

untrusted code is accompanied by a proof for its safety w.r.t. to some safety
property and the code receiver has just to generate $\text{table}(\text{prop})$

F e o

The overall objective is to allow a client to trust a code produced by an untrusted code producer. Our approach is especially suitable in cases where the client policy involves non trivial functional or safety requirements and thus, an automatic specification inference

e e o

We now review works which treat very similar problematic.

The JVer tool [8] is a similar tool for verifying that downloaded Java bytecode programs do not abuse client computational resources. The bytecode programs are annotated with

```

public class ListArray {
    Object[] list;
    //@requires list != null;
    //@ensures \result == ( \exists s int i; 0 <= i && i < list.length &&
        list[i] == o );
    public boolean isElem(Object obj)
    {
        int i = 0;
        //@loop_modifies i;
        //@loop_invariant i <= list.length && i

```

loop frame condition, which declares

result t = 1

→

→ var(0):

the loops in a metho

o e e e o e y e o e
oo

The purpose of this section is to give a comparison between bytecode and proof obligations. In particular, we illustrate the proof obligations of the example program in Fig.2.

We the relationship between the 2source code proof obligations generated by the 2standard

Hypothesis on bytecode:	Hypothesis on source level:
<code>l v[2]_at.ins_20</code> <code>len(#19(l v[0]))</code>	<code>i_at.ins_26</code> <code>len(ListArray:l</code>

e e e e

[1] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers-Principles, Techniques, and Tools*. [1] e

[15] G. C. Necula and P. Lee. The design and implementation of