just to e erate the verificatio co tition s a t type check the proof a ai st

the bytecode specification language BCSL and the JML compiler; Section 4 discusses the main features of the weakest precondition calculus; Section 5 considers specifications for JML and other

theorem prover

The producer also transforms the source proofs into bytecode proofs. This transformation is based on the observation, that proof obligations on the source code and on optimized bytecode respectively are syntactically the same modulo names and basic types.

The producer delivers to the client the class files along with their BCSL annotations and the transformed proofs.

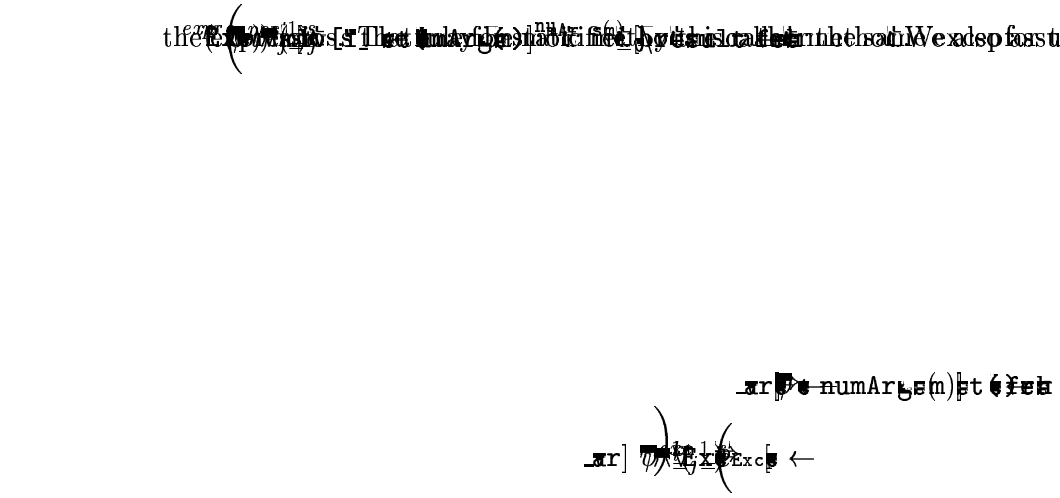
To implement this architecture we use JACK [4] as a verification condition generator both on the consumer and the producer site. J

```
public class ListArray {
  Object[] list;requires list != null;ensures \result ==  (\exists int i; 0 <= i
```

source li e a t the bytecote of a methot. The **Local_Variahl _Jahl** tescribes the local variables that appear i a methot. Those attributes are importa t for the ext phase of the JML compilatio .

2. from the source file a t the resulti class file compile the JML specificatio . I this phase, Ja r a t JMLsource ite tifiers are li ket ■ith their ite tifiers o bytecote level, amely ■ith the correspo t- i i texes either from the co sta t pool or the array of local variables tescribet i the **Local_Variahl _Jahl** stabute. If i the JML

\result



backet es i a methot's bytecote ■hich ca be itetifiet usi sta tartt tech iques ([1]).

Despite those cha es, the source a t bytecote oal respectively (hich are actually the postco titio) o bytecote a t source level are ot o ly

the Jack 1.8 release[4]. At this step, we have built a framework for Java pro-