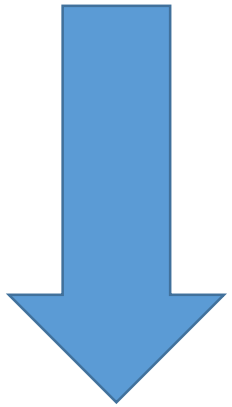
Анализ программ

Как понять, что ты – хороший программист.
И программы у тебя – хорошие.

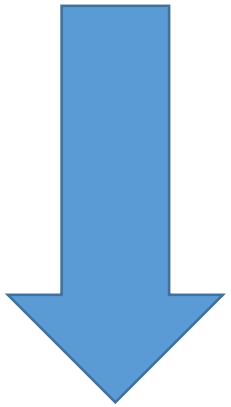
Алексей Кудрявцев
Alexey.Kudravtsev@jetbrains.com

Анализ программ

Хороша ль софтварь моя родная?
Много ль в ней еррорс, факапс и багз?



Динамический анализ



Статический анализ

Динамический анализ: Наивняк

Instant feedback

Примеры: Bret Victor: Inventing On Principle, Light Table

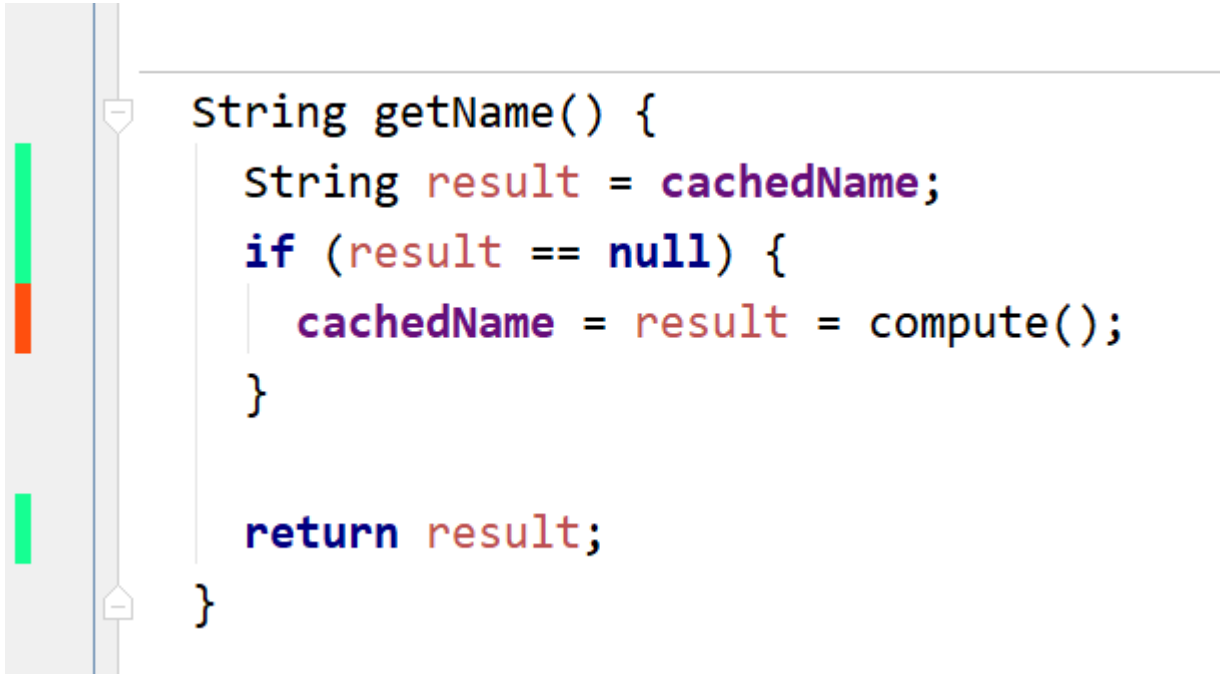
```
// tree
//

function drawTree () {
    var blossomPoints = [];

    resetRandom();
    drawBranches(0, -Math.PI/2, canvasWidth/2, canvasHeight, 30,

    resetRandom();
    drawBlossoms(blossomPoints);
}

function drawBranches (i,angle,x,y,width,blossomPoints) {
    ctx.save();

    var length = tween(i, 1, 60, 12, 3) * random(0.7, 1.3);
    if (i == 0) { length = 97; }

    ctx.translate(x,y);
    ctx.rotate(angle);
    ctx.fillStyle = "#000";
    ctx.fillRect(0, -width/2, length, width);

    ctx.restore();

    var tipX = x + (length - width/2) * Math.cos(angle);
    var tipY = y + (length - width/2) * Math.sin(angle);

    if (i > 4) {
        blossomPoints.push([x,y,tipX,tipY]);
    }

    if (i < 6) {
        drawBranches(i + 1, angle + random(-0.15, -0.05) * Math.
        drawBranches(i + 1, angle + random( 0.15,  0.05) * Math.
    }
    else if (i < 12) {
        drawBranches(i + 1, angle + random( 0.25, -0.05) * Math.
```

Динамический анализ: Элементарные частицы

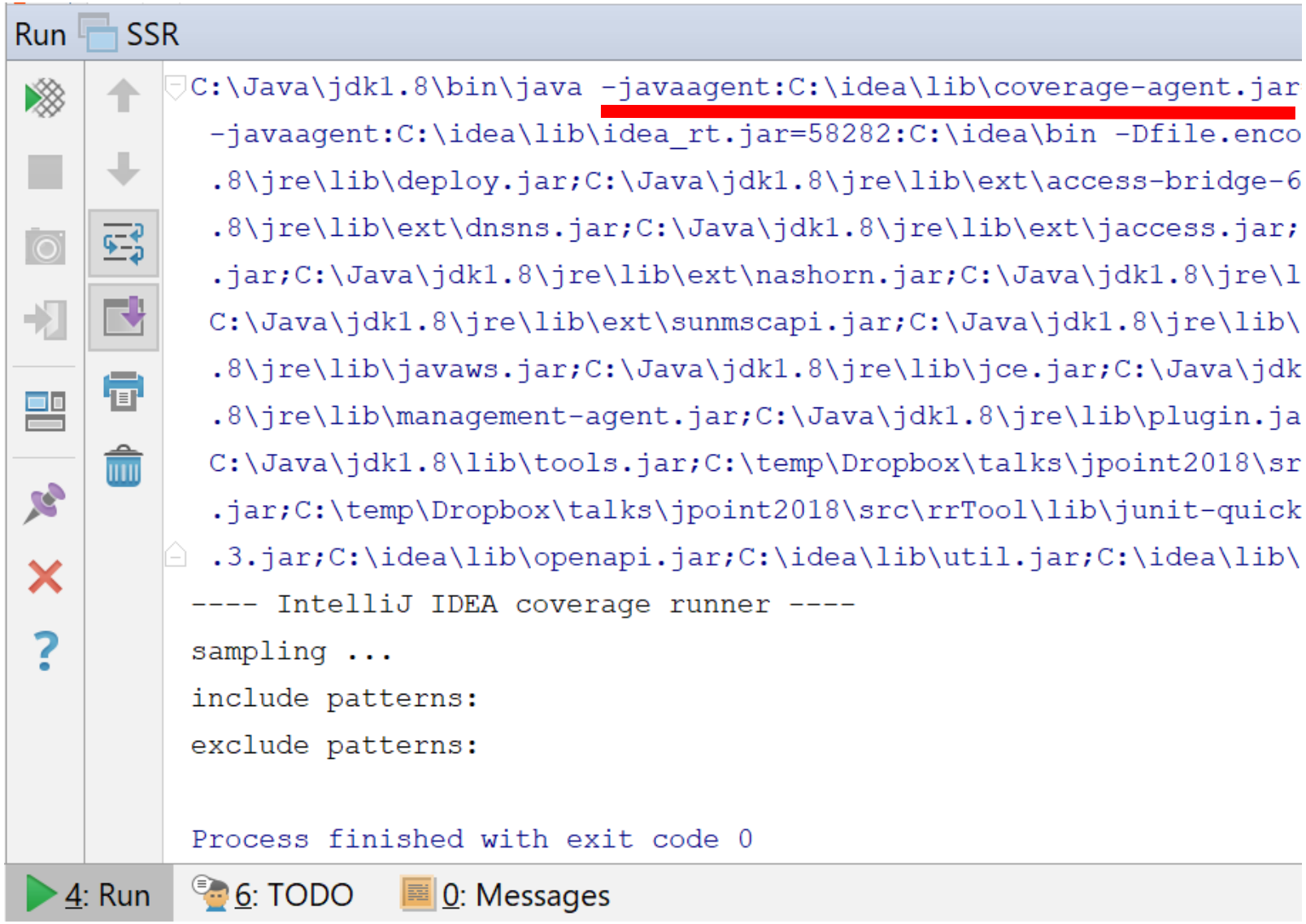
Detectors.

Примеры: Lin-Check data race detector, Road Runner framework, Coverage

Динамический анализ: Coverage



Динамический анализ: Coverage



Динамический анализ: RoadRunner detector

ANALYSIS	ALGORITHM
THREADLOCAL	Filters accesses to thread-local data
READONLY	Filters accesses to read-only data
PROTECTINGLOCK	Filters lock operations for locks protected by other locks
LOCKSET	Detects races using the LockSet algorithm
ERASERWITHBARRIER	Detects races using LockSet + a barrier analysis
HAPPENSBEFORE	Detects races using VectorClocks
DJIT+	Detects races using an optimized VectorClock algorithm
MULTIRACE	Detects races using a hybrid LockSet/VectorClock analysis
GOLDILOCKS	Detects races using an extended LockSet algorithm
FASTTRACK	Detects races using an Epoch/VectorClock analysis
VELODROME	Detects serializability errors
ATOMIZER	Detects atomicity violations using Lipton’s theory of reduction
SINGLETRACK	Detects determinism errors
JUMBLE	Adversarial memory implementation to utilize relaxed memory model nondeterminism
SIDETRACK	Detects generalized serializability errors

Динамический анализ: RoadRunner detector

```
public class AppRace implements Runnable {
  private int value;
  public void run() {
    for (int i=0; i<1000; i++) {
      value++;
    }
  }
  public static void main(String[] args) throws Exception {
    AppRace d = new AppRace();
    Thread t1 = new Thread(d);   t1.start();
    Thread t2 = new Thread(d);   t2.start();
    t1.join();   t2.join();
  }
}
```

Динамический анализ: RoadRunner detector

```
## YIKES: SafeFieldUpdater: Concurrent update.
## YIKES: concurrent initialization of guard state

## ============================================================================
## FastTrack Error
##
##          Thread: 2
##           Blame: x/Application.value_I
##           Count: 1    (max: 100)
##    Shadow State: [W=(1:3) R=(1:3) V=[]]
##  Current Thread: [tid=2    C=[(0:3) (1:0) (2:3) (3:0)]   E=(2:3)]
##           Class: class x.Application
##           Field: @92 x/Application.value_I
##         Message: Write-Read Race
##     Previous Op: Write by  null
##     Current Op: Read by  Thread-1[tid = 2]
##           Stack: tools.fasttrack_long.FastTrackTool.fieldError(FastTrackTool.java:753)
##                  tools.fasttrack_long.FastTrackTool.error(FastTrackTool.java:707)
##                  tools.fasttrack_long.FastTrackTool.read(FastTrackTool.java:371)
##                  tools.fasttrack_long.FastTrackTool.access(FastTrackTool.java:308)
##                  rr.tool.RREventGenerator.readAccess(RREventGenerator.java:146)
##                  x.Application.__$rr_get_value(Application.java)
##                  x.Application.__$rr_run_$rr_Original(Application.java:23)
##                  x.Application.run(Application.java:21)
##                  java.lang.Thread.run(Thread.java:748)
## ============================================================================
```

RoadRunner detector. А мне че?

```
@Abbrev("UNT")
public class UnnamedThread extends Tool {
  @Override
  public void postStart(StartEvent se) {
    String justCreatedThreadName =
se.getNewThread().getThread().getName();
    if (  justCreatedThreadName.startsWith("X
       || justCreatedThreadName.startsWith("Ж
      errors.error(se.getNewThread(), se.getI
          "reason:",
          “Bad thread name");
    }
  }
}
```

RoadRunner detector. А мне че?

```
public abstract class Tool  {...
  public void create(NewThreadEvent e) {...}...
  public void stop(ShadowThread td) {...}...
  public void access(AccessEvent fae) {...}...
  public void volatileAccess(VolatileAccessEvent fae) {...}...
  public void enter(MethodEvent me) {...}...
  public void exit(MethodEvent me) {...}...
  public void acquire(AcquireEvent ae) {...}...
  public void release(ReleaseEvent re) {...}...
  public void preWait(WaitEvent we) {...}...
  public void postWait(WaitEvent we) {...}...
  public void preNotify(NotifyEvent ne) {...}...
  public void postNotify(NotifyEvent ne) {...}...
  public void preSleep(SleepEvent e) {...}...
  public void postSleep(SleepEvent e) {...}...
  public void preJoin(JoinEvent je) {...}...
  public void postJoin(JoinEvent je) {...}...
  public void preStart(StartEvent se) {...}...
  public void postStart(StartEvent se) {...}...
  public void preInterrupt(InterruptEvent me) {...}...
  public void interrupted(InterruptedEvent e) {...}...
  public void classInitialized(ClassInitializedEvent e) {...}...
  public void classAccessed(ClassAccessedEvent e) {...}...
}
```

Динамический анализ: Больше ума

Property-based testing

Примеры: ScalaCheck, QuickCheck

Динамический анализ: Property-based testing

```
Build IntelliJ Platform Products :: IJ Platform - Master :: IDEA Trunk :: Tests :: Exploratory Tests #5065 failed

Changes included: 2 changes.
Change f98cbb8d by tagir.valeev (3 files): MismatchedCollectionQueryUpdate: support constructors with
Change 0621ee18 by tagir.valeev (1 file): java.lang.Math methods annotated as pure

Failed tests summary: 1 (1 new)
(new) com.intellij.java.propertyBased.JavaCodeInsightSanityTest.testRandomActivity

org.jetbrains.jetCheck.PropertyFalsified: Falsified on DependencyVisitorTest.java[
  InvokeCompletion{DependencyVisitorTest.java, offset=2165, selected 'StringBufferInputStream' with '\n'}
]
Minimized in 7 stages, by trying 16 examples
To reproduce the last iteration, run PropertyChecker.forAll(...).rechecking(-2168976476482686776L, 72).shouldHold(...)
Global seed: -1863187936284897163L
Property failure reason: java.lang.AssertionError: import java.io.StringBufferInputStream;class Dummy
  at com.intellij.psi.impl.source.tree.injected.DocumentW.calculateMinEditSequence(DocumentWindowImpl.java:788)
  at com.intellij.psi.impl.source.tree.injected.InjectedF.rootChanged(InjectedFileViewProvider.java:61)
  at com.intellij.psi.impl.source.PsiFileImpl.subtreeChanged(PsiFileImpl.java:376)
  at com.intellij.psi.impl.source.PsiJavaFileBaseImpl.subtreeChanged(PsiJavaFileBaseImpl.java:67)
  at com.intellij.psi.impl.source.tree.CompositeElement.subtreeChanged(CompositeElement.java:91)
  at com.intellij.psi.impl.source.tree.CompositeElement.rawAddChildren(CompositeElement.java:774)
```

Property-based testing. А как же я?

```
private String generateComplexHtmlPreview
                    (String title, String text) {
  return title + text;
}

void example() {
  generateComplexHtmlPreview("Фе", "Гня");
}
```

Property-based testing. А как же я?

```
@RunWith(JUnitQuickcheck.class)
public class SpellCheck {
  @Property
  public void noSwear(String text,
                      String title) {
    String generated =
        preview(text, title);
    assertFalse(generated.contains("Ж
  }
}
```

Property-based testing. Ключевые слова

- Generators
- Constraints
- Shrinking

```
@GeneratorConfiguration
@interface SwearWordParts {}

static class SwearGenerator extends Generator<String> {
  public String generate(SourceOfRandomness random,
                         GenerationStatus status) {
    int n = random.nextInt();
    return swearWordPart(n);
  }
  public void configure(SwearWordParts anno) {
  }
}

@RunWith(JUnitQuickcheck.class)
public class RunCheck {
  @Property
  public void noSwear
         (@SwearWordParts String text,
          @SwearWordParts String title) {
    assertFalse(preview(title, text).contains("Ж"));
  }
}
```

Property-based testing. Ключевые слова

- Generators
- Constraints
- Shrinking

```
@RunWith(JUnitQuickcheck.class)
public class RunCheck2 {
  @Property
  public void noSwear(String text,
       @InRange(minInt = 1900,
                maxInt = 1999) Year year) {
    assumeThat(text.length(), greaterThan(4));
    assertFalse(preview(year.toString(), text).contains ("
  }
```

Property-based testing. Ключевые слова

- Generators
- Constraints
- Shrinking

```
@Override
public List<String>
doShrink(SourceOfRandomness r, String w) {
  return Arrays.asList(
      w.substring(0, w.length() / 2),
      w.substring(w.length() / 2),
      w.substring(0, w.length() / 3),
      w.substring(2 * w.length() / 3));
}
```

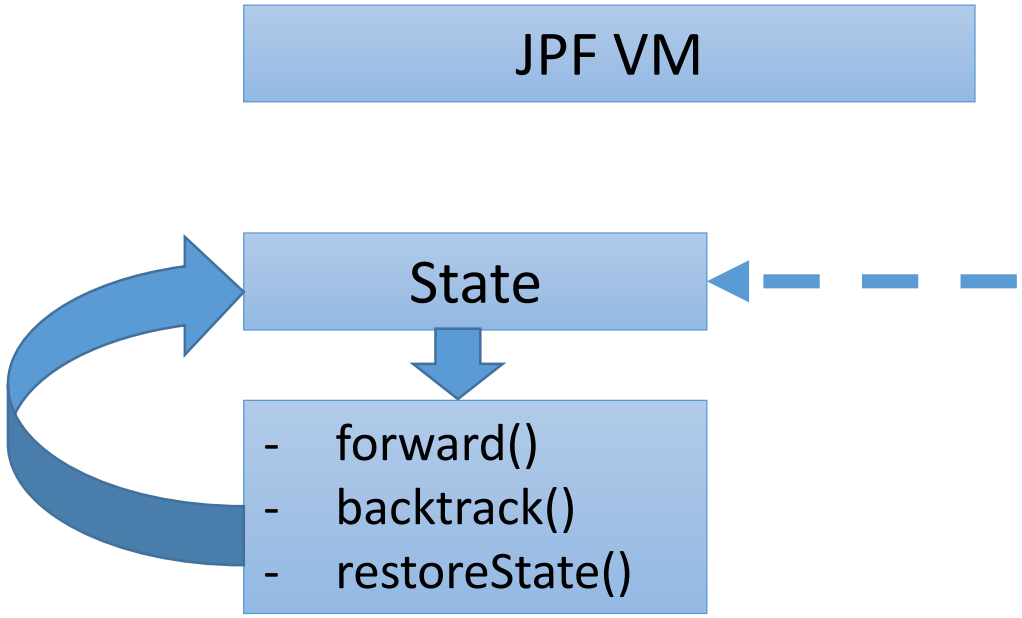
Динамический анализ: Яйцеголовость

Model checking

Пример: JPF

Динамический анализ: Java Path Finder

java gov.nasa.jpf.JPF MyTestClass



Projects/Listeners

- jpf-actor - Tool and framework for systematic testing of actor programs (e.g. Scala)
- jpf-aprop - Java annotation based properties and their corresponding checkers
- jpf-awt - JPF specific library implementations for `java.awt` and `javax.swing`
- jpf-awt-shell - specialized JPF shell for model checking `java.awt` and `javax.swing` applications
- jpf-concurrent - optimized `java.util.concurrent` library implementation for JPF
- jpf-delayed - postpones non-deterministic choice of values until they are used
- jpf-guided-test - Framework for guiding the search using heuristics and static analysis
- jpf-mango - specification and proof artifact generation
- jpf-numeric - an alternative bytecode set for inspection of numeric programs
- jpf-racefinder - a precise data race detector in a relaxed Java memory model
- jpf-rtembed - programs for real-time and embedded platforms (e.g., RTSJ and SCJ)
- jpf-statechart - UML statechart modeling
- net-iocache - I/O cache extension to handle network communication
- jpf-cv - compositional verification using JPF
- jpf-nas - model checking of distributed multithreaded programs

Динамический анализ: Java Path Finder

```
boolean workTime(int hour, int minute) {
  if (hour >= 20) return false;
  if (hour < 8) return false;
  if (hour == 14) { // обед
    return minute > 30;
  }
  return minute > 5; // зарядка каждый час
}
```

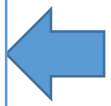
Динамический анализ: Java Path Finder

```
boolean workTime(int hour, int minute) {
  if (hour >= 20) return false;
  if (hour < 8) return false;
  if (hour == 14) { // обед
    return minute > 30;
  }
  return minute > 5; // зарядка каждый час
}
```

Symbolic executor:
- Symbol(hour)
- Symbol(minute)

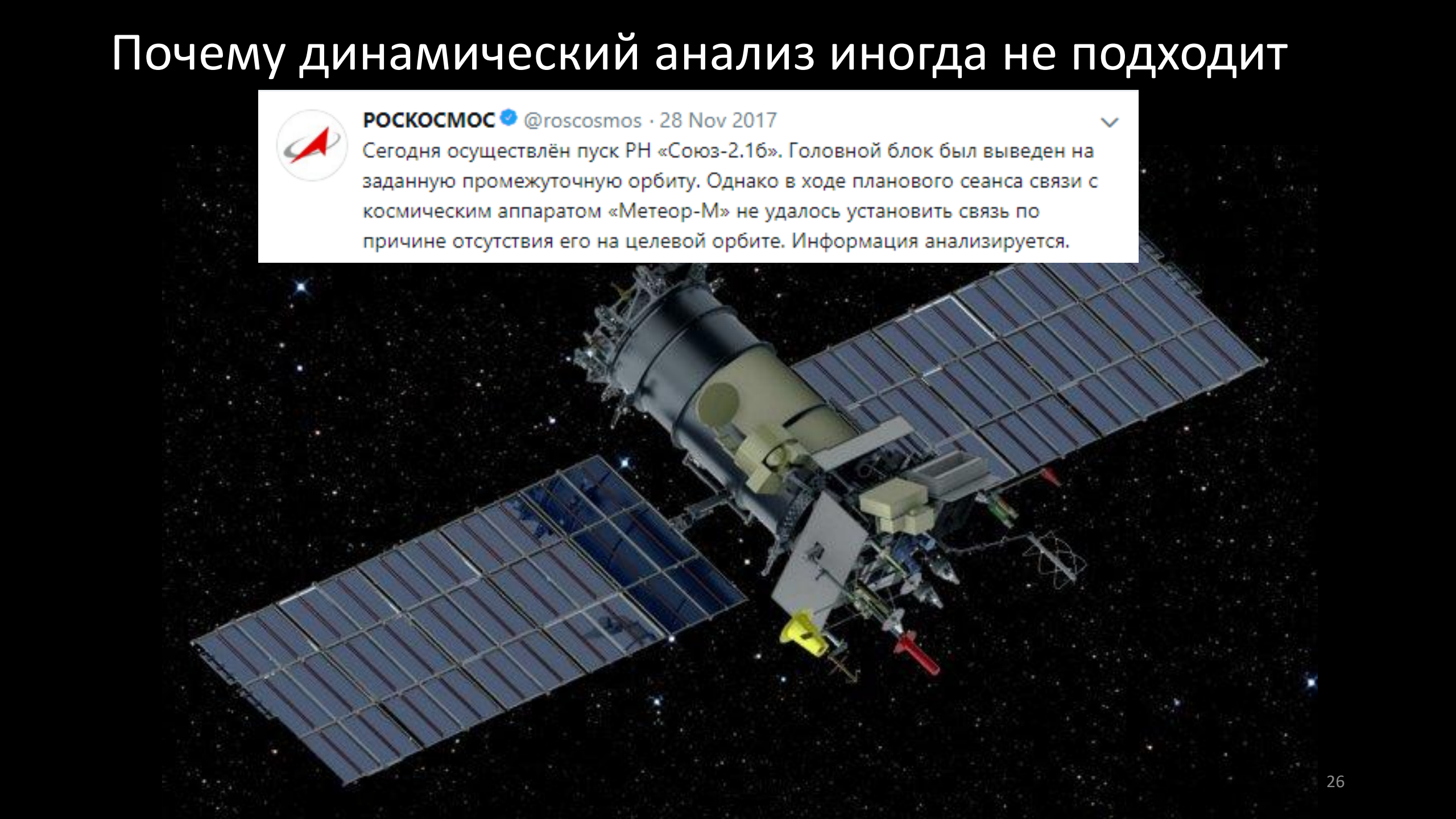
Constraints:
- Hour >= 20
- Hour=14 Λ minute <= 30
- …

Constraint solver:
- Hour >= 20
- Hour=14 Λ minute=(-∞..30]
- …



Test cases:
- workTime(20, 0)
- worktime(14, 30)
- …

Почему динамический анализ иногда не подходит



Статический анализ

- Без запуска программы!
- Без специфического железа!
- Без окружения!

Статический анализ обречен

Остановится или зависнет?

```
public class CollatzConjecture {
  public static void test(int n) {
    while (n != 1) {
      n = n % 2 == 0 ? n / 2 : 3 * n + 1;
    }
  }
  public static void main(String[] args) {
    for (int i = 0; ; i++) test(i);
  }
}
```

Статический анализ обречен: проблема останова

```
public class H {
  static native boolean halts(String path);

  static void blowYourMind(String path) {
    if (halts(path)) {
      while (true);
    }
  }

  public static void main(String[] args) {
    blowYourMind("H.java:blowYourMind");
  }
}
```

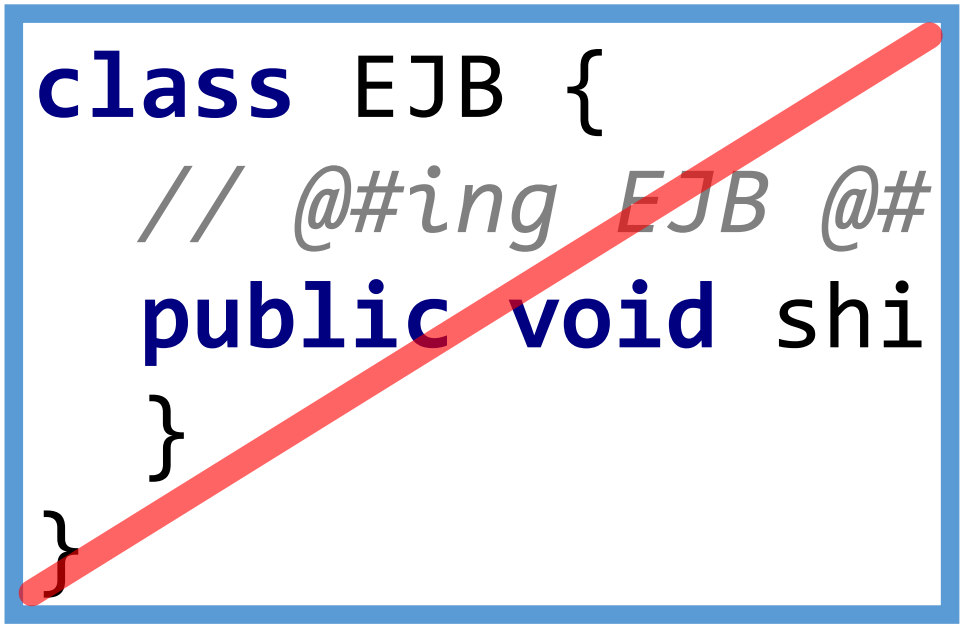
Стат.анализ ваще бесполезен: теорема Райса

```
fun isPlusOne(program: String): Boolean { … }

fun halts(program: String): Boolean {
    fun stlah(a: Int): Int {
        exec(program);
        return a + 1
    }
    return isPlusOne("Rice.kt:stlah")
}
```

Статический анализ: Ну хоть что-нибудь?

Pattern Checkers

Пример: javac plugin, Error Prone



Статический анализ: javac plugin

```
public class SwearJavacPlugin implements Plugin {
  @Override
  public void init(JavacTask task, String... args) {
    task.addTaskListener(new TaskListener() {
      @Override
      public void started(TaskEvent e) {
        if (e.getKind() == TaskEvent.Kind.GENERATE) {
          CompilationUnitTree tree = e.getCompilationUnit();
          tree.accept(new TreeScanner<Void, Void>() {
            @Override
            public Void visitMethod(MethodTree methodNode,
                                    Void aVoid) {
              if (methodNode.getName().toString().contains("Fuc
                Trees trees = Trees.instance(task);
                trees.printMessage(Diagnostic.Kind.ERROR,
                    "No swearing!", methodNode, tree);
              }
              return super.visitMethod(methodNode, aVoid);
            }
          }, null);
        }
      }
```

Статический анализ: javac plugin

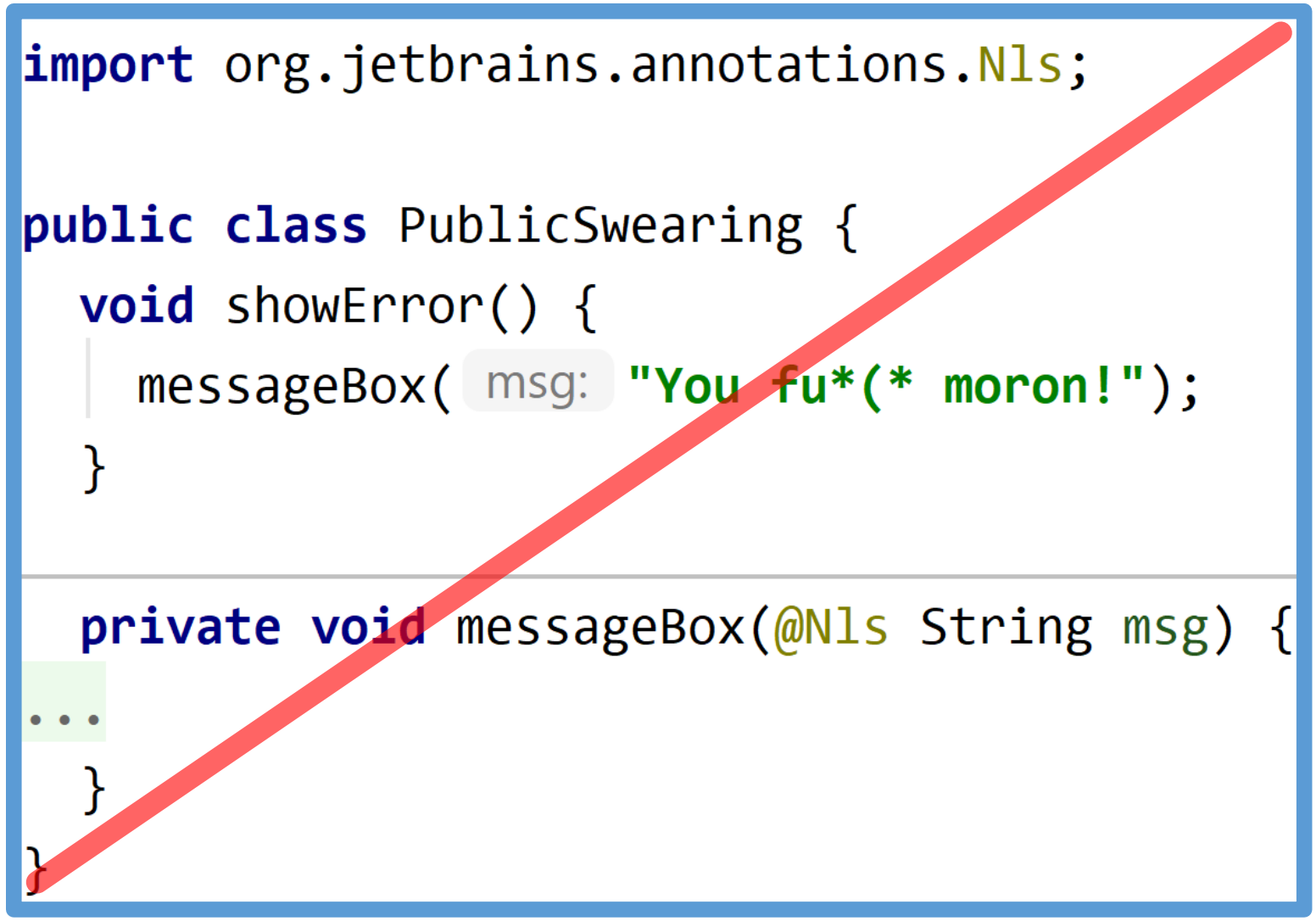
```
public class Swear {
  public void printFuchsia() {}
}
```

```
C:\rrTool>javac -processorpath production\rrTool -Xplugin:NoSwearingPlugin src\x\Swear.java
src\x\Swear.java:4: error: No swearing!
  public void printFuchsia() {}
              ^
1 error
```

Статический анализ: Удобнее паттерны

Inspections

Примеры: Intellij IDEA Inspections, FindBugs, SpotBugs



Статический анализ: Inspections

```
public class PublicSwearInspection extends LocalInspectionTool {
  @NotNull
  @Override
  public PsiElementVisitor buildVisitor(@NotNull ProblemsHolder holder, boo…
    return new JavaElementVisitor() {
      @Override
      public void visitLiteralExpression(PsiLiteralExpression expression) {
        if (isPublicArgument(expression) &&
            expression.toString().contains("Fuc")) {
          holder.registerProblem(expression, "Fuc! You swearing again");
        }
      }
    };
  }

  private boolean isPublicArgument(PsiLiteralExpression expression) {
    PsiElement parent = expression.getParent();
    if (!(parent instanceof PsiExpressionList)) return false;
```

Статический анализ: Inspections

```
public void foo(String s) {
    foo( s: "Swear");
}
```

PSI Structure

- PsiMethodCallExpression:foo("Swear")
  - PsiReferenceExpression:foo
    - PsiReferenceParameterList
    - PsiIdentifier:foo
  - PsiExpressionList
    - PsiJavaToken:LPARENTH
    - PsiLiteralExpression:"Swear"
      - PsiJavaToken:STRING_LITERAL
    - PsiJavaToken:RPARENTH
  - PsiJavaToken:SEMICOLON

Статический анализ: Inspections

```
private boolean isPublicArgument(PsiLiteralExpression expression) {
  PsiElement parent = expression.getParent();
  if (!(parent instanceof PsiExpressionList)) return false;
  PsiElement call = parent.getParent();
  if (!(call instanceof PsiCall)) return false;
  PsiMethod method = ((PsiCall) call).resolveMethod();
  if (method == null) return false;
  PsiExpression[] expressions = ((PsiExpressionList) parent).getExpressions();
  int argIndex = ArrayUtil.indexOf(expressions, expression);
  PsiParameter[] parameters = method.getParameterList().getParameters();
  if (parameters.length <= argIndex) return false;
  PsiParameter parameter = parameters[argIndex];
  return AnnotationUtil.isAnnotated(parameter, anno, true);
}
private static final String anno = "org.jetbrains.annotations.Nls";
```

Статический анализ: Inspections

```
public class PublicSwearInspection extends LocalInspectionTool {
  @NotNull
  @Override
  public PsiElementVisitor buildVisitor(@NotNull ProblemsHolder holder, boolean isOnTheFly,
    return new JavaElementVisitor() {
      @Override
      public void visitLiteralExpression(PsiLiteralExpression expression) {
        if (isPublicArgument(expression) &&
            expression.toString().contains("Fuc")) {
          holder.registerProblem(expression, descriptionTemplate: "Fuc! You swearing again");
        }
      }
    };
  }
```

Fuc! You swearing again more... (Ctrl+F1)

Статический анализ: Еще более удобные паттерны

Пример: Structural code search/replace

Статический анализ: Structural search

```
public class SSR {
  volatile String cachedName;

  String getName() {
    String result = cachedName;
    if (result == null) {
      result = cachedName = compute();
    }

    return result;
  }
```

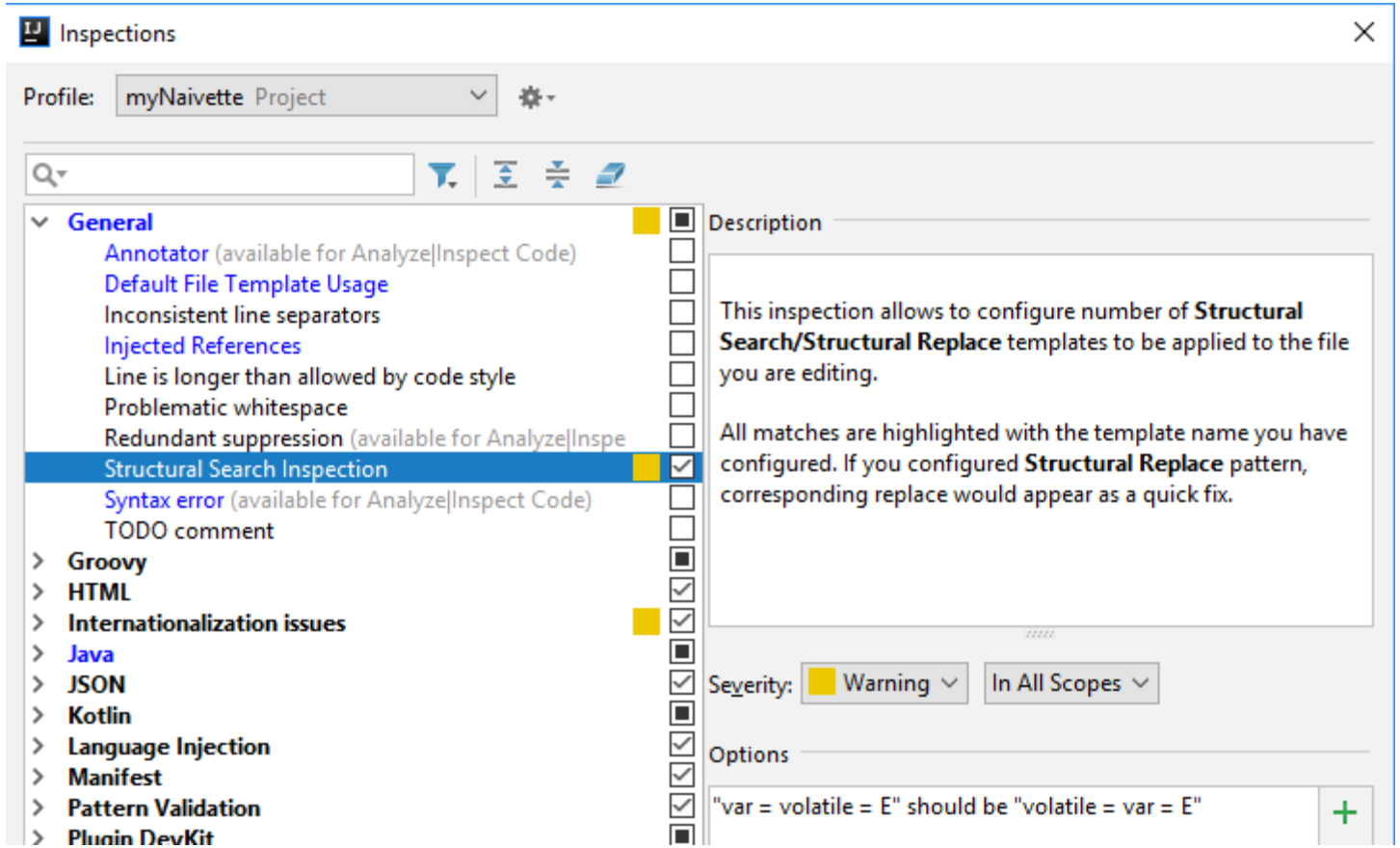
"var = volatile = E" should be "volatile = var = E" more... (Ctrl+F1)

Статический анализ: Structural search

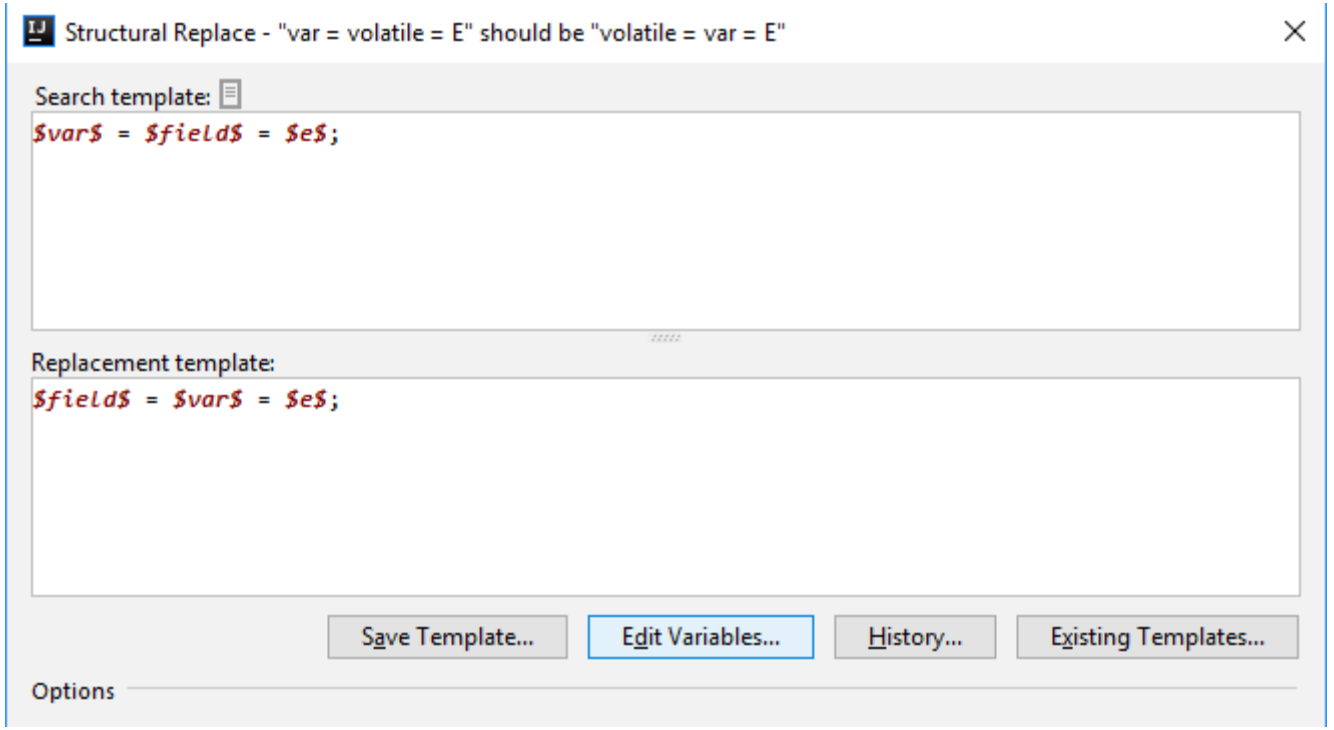
```
public class SSR {
  volatile String cachedName;

  String getName() {
    String result = cachedName;
    if (result == null) {
      result = cachedName = compute();
    }

    return result;
  }
```

Replace with 'cachedName = result = compute();'

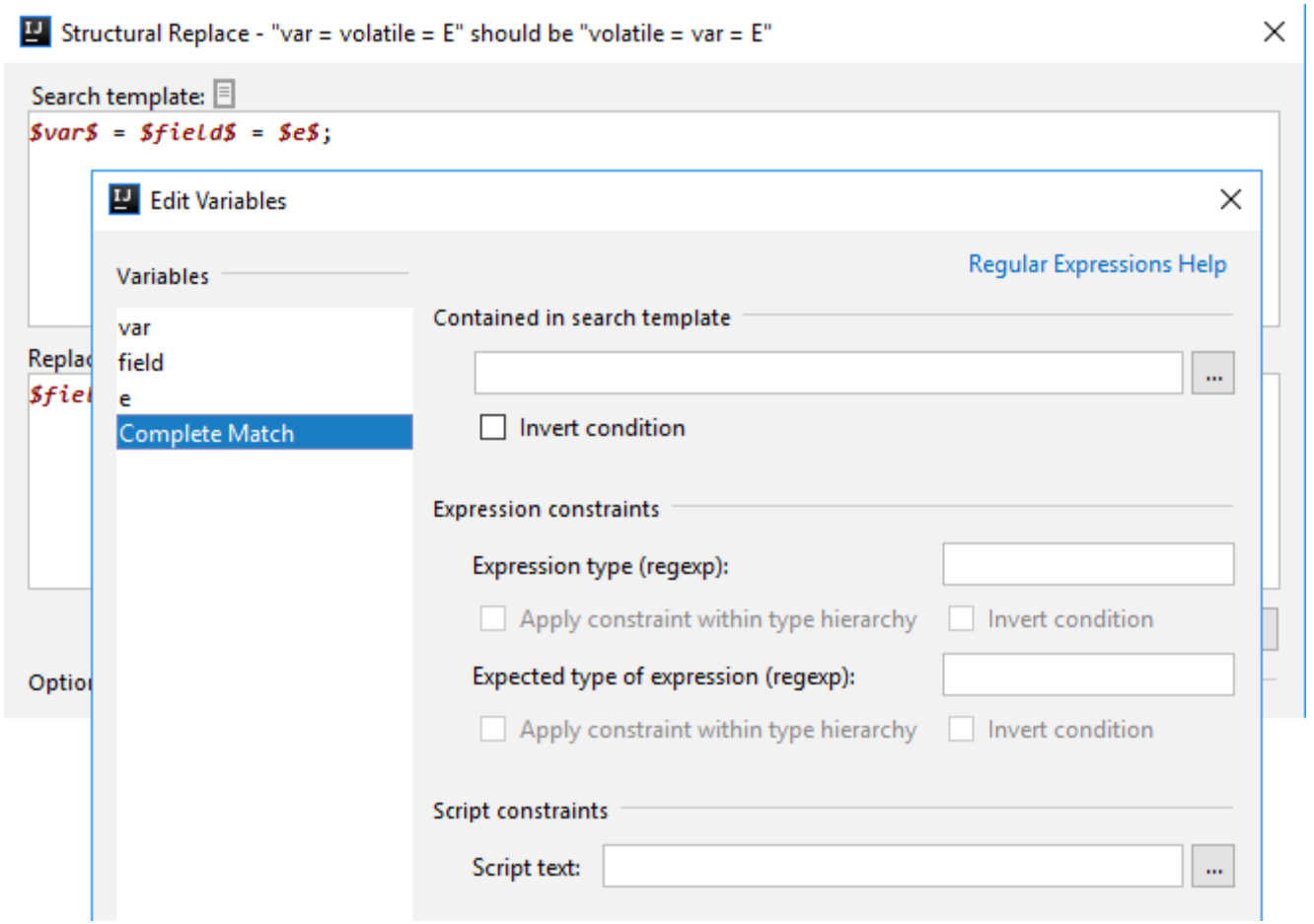
Статический анализ: Structural search



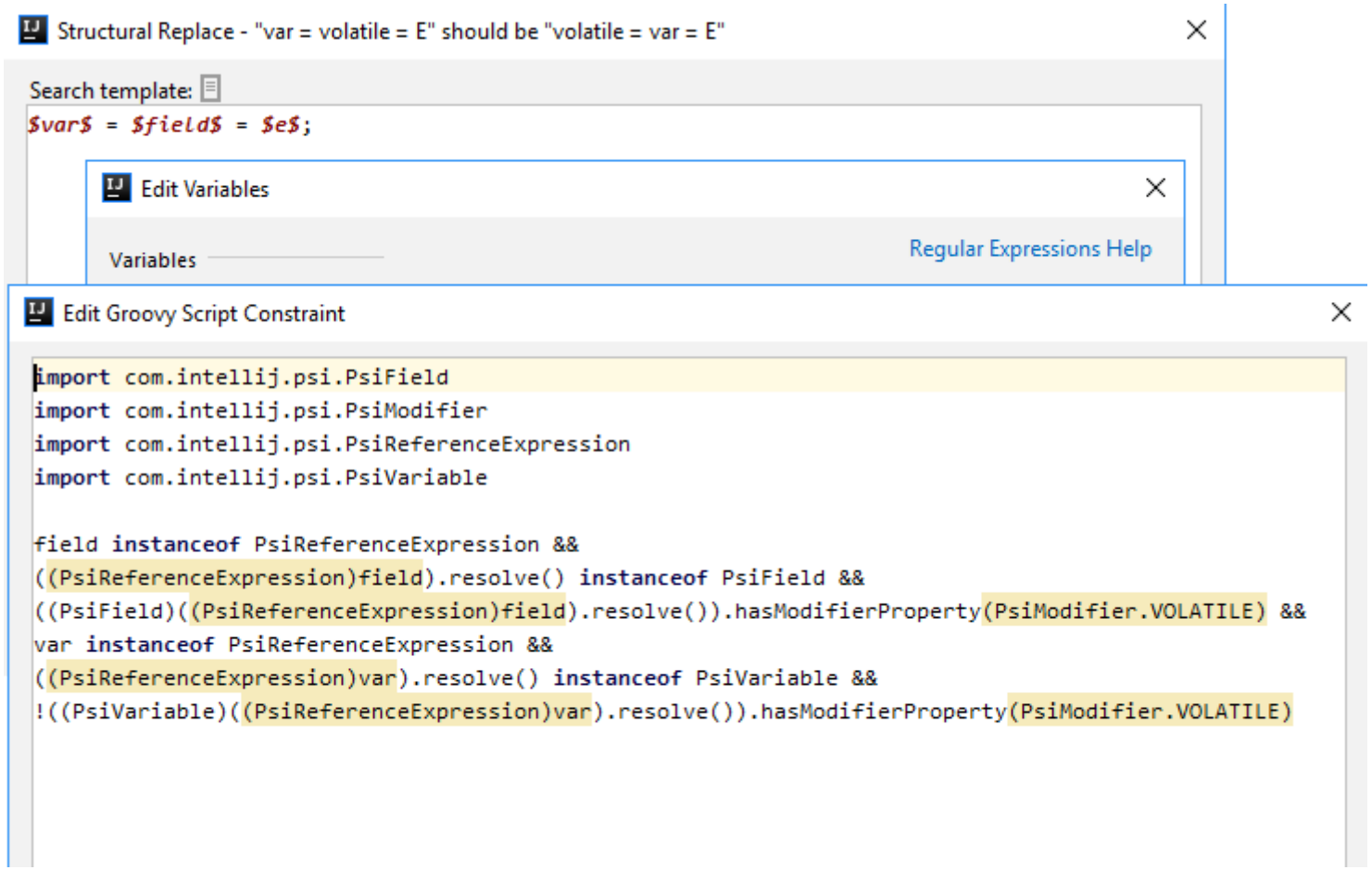
Статический анализ: Structural search



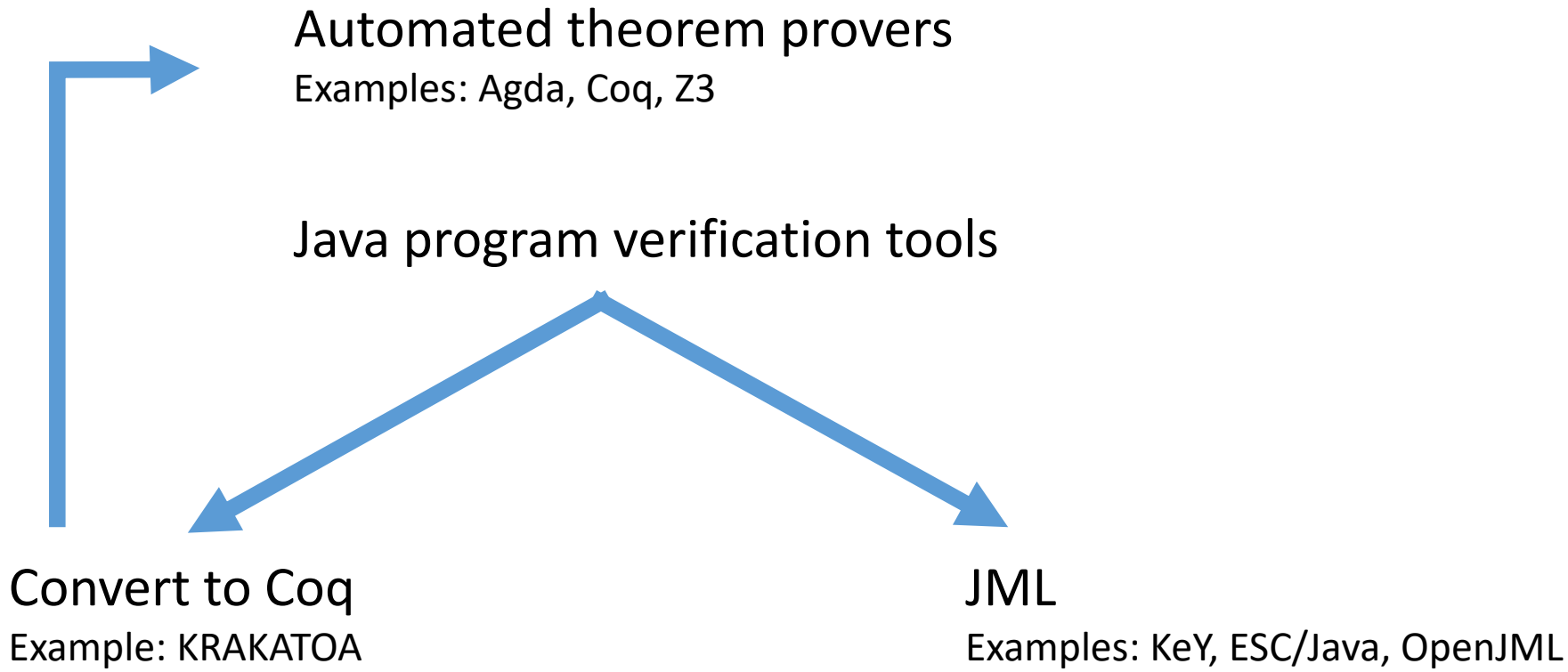
Статический анализ: Structural search



Статический анализ: Structural search



Автоматическая верификация: Чиста чтоб поржать



Автоматическая верификация: Coq

```
Inductive natlist : Type :=
  | nil : natlist
  | cons : nat → natlist → natlist.

Notation "x :: l" := (cons x l)
(at level 60, right associativity).

Fixpoint append (l1 l2 : natlist) : natlist :=
  match l1 with
  | nil ⇒ l2
  | h :: t ⇒ h :: (append t l2)
  end.

Notation "x ++ y" := (append x y)
                     (right associativity, at level 60).
```

Coq: пример доказательства

```
Theorem append_assoc : ∀l1 l2 l3 : natlist,
  (l1 ++ l2) ++ l3 = l1 ++ (l2 ++ l3).
Proof.
  intros l1 l2 l3. induction l1 as [| n l1'].
  Case "l1 = nil".
    reflexivity.
  Case "l1 = cons n l1'".
    simpl. rewrite → IHl1'. reflexivity.
Qed.
```

Coq: объяснение доказательства

```
Theorem append_assoc : ∀l1 l2 l3 : natlist,   (l1 ++ l2) ++ l3 = l1 ++ (l2 ++ l3).
Proof.  intros l1 l2 l3. induction l1 as [| n l1'].
  Case "l1 = nil".    reflexivity.
  Case "l1 = cons n l1'". simpl. rewrite → IHl1'. reflexivity. Qed.
```

Теорема: Для всех `l1`, `l2`, `l3` выполняется `(l1 ++ l2) ++ l3 = l1 ++ (l2 ++ l3)`.

Доказательство: Индукция по `l1`.

•База, `l1 = []`. Нужно показать

```
  ([] ++ l2) ++ l3 = [] ++ (l2 ++ l3),
```

что прямо следует из определения `++`.

•Шаг индукции: если `l1 = n::l1'`, то

```
  (l1' ++ l2) ++ l3 = l1' ++ (l2 ++ l3)
```

(по индукционному предположению). Осталось показать

```
  ((n :: l1') ++ l2) ++ l3 = (n :: l1') ++ (l2 ++ l3).
```

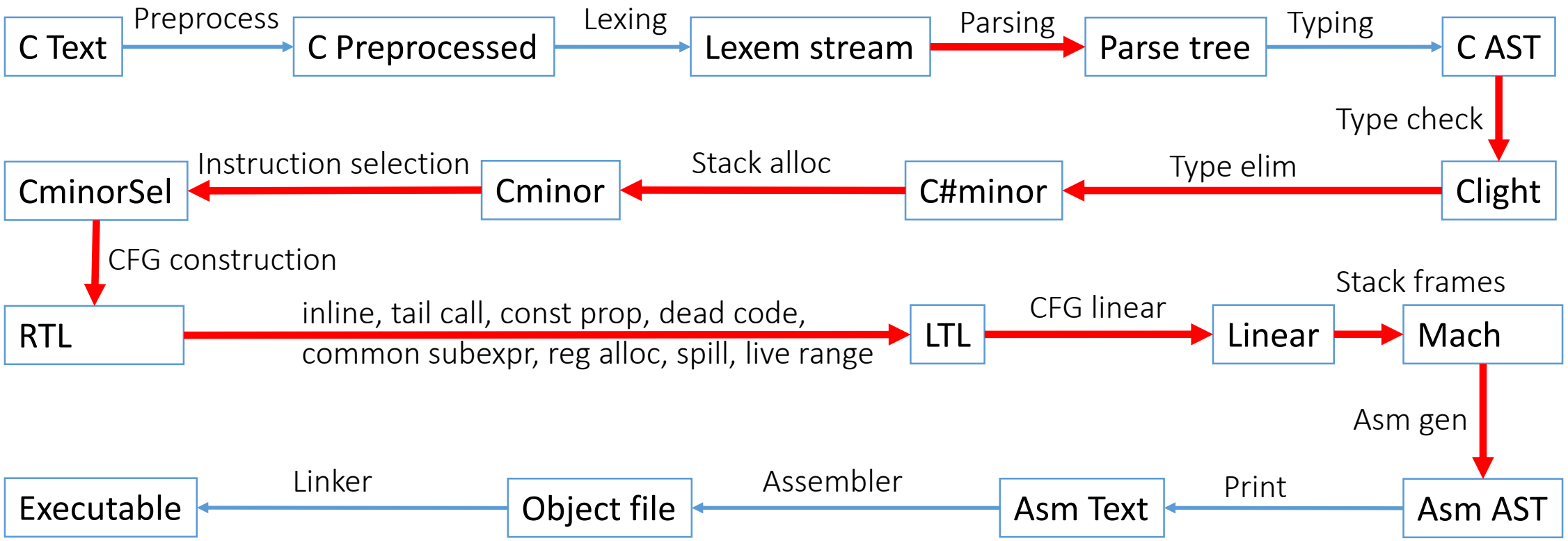
По определению `++`, это следует из

```
  n :: ((l1' ++ l2) ++ l3) = n :: (l1' ++ (l2 ++ l3)),
```

Что непосредственно вытекает из предположения индукции.

Coq & CompCert: впечатляет

CompCert C: a trustworthy compiler



Coq: как это выглядит в поле

CoqIde

File Edit View Navigation Try Tactics Templates Queries Tools Compile Windows Help

hilb.v | Compiler.v | Coqlib.v | compcert.v

```
Theorem c_semantic_preservation:
  forall p tp,
  match_prog p tp ->
  backward_simulation (Csem.semantics p) (Asm.semantics tp).
Proof.
  intros.
  apply compose_backward_simulation with (atomic (Cstrategy.semantics p)).
  eapply sd_traces; eapply Asm.semantics_determinate.
  apply factor_backward_simulation.
  apply Cstrategy.strategy_simulation.
  apply Csem.semantics_single_events.
  eapply ssr_well_behaved; eapply Cstrategy.semantics_strongly_receptive.
  exact (proj2 (cstrategy_semantic_preservation _ _ H)).
Qed.
```

Coq: как это выглядит глубже

```
Lemma compose_backward_simulation:
  forall L1 L2 L3,
  single_events L3 -> backward_simulation L1 L2 -> backward_simulation L2 L3 ->
  backward_simulation L1 L3.
Proof.
  intros L1 L2 L3 L3single S12 S23.
  destruct S12 as [index order match_states props].
  destruct S23 as [index' order' match_states' props'].
  apply Backward_simulation with (bb_order order order') (bb_match_states L1 L2 L3 match_states match_states');
  constructor.
- (* well founded *)
  unfold bb_order. apply wf_lex_ord. apply wf_clos_trans. eapply bsim_order_wf; eauto. eapply bsim_order_wf; eauto.
- (* initial states exist *)
  intros. exploit (bsim_initial_states_exist props); eauto. intros [s2 A].
  eapply (bsim_initial_states_exist props'); eauto.
- (* match initial states *)
  intros s1 s3 INIT1 INIT3.
  exploit (bsim_initial_states_exist props); eauto. intros [s2 INIT2].
  exploit (bsim_match_initial_states props'); eauto. intros [i2 [s2' [INIT2' M2]]].
  exploit (bsim_match_initial_states props); eauto. intros [i1 [s1' [INIT1' M1]]].
  exists (i1, i2); exists s1'; intuition auto. eapply bb_match_at; eauto.
- (* match final states *)
  intros i s1 s3 r MS SAFE FIN. inv MS.
  exploit (bsim_match_final_states props'); eauto.
    eapply star_safe; eauto. eapply bsim_safe; eauto.
  intros [s2' [A B]].
  exploit (bsim_E0_star props). eapply star_trans. eexact H0. eexact A. auto. eauto. auto.
  intros [i1' [s1' [C D]]].
  exploit (bsim_match_final_states props); eauto. eapply star_safe; eauto.
  intros [s1'' [P Q]].
  exists s1''; split; auto. eapply star_trans; eauto.
- (* progress *)
  intros i s1 s3 MS SAFE. inv MS.
  eapply (bsim_progress props'). eauto. eapply star_safe; eauto. eapply bsim_safe; eauto.
- (* simulation *)
  apply bb_simulation; auto.
- (* symbols *)
  intros. transitivity (Senv.public_symbol (symbolenv L2) id); eapply bsim_public_preserved; eauto.
Qed.
```

Coq & CompCert: как это выглядит на самом деле

- 2039 теорем
- 4067 лемм
- 176476 строк непонятной фигни

Автоверификация Java: KeY

```
public static int[] mergeArrays(int[] a1, int[] a2) {
  if (a1.length == 0) {
    return a2;
  }
  else if (a2.length == 0) {
    return a1;
  }
  else {
    int[] result = new int[a1.length + a2.length];
    copy(a1, result, 0, a1.length);
    copy(a2, result, a1.length, a2.length);

    return result;
  }
}

private static void copy(int[] array, int[] result, int out, int n) {
  for (int i = 0; i< n; i++) {
    result[i+out] = array[i];
  }
}
```

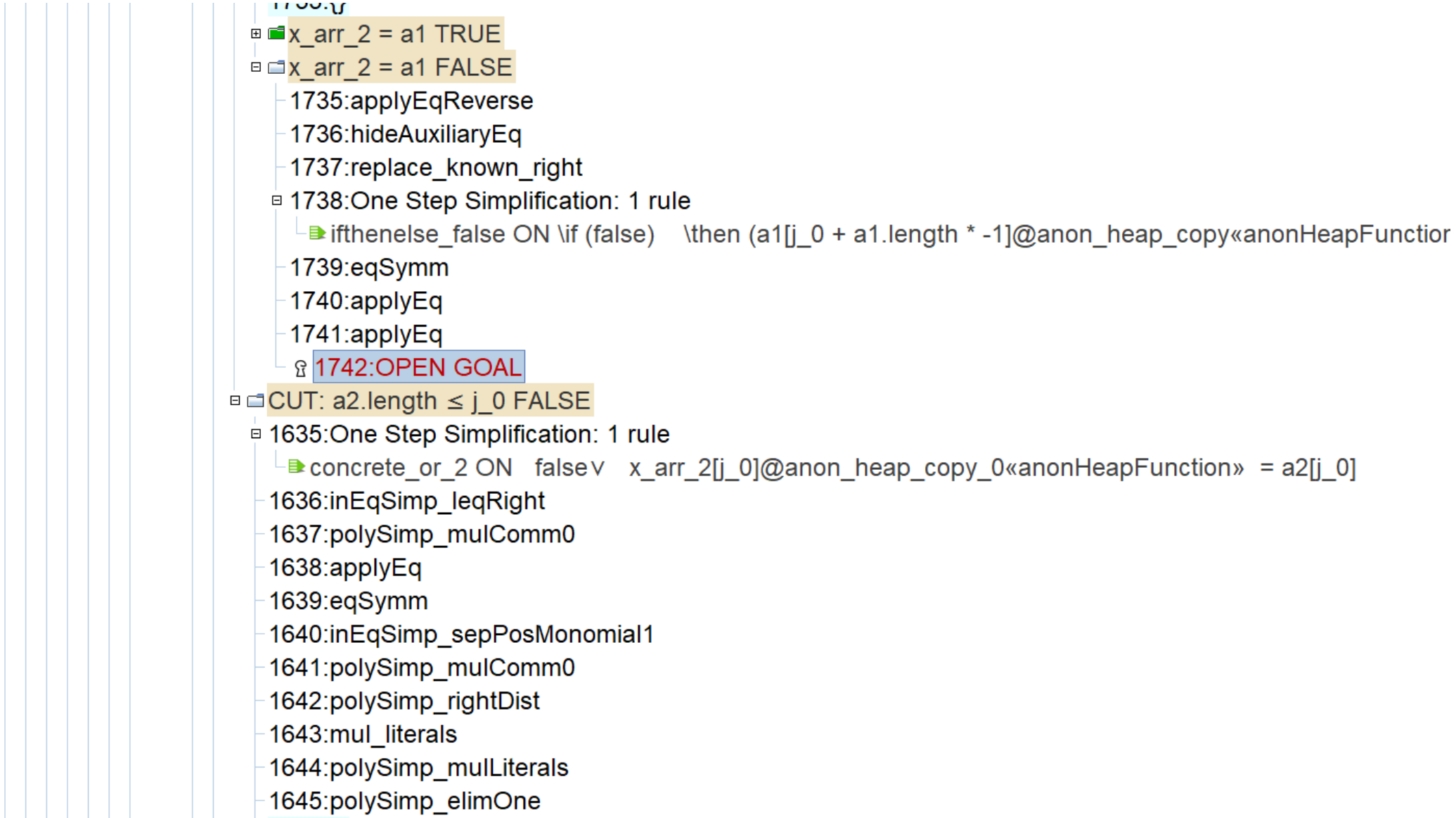
Автоверификация Java: KeY

```
/*@ normal_behaviour
  @ requires a1 != null && a2 != null;
  @ ensures (\result != null && \result.length == a1.length + a2.length);
  @ ensures (\forall int j; 0 <= j && j < a1.length; \result[j] == a1[j]);
  @ ensures (\forall int j; a1.length <= j && j < a1.length + a2.length; \result[j] == a2[j-
a1.length]);
  @ assignable \nothing;
  @*/
public static int[] mergeArrays(int[] a1, int[] a2) {
  if (a1.length == 0) {
    return a2;
  }
  else if (a2.length == 0) {
    return a1;
  }
  else {
    int[] result = new int[a1.length + a2.length];
    copy(a1, result, 0, a1.length);
    copy(a2, result, a1.length, a2.length);

    return result;
  }
```

Автоверификация Java: KeY

```
/*@ normal_behaviour
  @ requires array != null && n >=0 && n <= array.length && n <= result.length;
  @ requires array != result;
  @ requires out >= 0 && out <= result.length - n;
  @ ensures (\forall int j; 0 <= j && j < n; result[j+out] == array[j]);
  @ assignable result[out..out+n-1];
  @*/
private static void copy(int[] array, int[] result, int out, int n) {
/*@
  @ loop_invariant i>=0 && i <= n;
  @ loop_invariant (\forall int j; 0 <= j && j < i; result[j] == array[j]);
  @ decreasing n-i;
  @ assignable result[out..out+n-1];
  @*/
  for (int i = 0; i< n; i++) {
    result[i+out] = array[i];
  }
}
```

KeY: головой ап стену



```
a1.length ≤ j_0,
a2.length ≥ 1 + j_0 + a1.length * -1,
x_arr_2.length = a1.length + a2.length,
int[]::exactInstance(x_arr_2) = TRUE,
a2.length ≥ 1,
j_0 ≥ 1,
a1.length ≥ 1,
a1.length ≥ j_0 * -1,
a2.length ≥ a1.length * -1,
wellFormed(heap),
a1.<created> = TRUE,
a2.<created> = TRUE,
measuredByEmpty,
wellFormed(anon_heap_copy«anonHeapFunction»),
  heap[create(x_arr_2)]
      [x_arr_2.<transient> := 0]
      [x_arr_2.<transactionConditionallyUpdated> := FALSE]
      [x_arr_2.<initialized> := FALSE]
      [memset(arrayRange(x_arr_2, 0, -1 + a1.length + a2.length), 0)]
      [x_arr_2.<initialized> := TRUE]
      [anon(arrayRange(x_arr_2, 0, -1 + a1.length), anon_heap_copy«anonHeapFunction»)]
= heapAfter_copy,
exc_0 = null,
a2[j_0 + a1.length * -1] = a1[j_0 + a1.length * -1],
∀ int j; (j ≤ -1 ∨ j ≥ a1.length ∨ x_arr_2[j]@heapAfter_copy = a1[j]@heapAfter_copy),
wellFormed(anon_heap_copy_0«anonHeapFunction»),
heapAfter_copy[anon(arrayRange(x_arr_2, a1.length, -1 + a1.length + a2.length), anon_heap_copy_0«anonHeapFunction»)] = heapAfter_copy_0,
exc_1 = null,
a2.length ≥ 1 + j_0 + a1.length * -2,
a1.length * 2 ≥ 1 + j_0,
x_arr_2[j_0 + a1.length * -1]@anon_heap_copy«anonHeapFunction» = a1[j_0 + a1.length * -1],
a2.length ≤ j_0,
∀ int j; (j ≤ -1 ∨ j ≥ a2.length ∨ x_arr_2[j]@heapAfter_copy_0 = a2[j]@heapAfter_copy_0)
==>
x_arr_2 = a1,
x_arr_2 = a2,
x_arr_2.<created> = TRUE,
x_arr_2 = null,
a1 = null,
a2 = null,
x_arr_2[j_0]@anon_heap_copy_0«anonHeapFunction» = a1[j_0 + a1.length * -1]
```

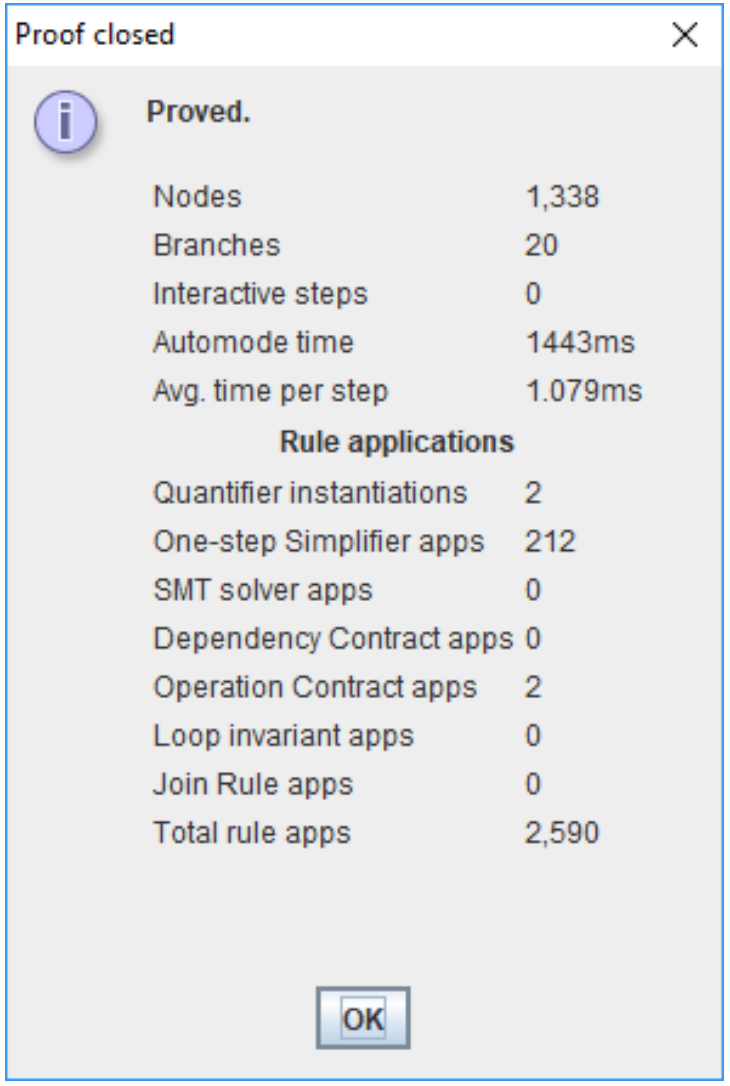
KeY: как найти иголку в бочке

```
/*@ normal_behaviour
  @ requires array != null && n >=0 && n <= array.length && n <= result.length;
  @ requires array != result;
  @ requires out >= 0 && out <= result.length - n;
  @ ensures (\forall int j; 0 <= j && j < n; result[j] == array[j]);
  @ assignable result[out..out+n-1];
  @*/
private static void copy(int[] array, int[] result, int out, int n) {
/*@
  @ loop_invariant i>=0 && i <= n;
  @ loop_invariant (\forall int j; 0 <= j && j < i; result[j+out] == array[j]);
  @ decreasing n-i;
  @ assignable result[out..out+n-1];
  @*/
  for (int i = 0; i< n; i++) {
    result[i+out] = array[i];
  }
}
```

KeY: как найти иголку в бочке

```
/*@ normal_behaviour
  @ requires array != null && n >=0 && n <= array.length && n <= result.length;
  @ requires array != result;
  @ requires out >= 0 && out <= result.length - n;
  @ ensures (\forall int j; 0 <= j && j < n; result[j+out] == array[j]);
  @ assignable result[out..out+n-1];
  @*/
private static void copy(int[] array, int[] result, int out, int n) {
/*@
  @ loop_invariant i>=0 && i <= n;
  @ loop_invariant (\forall int j; 0 <= j && j < i; result[j+out] == array[j]);
  @ decreasing n-i;
  @ assignable result[out..out+n-1];
  @*/
  for (int i = 0; i< n; i++) {
    result[i+out] = array[i];
  }
}
```

KeY: ну наконец-то



Мораль

- Динамический анализ:
  - Instant Feedback - Нет
  - Detectors – Да
  - Property-based testing – Очень Да
  - Model checking – Еще не очень
- Статический анализ:
  - Checkers/inspections – Да
  - IDE/AST matchers – Да
  - Auto verification – Даже не думайте