source le. At this stage, the Java class les contain all the information that will allow the client to check if the bytecode does not violate his requirements. In particular, the client will generate proof obligations from the untrusted annotated bytecode

internal array list **contains the object referenced by the argument** obj **in its postcondition(**ensures**). The method loop is also speci ed by its invariant (**loop_invariant**) which basically says that whenever the loop entry is reached the elements inspected already by the loop are all di erent from** obj**.**

```
public class ListArray {
  Object[] list;
  //@requires list != null;
  //@ensures -Tj 6st ==
```

the instruction at which the loop invariant must hold (the loop entry instruction). this is di erent from JML where loop invariants are written at the beginning of the declaration of the loop statement, while the BCSL speci cation are separated from the bytecode

predicates from rst order logic

expressions from the programming language, like eld

nresul t **= 1**

( )

9v

ants, assertions at particular program point among which loop invariants (if
there is no explicite speci cation default one is taken into account: pre-
conditions, postconditions and invariants are taken to be true, exceptional
postcondition is by default false) is taken into account. In the rest of the
section, we consider the 9781 0 T5454 0 Td (sp)Tj 10.7781 0 Td (eci c)Tj 26.5854 0 Td (features)Tj 40.4727 0 Td

The Java bytecode language is stack based, i.e. the instructions take
their arguments from the method execution stack and put the result on the
stack. In Fig. 5 we show the wp rules for som 9781 0 T1963 0 Td (sp)Tj 5.7491 0 Td (bytecode)Tj 25.235

_load i

incrj 32.2114[2.6327 0 Td (ts)Tj] 12.6981 0 Td (the)Tj 19.4073 0 Td (stac)Tj 18.3164 0 Td (k)Tj 10.065

wp(invoke m**;** **;** ^{exc}**) =**

$\,^{e}$**(m) ^**

8$_{j=1..}$ **e**$_{j}$**:(**

$^{o\ \ t}$**(**m**)[**lv[i] st(c + i - numArgs(m))**]**$_{i=0}^{\text{numArgs(m)}}$

[nresul t fresh_var**]**

) **[**c c numArgs**(**m**)][**st(c) fresh

which the JACK source veri cation condition generator will discard.

Hypothesis on bytecode:	**Hypothesis on source level:**
lv[2]_**at_ins_20**	i_**at_ins_26**

[4] L. Burdy and M. Pavlova. From JML to BCSL. Technical report, INRIA, Sophia-Antipolis, 2004. draft.

[5] L. Burdy, A. Requet, and J.-L. Lanet. Ja18.0982 0 Td (E)Tj FA developer-oriented pproach. In K. Araki, S. Gnesi, and D. Mandrioli, editors, FME er2003Formal Methods: International Symp