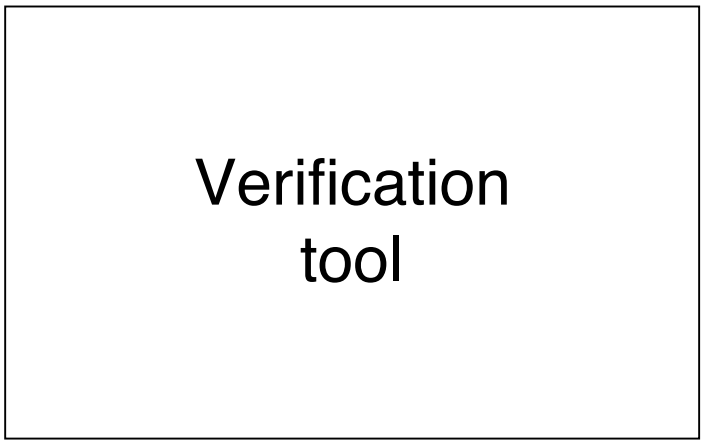
Techniques for the Verification of Java Card Programs on the Source Code Level

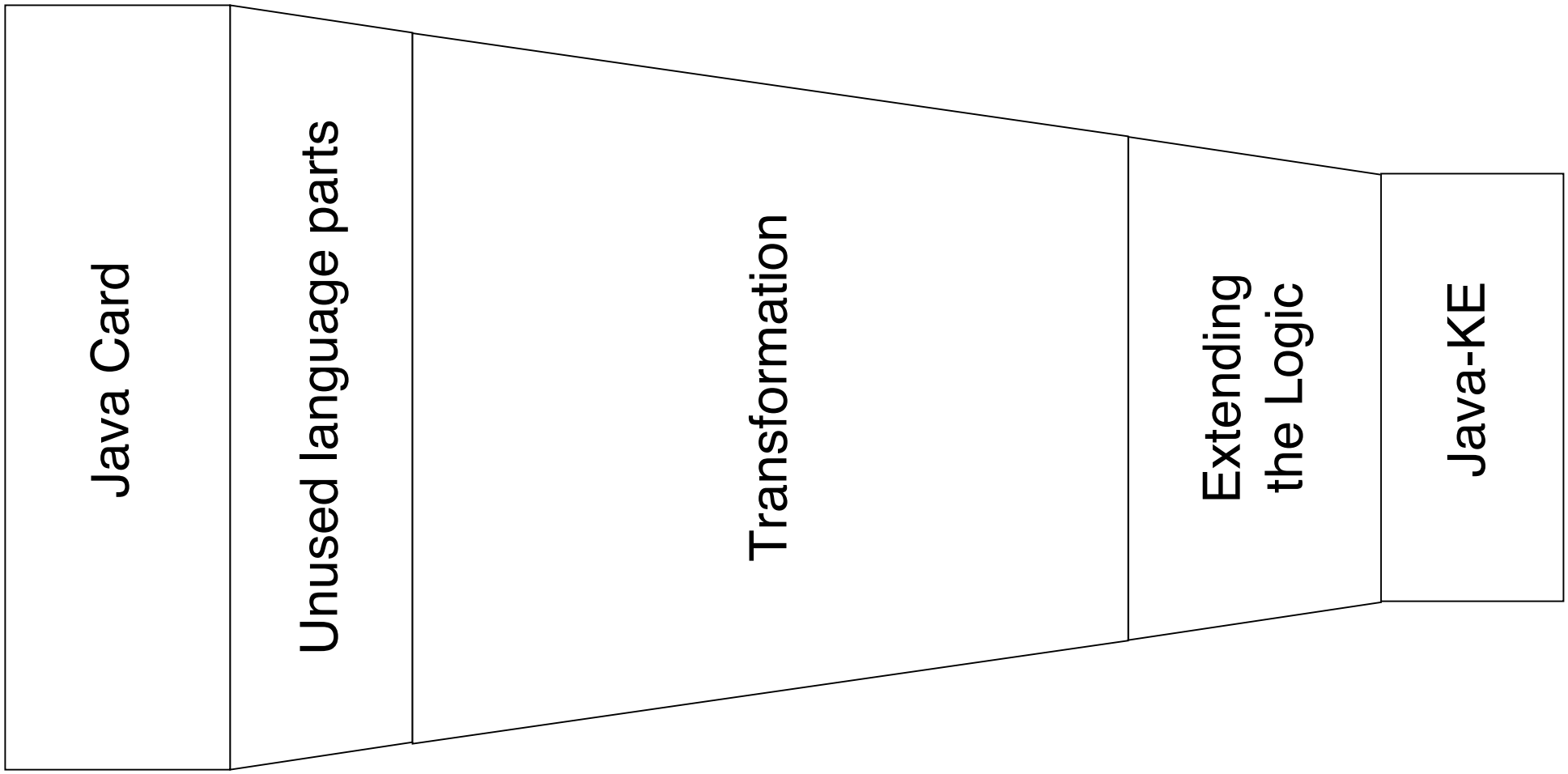
Nicole Rauch
Nicole.Rauch@Fernuni-Hagen.de
Praktische Informatik V
FernUniversität Hagen
Hagen, Germany

Java Card Code Verification

```
class A {
   //@ public invariant ...

   /*@ public behavior
     @   requires ...
     @   assignable ...
     @   ensures ...
     @   signals ...
     @*/
   void m() {
      // ...
   }
}
```



„Source code satisfies specification“


Java Card
Unused language parts
Transformation
Extending the Logic
Java-KE

The Abstract Grammars

Java Card

```
Statement    =
   If_then_else_stmt | While_stmt
   | Block | Try_stmt
   | Expression

   Break_stmt | Continue_stmt |
   Return_stmt | Throw_stmt |
   For_stmt | Do_stmt | Switch_stmt
   | Empty
Expression  =
   Field_access | Assignment |
   Cast_expr |
   Class_instance_creation |
   Method_invocation |
   Literal |
   UsedName |
   Unary_expr | Binary_expr |

   Conditional_expr | This |
   Class_or_interface_type |
   Array_creation_expr |
   Array_access
```

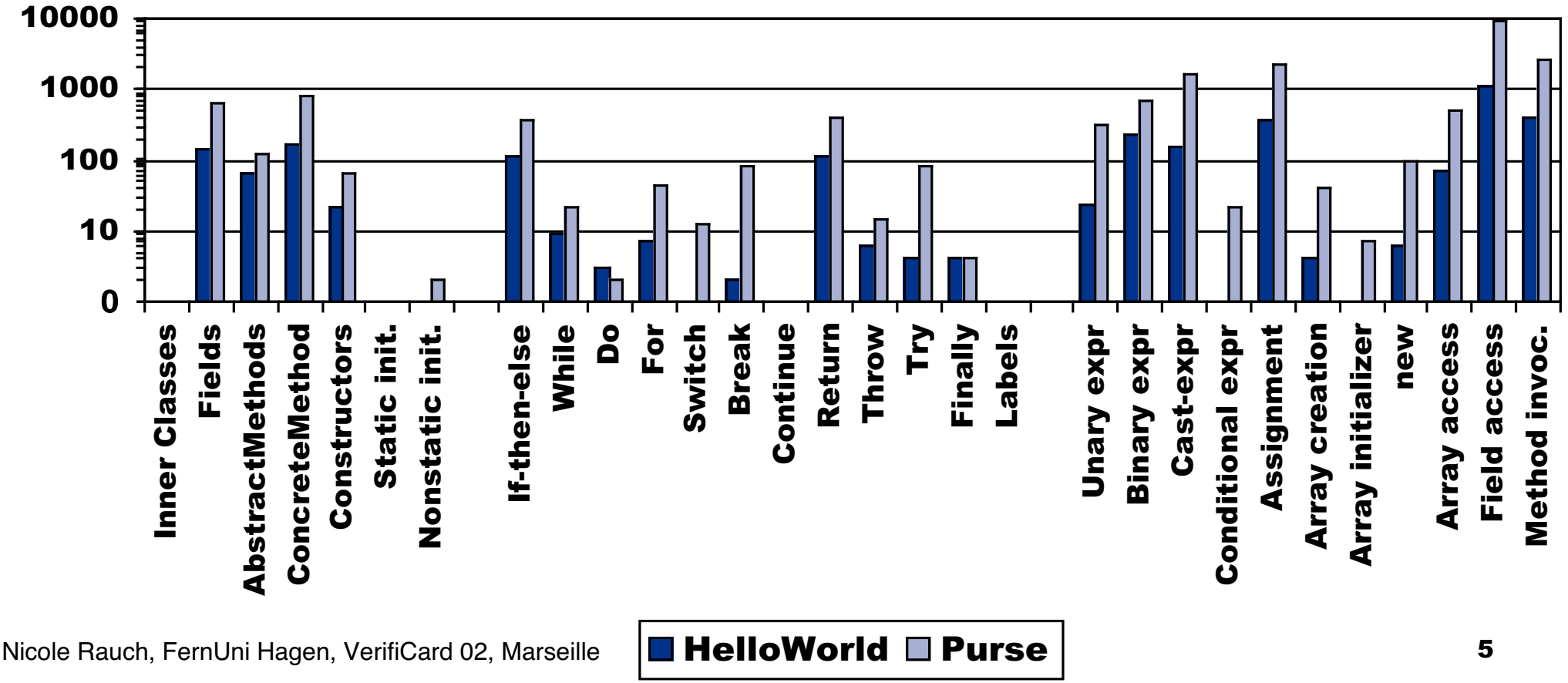
Java-KE

```
Statement  =
   if | while
   | seq | block | catch
   | fread | fwrite | cassign | new
   | invoc | call



Exp        =


   ic | bc | null |
   id |
   unary | binary
```

Amount of Syntactical Elements

HelloWorld:
25 Class declarations
3 Interface declarations
1344 lines of code

Purse:
81 Class declarations
14 Interface declarations
> 10000 lines of code



Unused parts of Java Card

- Inner classes
- Labels

Extending the Logic

- Arrays
- `break`-statements
- `return`-statements
- `throw`-statements

Source Code Transformation

- `for`-statement → `while`-statement
- `do`-statement → `while`-statement
- Array initializer → list of assignments
- constructors, nonstatic initializers → methods

Further Work

- Correctness of transformations
- Modelling `break`, `return` and `throw` in the logic
- Adding arrays to the logic