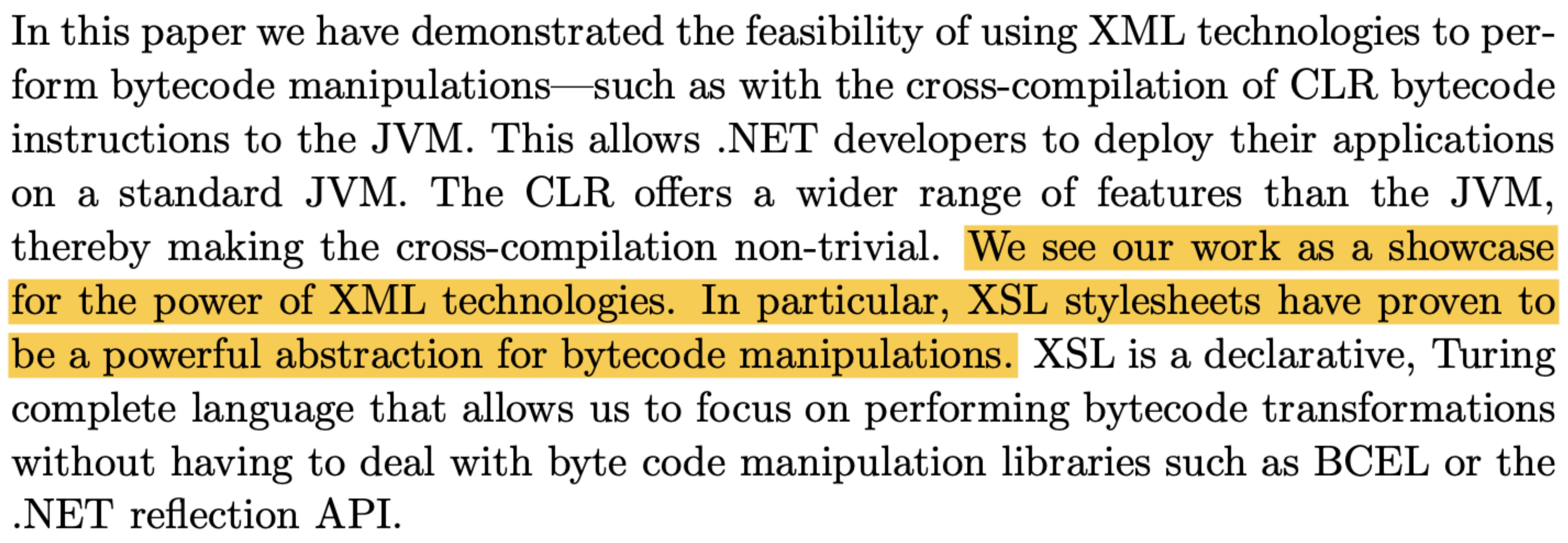


Another way to look at XMLVM is that it defines an assembly language for those virtual machines using XML as the syntax. The benefit of using XML is that it creates an abstraction in the sense that it hides the complexities of bytecode manipulation libraries. It thereby allows us to focus on the bytecode manipulations

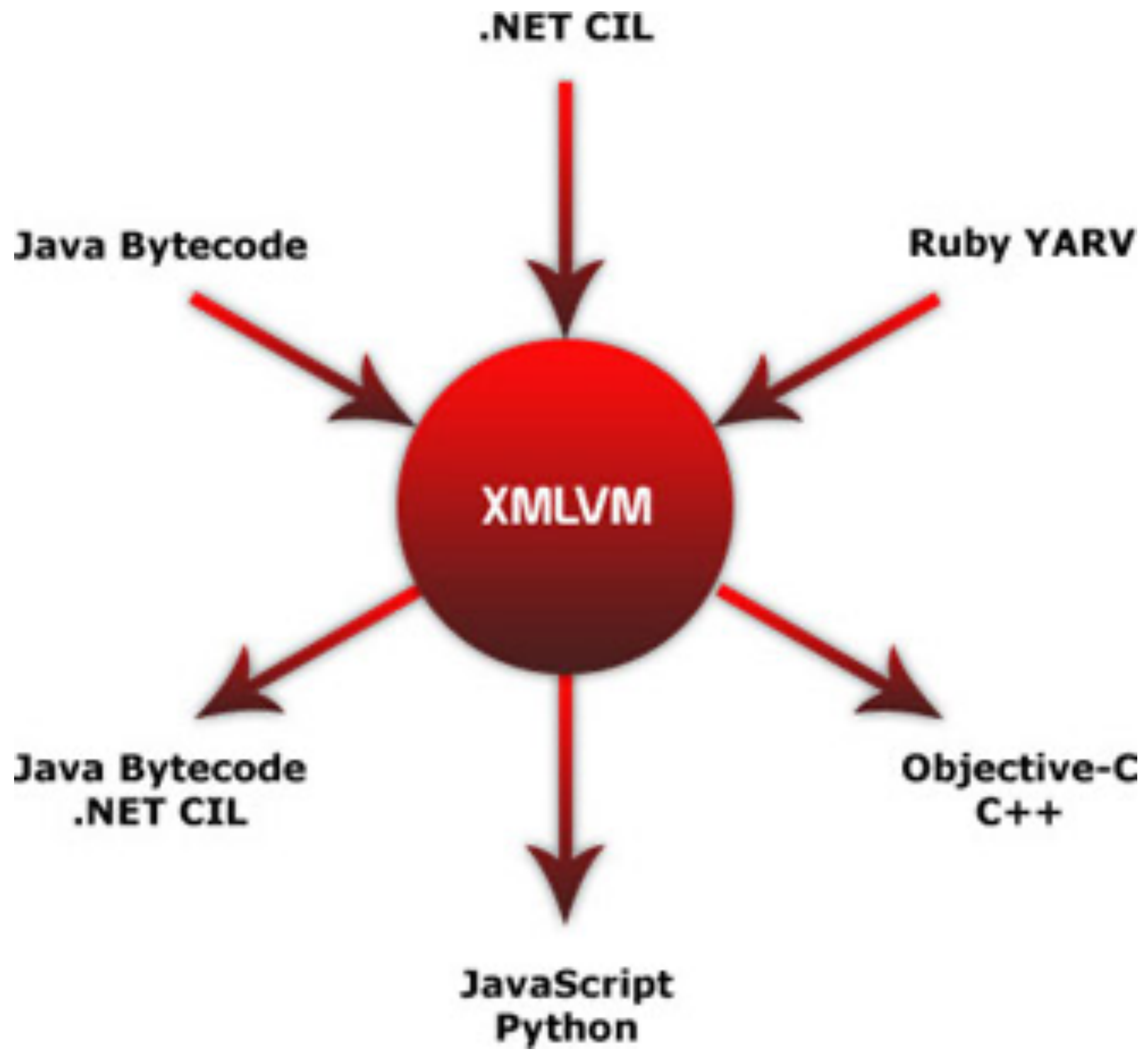


Contribution 👍

1. Cross-compile toolchain
2. http://www.xmlvm.org/overview/
3. https://github.com/shaeberling/xmlvm

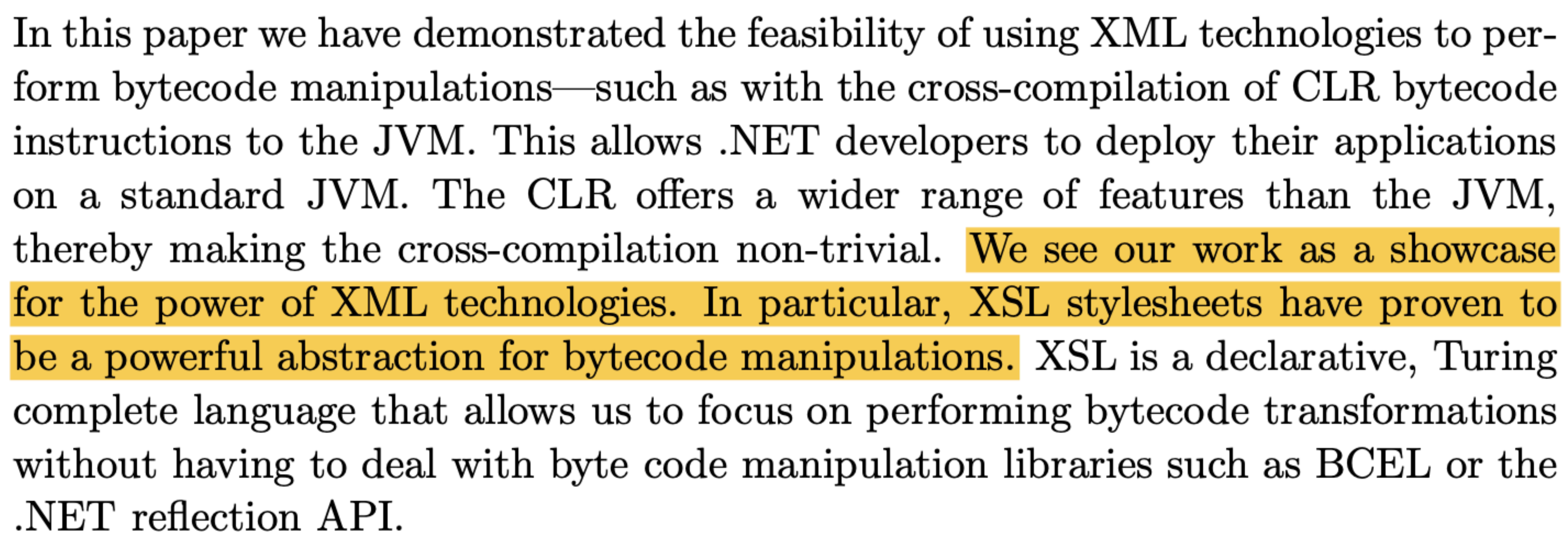


In this paper we have demonstrated the feasibility of using XML technologies to perform bytecode manipulations—such as with the cross-compilation of CLR bytecode instructions to the JVM. This allows .NET developers to deploy their applications on a standard JVM. The CLR offers a wider range of features than the JVM, thereby making the cross-compilation non-trivial. We see our work as a showcase for the power of XML technologies. In particular, XSL stylesheets have proven to be a powerful abstraction for bytecode manipulations. XSL is a declarative, Turing complete language that allows us to focus on performing bytecode transformations without having to deal with byte code manipulation libraries such as BCEL or the .NET reflection API.



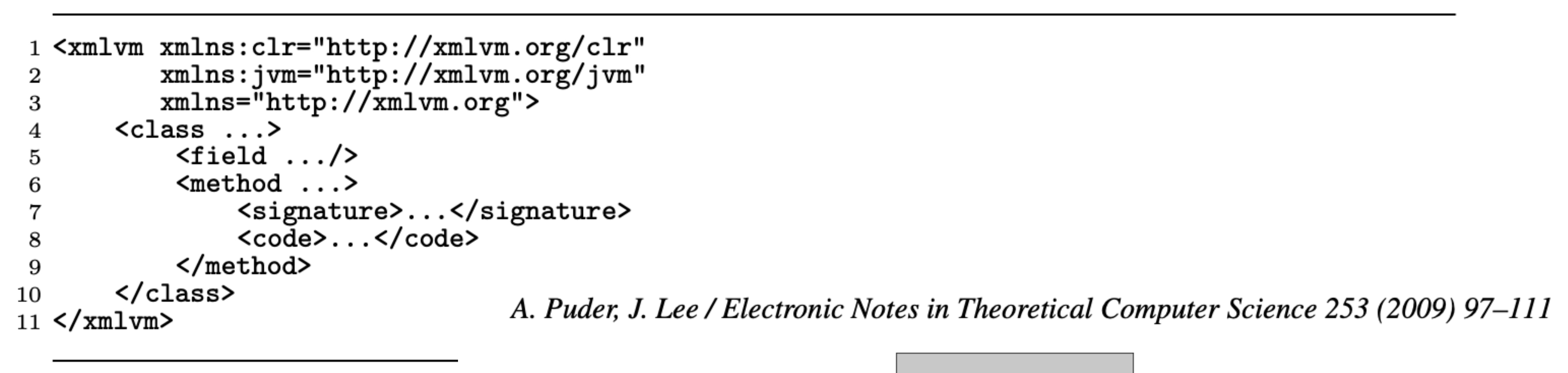
Logical correctness 👎
Proof of statements 👎

1. Related works?
2. Evaluation?
3. Scientific contribution?


In this paper we have demonstrated the feasibility of using XML technologies to perform bytecode manipulations—such as with the cross-compilation of CLR bytecode instructions to the JVM. This allows .NET developers to deploy their applications on a standard JVM. The CLR offers a wider range of features than the JVM, thereby making the cross-compilation non-trivial. We see our work as a showcase for the power of XML technologies. In particular, XSL stylesheets have proven to be a powerful abstraction for bytecode manipulations. XSL is a declarative, Turing complete language that allows us to focus on performing bytecode transformations without having to deal with byte code manipulation libraries such as BCEL or the .NET reflection API.

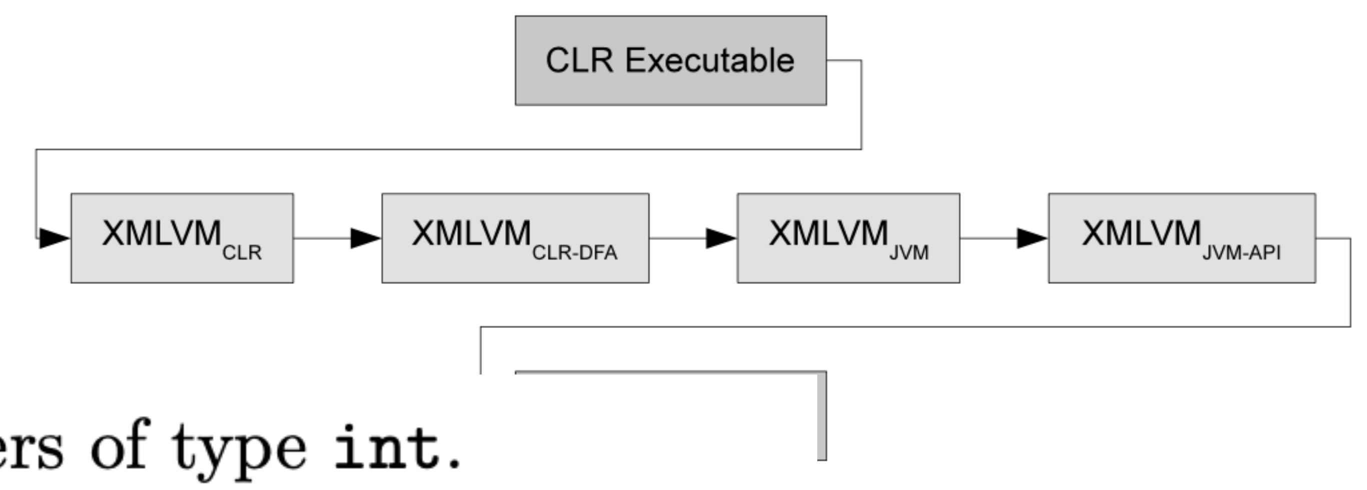
Readability 👍

1. Nice <u>engineering</u> article
2. Clear and logical narration
3. Many examples

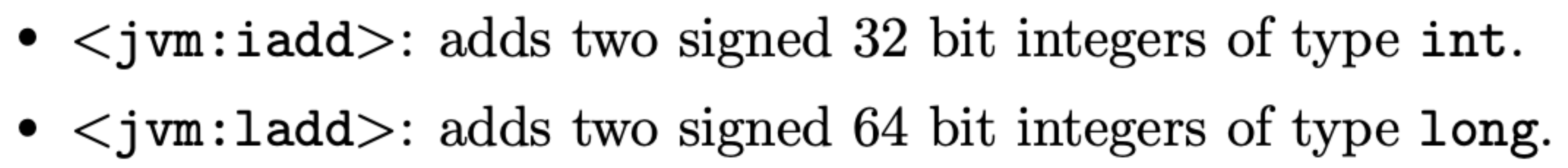


```
<xmlvm xmlns:clr="http://xmlvm.org/clr"
       xmlns:jvm="http://xmlvm.org/jvm"
       xmlns="http://xmlvm.org">
    <class ...>
        <field .../>
        <method ...>
            <signature>...</signature>
            <code>...</code>
        </method>
    </class>
</xmlvm>
```

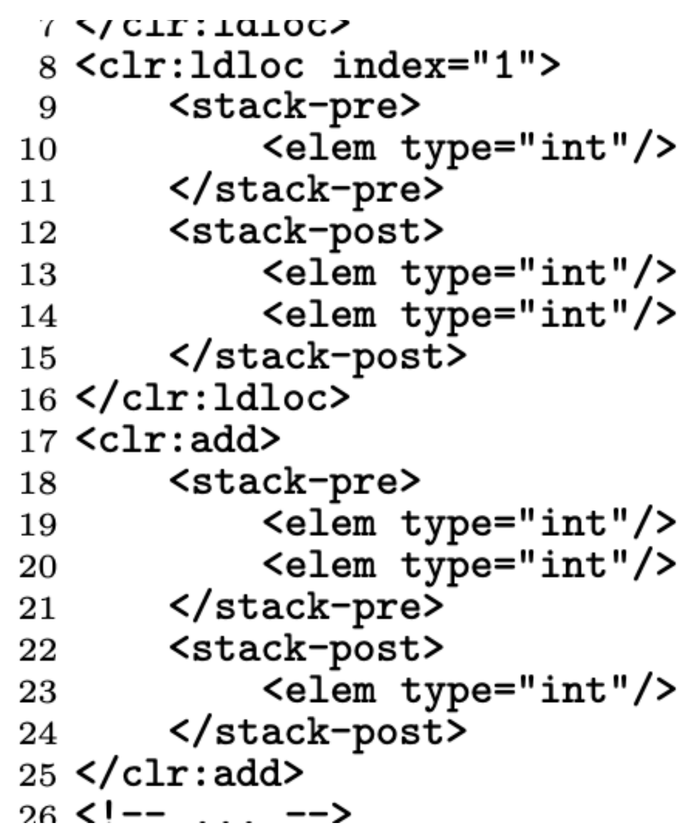
A. Puder, J. Lee / Electronic Notes in Theoretical Computer Science 253 (2009) 97–111



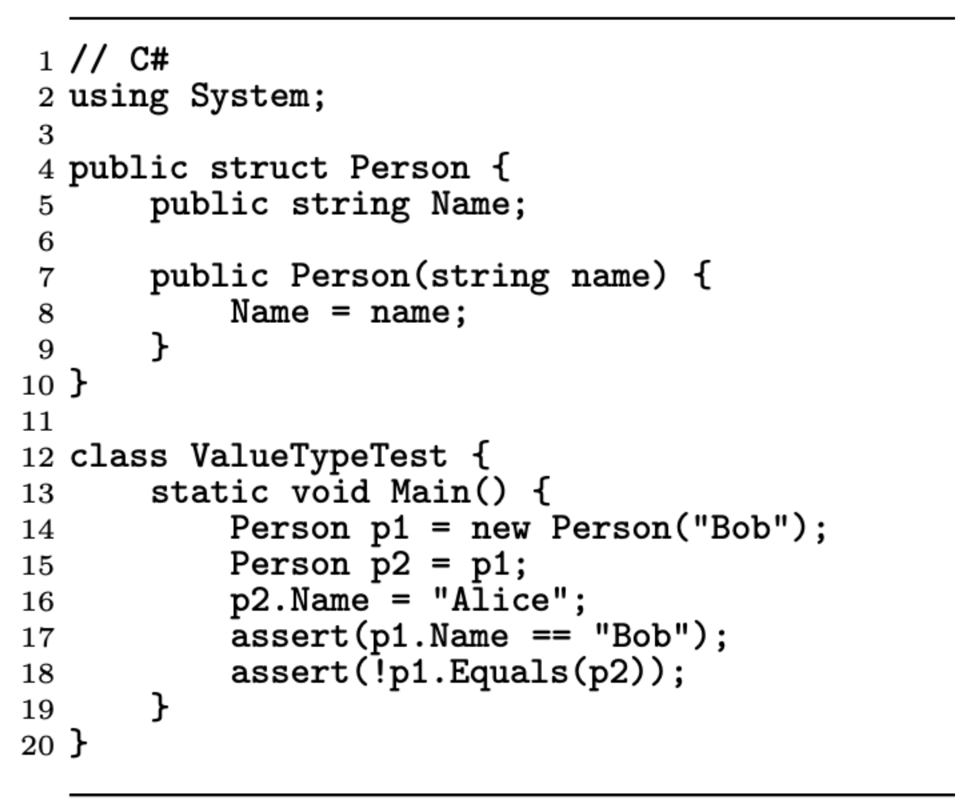
lchain.



- <jvm:iadd>: adds two signed 32 bit integers of type `int`.
- <jvm:ladd>: adds two signed 64 bit integers of type `long`.



```
</clr:ldloc>
<clr:ldloc index="1">
    <stack-pre>
        <elem type="int"/>
    </stack-pre>
    <stack-post>
        <elem type="int"/>
        <elem type="int"/>
    </stack-post>
</clr:ldloc>
<clr:add>
    <stack-pre>
        <elem type="int"/>
        <elem type="int"/>
    </stack-pre>
    <stack-post>
        <elem type="int"/>
    </stack-post>
</clr:add>
<!-- ... -->
```

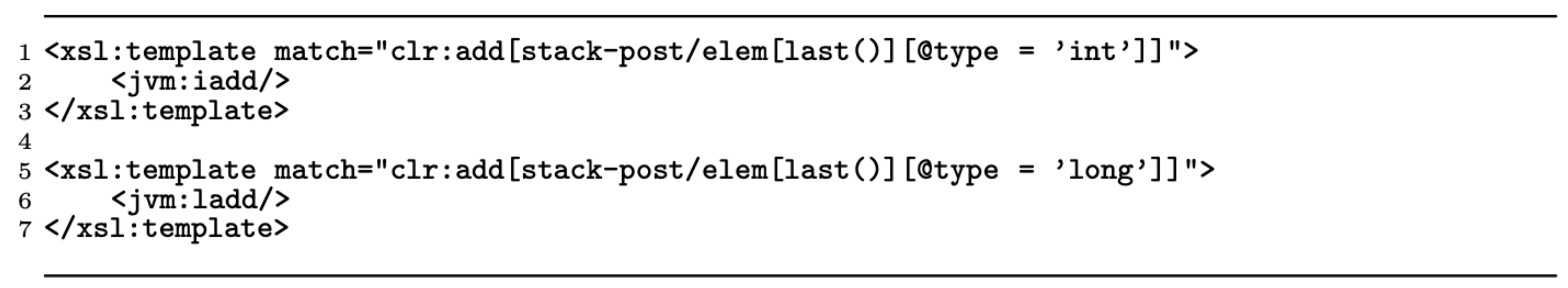


```
// C#
using System;

public struct Person {
    public string Name;

    public Person(string name) {
        Name = name;
    }
}

class ValueTypeTest {
    static void Main() {
        Person p1 = new Person("Bob");
        Person p2 = p1;
        p2.Name = "Alice";
        assert(p1.Name == "Bob");
        assert(!p1.Equals(p2));
    }
}
```



```
<xsl:template match="clr:add[stack-post/elem[last()][@type = 'int']]">
    <jvm:iadd/>
</xsl:template>

<xsl:template match="clr:add[stack-post/elem[last()][@type = 'long']]">
    <jvm:ladd/>
</xsl:template>
```

Conclusion

👎