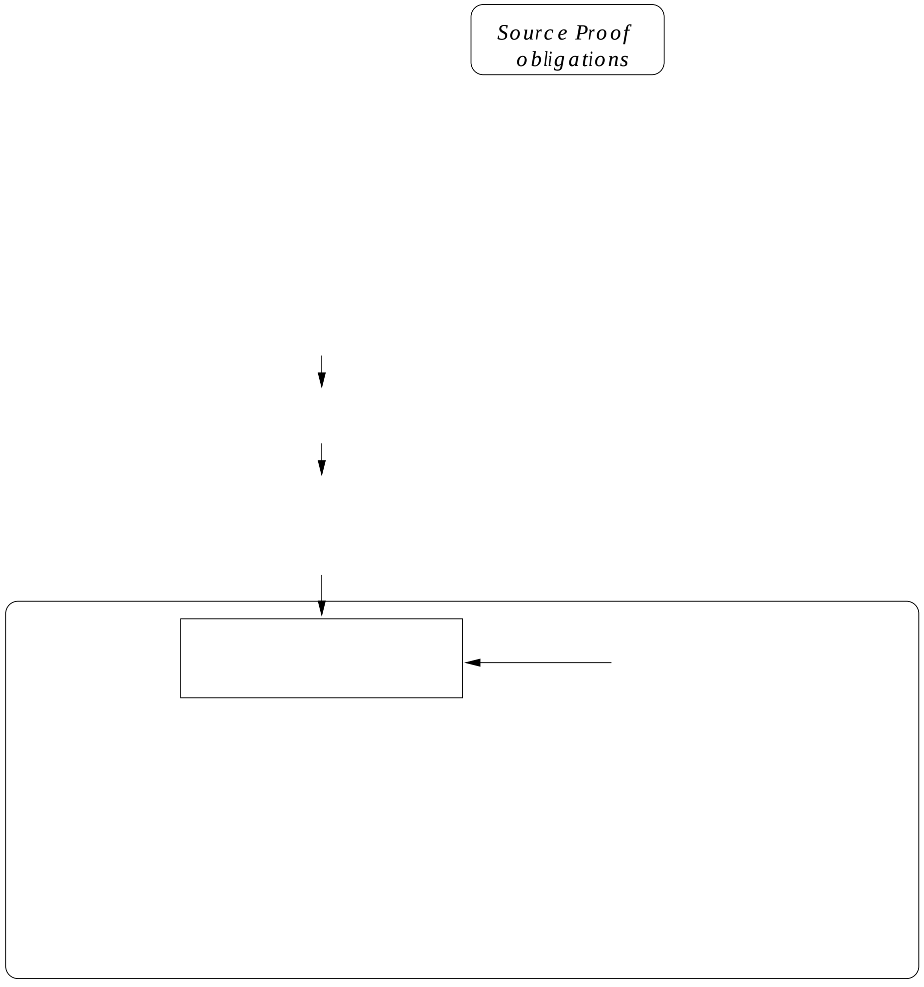
untrusted code is accompanied by a proof for its safety w.r.t. to some safety property and the code receiver has just to generatecode(prop)j 19.78989 0Td (the)Tj5.013.5956d (y)Tj -p1.8685 r' O

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 can not be applied. To this end, we propose a PCC technique that exploits the JML compiler and the weakest h250 (clien)Tj 19.6654 0 Td (t)TTj 9.4711


Source Proof
obligations

d o

We now review works which treat very similar h14Jlematic.

The JVer tool [8] is a similar tool for verifying that downloaded Java bytecode h14grams do not abuse client computational

```
public class ListArray {
  Object[]list;
  //@requires list != null;
  //@ensures \result ==  (\existsint 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 **= 1**

∃**var(0):**

the loops in a metho

5.1

o r son n so nd y od s oo s

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

We the relationship between the 2ource code proof obligations generated by the 2tandard

Hypothesis on bytecode:	**Hypothesis on source level:**
lv[2]**_at_ins_20** ≥ **len(#19(**lv[0]**))**	i**_at_ins_26** ≥ **len(ListArray:l**

References

[1] **A. V. Aho, R. Sethi, and J. D. Ullman.** *Compilers: Principles,* [illegible]

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