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 the (prop)j 19.78989 0Td (the)Tj5.013.5956d (y)Tj -p1.8685

2 Framework

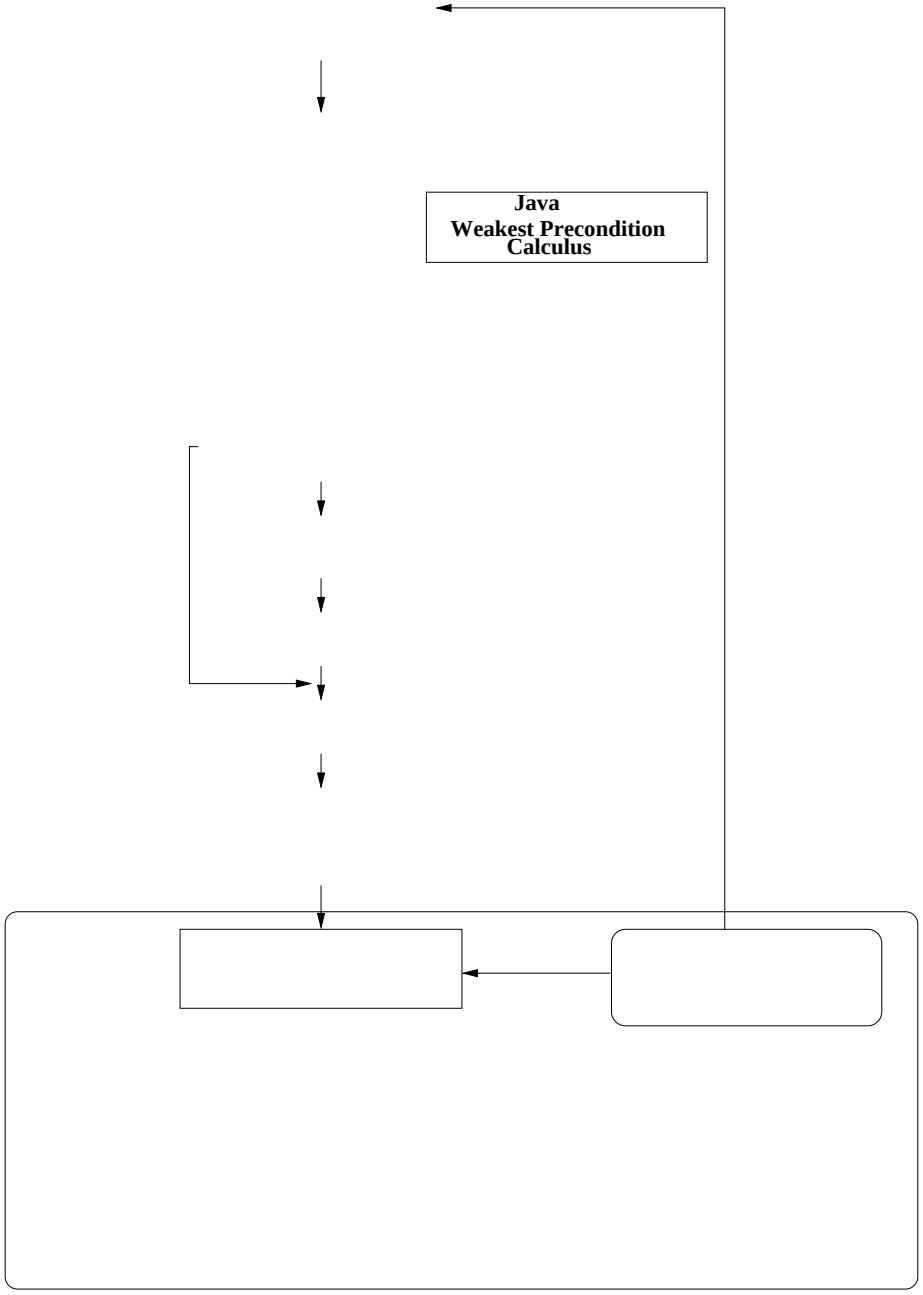
The overall objective is to allow a client to trust a code produced by an untrusted code producer. Our approach is especially suitable in cases where the client policy involves non trivial functional or safety requirements and thus, an automatic speci cation inference can not be applied. To this end, we propose a PCC technique that exploits the JML compiler and the weakest predicate function presented in the article.

The framework is presented in Fig. 1; note that certi cates are not yet implemented and thus are presented using special font and dotted lines.

In the rst stage of the process the client provides the functional and (or) security requirements to the producer. The requirements can be in di erent form:

- A speci ed interface that describes the application to be developed. In that case, the client speci es in JML the features that have to be implemented by the code producer.

- An API with Inf restricted access to some method. In this case, the client can protect its system by restricting the API usage. For example, suppose that the client API provides transaction management facilities. The API openj 10.3109 0 Td (syem)Tj 26.02820 Td (and)Tj 19.3067 0 Td (catho)Tj 27.3384 0 Td (
method open Inf


Java
Weakest Precondition
Calculus

3 Related Work

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

nresult **= 1**

( )

9var(0):

the loops in a metho

5.1

6 Comparison between source and bytecodes proofs

The purpose of this section is to give a comparison between bytecode and proof obligations. In particular, we illustrate this 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, T[1] e[2]

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