Mining Temporal Specifications from Object Usage

Andrzej Wasylkowski • Andreas Zeller
Saarland University

aspectj

crosscutting objects for better modularity

bug.aj

```
@interface A {}

aspect Test {
  declare @field : @A int var* : @A;
  declare @field : int var* : @A;

  interface Subject {}

  public int Subject.vara;
  public int Subject.varb;
}

class X implements Test.Subject {}
```

AspectJ Stack Trace

```
java.util.NoSuchElementException
at java.util.AbstractList$Itr.next(AbstractList.java:427)
at org.aspectj.weaver.bcel.BcelClassWeaver.
  weaveAtFieldRepeatedly(BcelClassWeaver.java:1016)
…
```

AspectJ Stack Trace

```
java.util.NoSuchElementException
at java.util.AbstractList$Itr.next(AbstractList.java:427)
at org.aspectj.weaver.bcel.BcelClassWeaver.
   weaveAtFieldRepeatedly(BcelClassWeaver.java:1016)
…
```

weaveAtFieldRepeatedly

```
for (Iterator it = c1.iterator(); it.hasNext();) {
  E e1 = (E) it.next();
  …
  for (Iterator it2 = c2.iterator(); it.hasNext();) {
    E e2 = (E) it2.next();
    …
  }
}
```

weaveAtFieldRepeatedly

```
for (Iterator it = c1.iterator(); it.hasNext();) {
  E e1 = (E) it.next();
  …
  for (Iterator it2 = c2.iterator(); it.hasNext();) {
    E e2 = (E) it2.next();
    …
  }
}
```

should be `it2`

weaveAtFieldRepeatedly

```
for (Iterator it = c1.iterator(); it.hasNext();) {
  E e1 = (E) it.next();
  …
  for (Iterator it2 = c2.iterator(); it.hasNext();) {
    E e2 = (E) it2.next();
    …
  }
}
```

should be `it2`

- Invalid iterator usage:
  hasNext() should precede next()

Preconditions

- Invoking next() with no next element violates a *precondition*
- Traditional preconditions are axiomatic – describing the *state* of the system
- How do we reach that state?

Axiomatic Preconditions

```
/*@ requires propertyList.size() >= 1
  @ requires propertyList.get(0) instanceof Class
  @ requires \forall int i; 0 < i && i < propertyList.size();
  @   propertyList.get(i) instanceof StructuralPropertyDescriptor
  @   && ((StructuralPropertyDescriptor)propertyList.get(i)).
  @     getNodeClass() == propertyList.get(0)
  @*/

static List reapPropertyList(List propertyList)
```

Operational Preconditions

```
public List getPL (Set ps) {
  List l = new ArrayList ();
  createPropertyList (this.cl, l);
  Iterator it = ps.iterator ();
  while (it.hasNext ()) {
    Property p = (Property) it.next ();
    addProperty (p, l);
  }
  reapPropertyList (l);
  if (l.size () == 1)
    Debug.log ("Empty property list");
  return l;
}
```

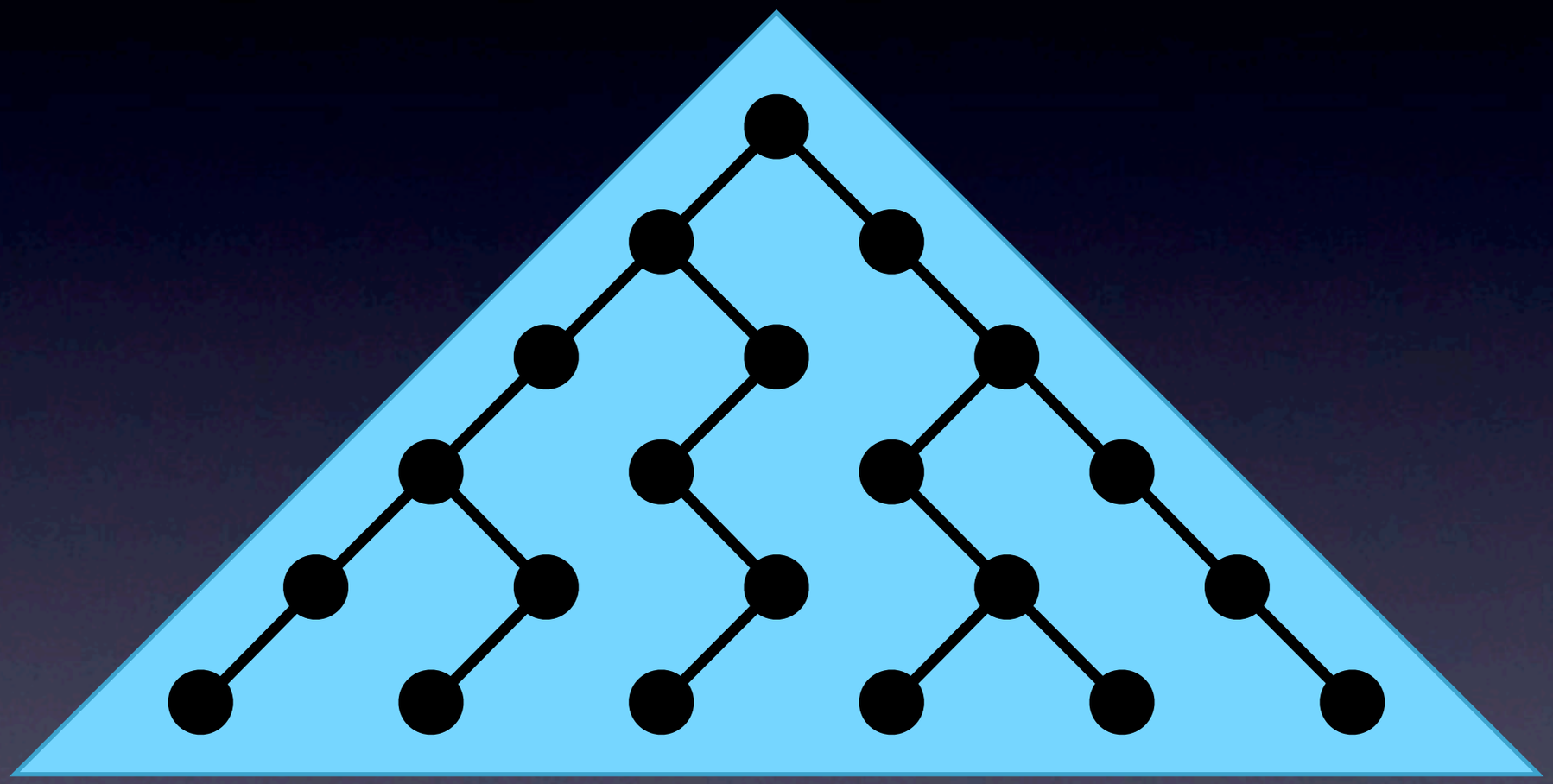
Operational Preconditions

```
public List getPL (Set ps) {
  List l = new ArrayList ();
  createPropertyList (this.cl, l);
  Iterator it = ps.iterator ();
  while (it.hasNext ()) {
    Property p = (Property) it.next ();
    addProperty (p, l);
  }
  reapPropertyList (l);
  if (l.size () == 1)
    Debug.log ("Empty property list");
  return l;
}
```

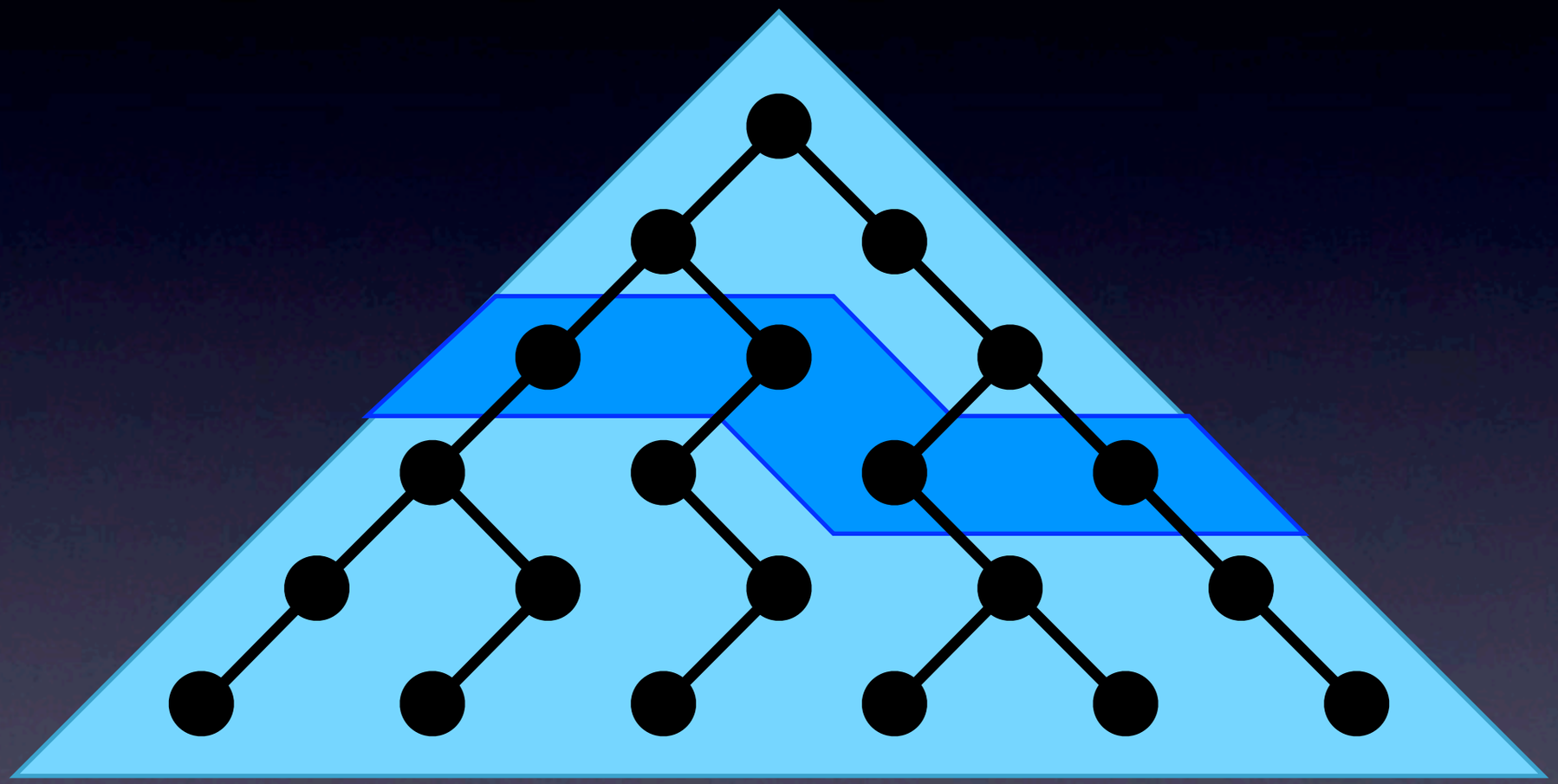
CTL

- Computation Tree Logic
- Standard logic operators ($\wedge, \vee, \neg, \Rightarrow, \ldots$)
- Path operators (EF, AF, EG, AG, …)

CTL Path Operators



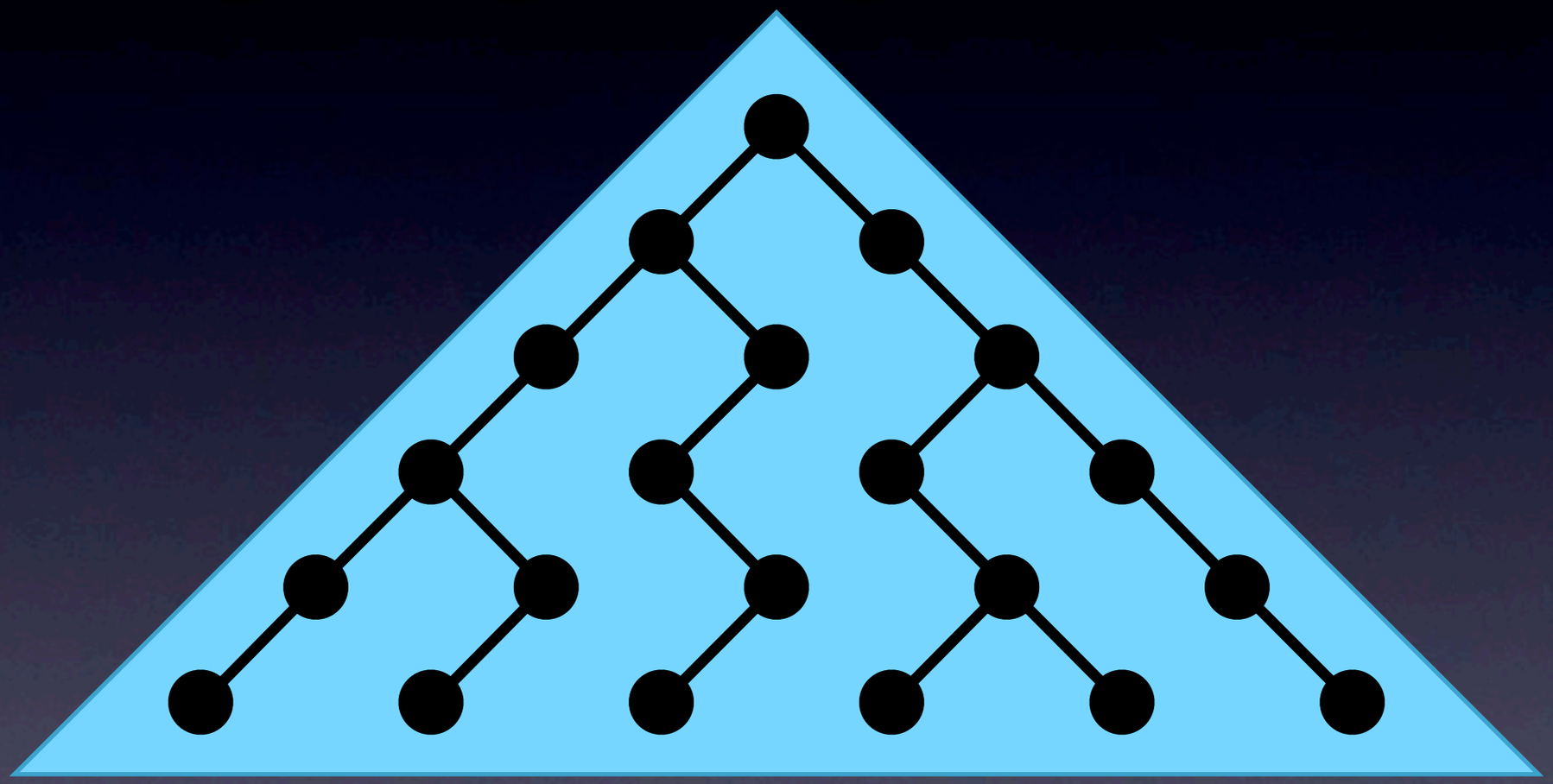
CTL Path Operators



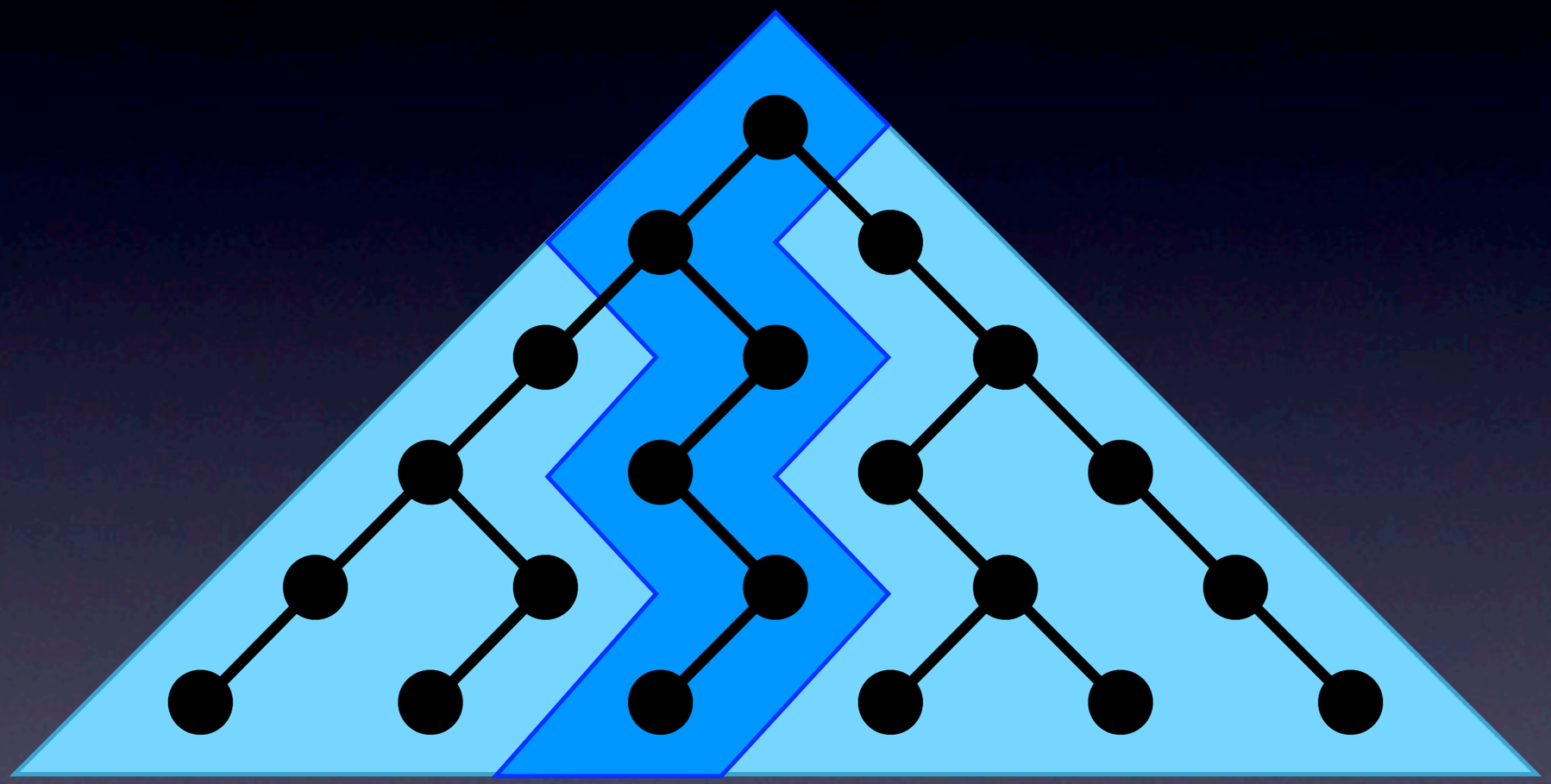
AF p

"on all paths finally p"

CTL Path Operators



CTL Path Operators



EG p

"there exists a path on which globally p"

Operational Preconditions

```
public List getPL (Set ps) {
  List l = new ArrayList ();
  createPropertyList (this.cl, l);
  Iterator it = ps.iterator ();
  while (it.hasNext ()) {
    Property p = (Property) it.next ();
    addProperty (p, l);
  }
  reapPropertyList (l);
  if (l.size () == 1)
    Debug.log ("Empty property list");
  return l;
}
```

Operational Preconditions

```
public List getPL (Set ps) {
  List l = new ArrayList ();
  createPropertyList (this.cl, l);
  Iterator it = ps.iterator ();
  while (it.hasNext ()) {
    Property p = (Property) it.next ();
    addProperty (p, l);
  }
  reapPropertyList (l);
  if (l.size () == 1)
    Debug.log ("Empty property list");
  return l;
}
```

Operational Precondition

```
AF ArrayList.<init>
AF createPropertyList
AG (ArrayList.<init> =>
  AF createPropertyList)
AG (createPropertyList =>
  EF addProperty)
…
```

Operational Preconditions

- Describe how the state can be reached
- Can be fully automatically learned and used to check the program
- Violations include "how to fix" suggestions

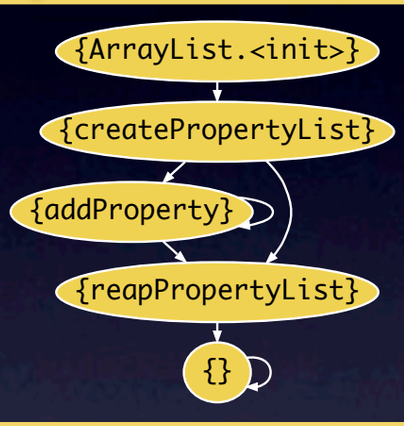
Operational Precondition

```
AF ArrayList.<init>
AF createPropertyList
AG (ArrayList.<init> =>
  AF createPropertyList)
AG (createPropertyList =>
  EF addProperty)
…
```

Tikanga

Program

Kripke structures



CTL specifications

```
AF ArrayList.<init>
EF addProperty
EF reapPropertyList
```

Operational preconditions

```
AF ArrayList.<init>
EF reapPropertyList
```

Ranked violations

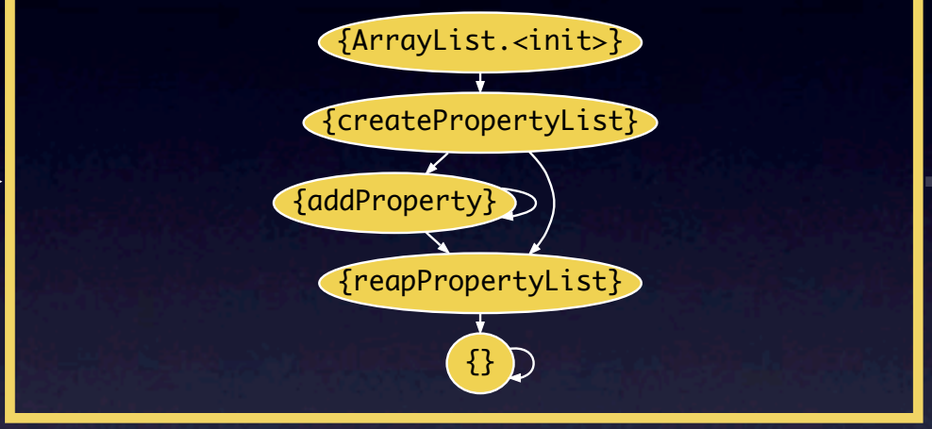
✓
```
AF ArrayList.<init>
EF reapPropertyList
```

✗
```
~~AF ArrayList.<init>~~
EF reapPropertyList
```

Tikanga

Program

Kripke structures



CTL specifications

```
AF ArrayList.<init>
EF addProperty
EF reapPropertyList
```

Operational preconditions

```
AF ArrayList.<init>
EF reapPropertyList
```

Ranked violations

✓
```
AF ArrayList.<init>
EF reapPropertyList
```

✗
```
~~AF ArrayList.<init>~~
EF reapPropertyList
```

Kripke Structures

```
public List getPL (Set ps) {
  List l = new ArrayList ();
  createPropertyList (this.cl, l);
  Iterator it = ps.iterator ();
  while (it.hasNext ()) {
    Property p = (Property) it.next ();
    addProperty (p, l);
  }
  reapPropertyList (l);
  if (l.size () == 1)
    Debug.log ("Empty property list");
  return l;
}
```

Kripke Structures

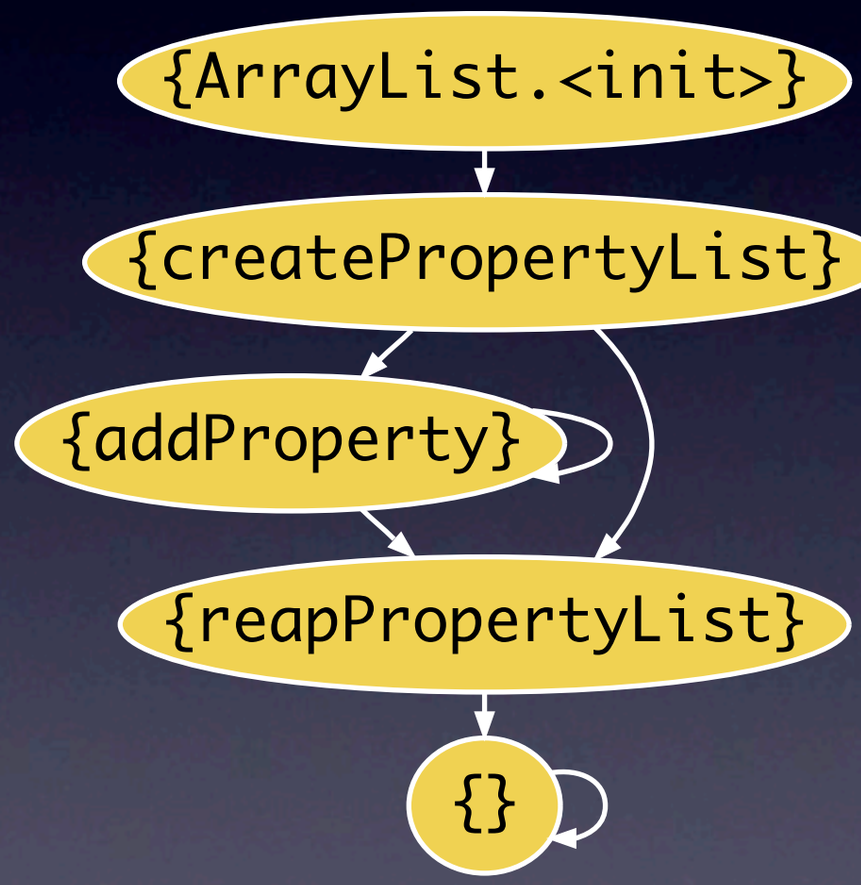
```
public List getPL (Set ps) {
  List l = new ArrayList ();
  createPropertyList (this.cl, l);
  Iterator it = ps.iterator ();
  while (it.hasNext ()) {
    Property p = (Property) it.next ();
    addProperty (p, l);
  }
  reapPropertyList (l);
  if (l.size () == 1)
    Debug.log ("Empty property list");
  return l;
}
```

Kripke Structures

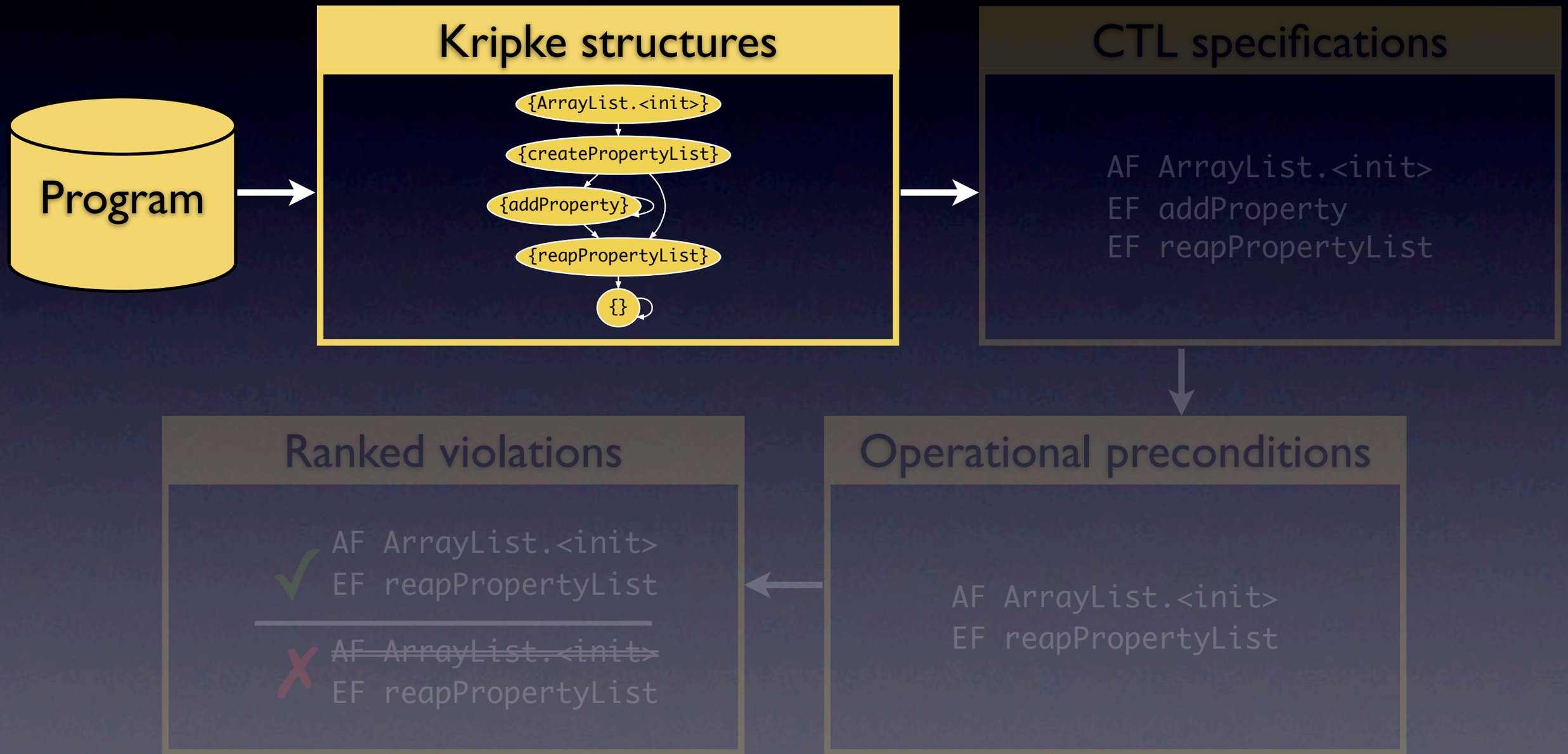
```
public List getPL (Set ps) {
  List l = new ArrayList ();
  createPropertyList (this.cl, l);
  Iterator it = ps.iterator ();
  while (it.hasNext ()) {
    Property p = (Property) it.next ();
    addProperty (p, l);
  }
  reapPropertyList (l);
  if (l.size () == 1)
    Debug.log ("Empty property list");
  return l;
}
```

Kripke Structures

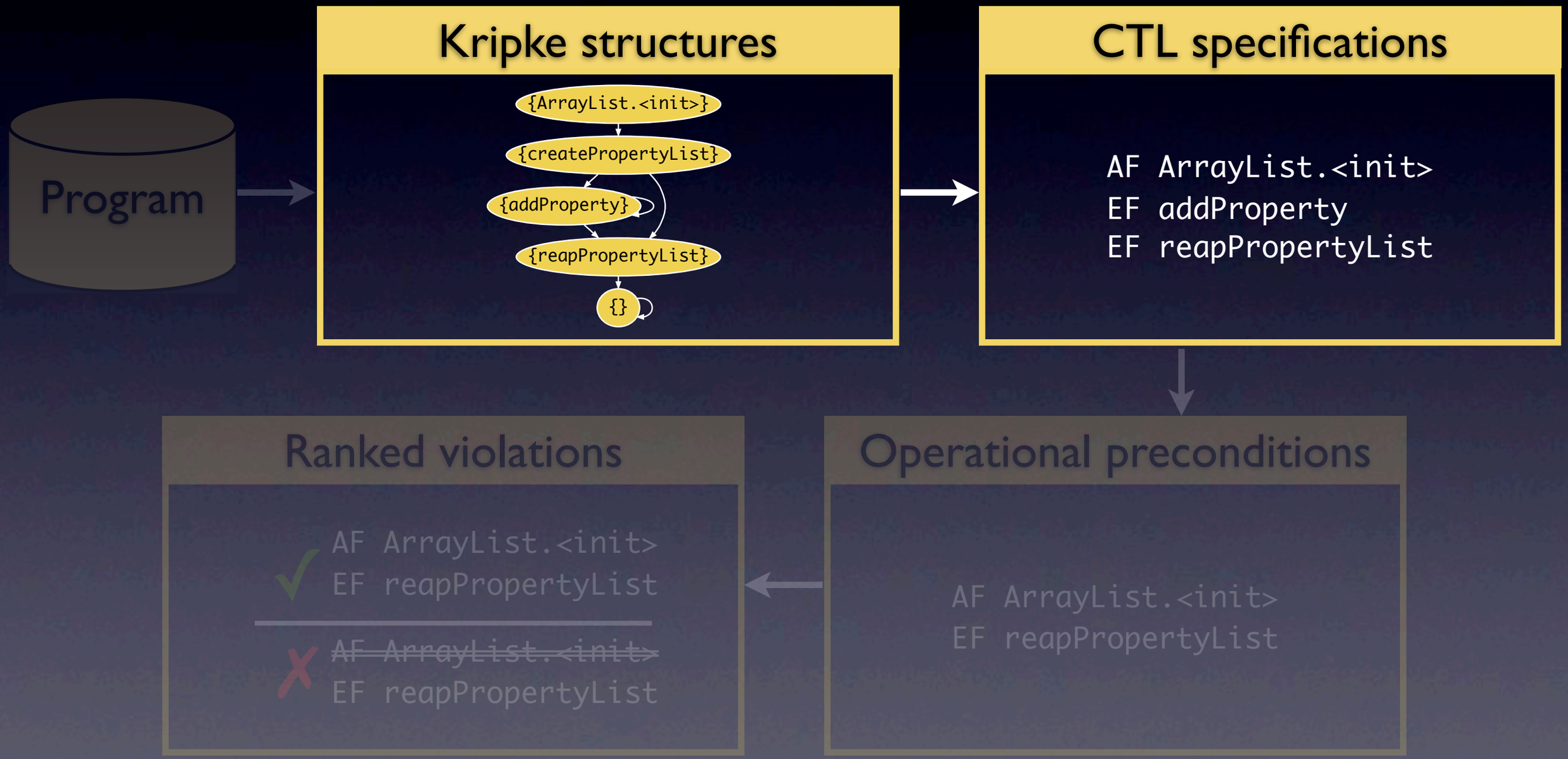
```
public List getPL (Set ps) {
  List l = new ArrayList ();
  createPropertyList (this.cl, l);
  Iterator it = ps.iterator ();
  while (it.hasNext ()) {
    Property p = (Property) it.next ();
    addProperty (p, l);
  }
  reapPropertyList (l);
  if (l.size () == 1)
    Debug.log ("Empty property list");
  return l;
}
```



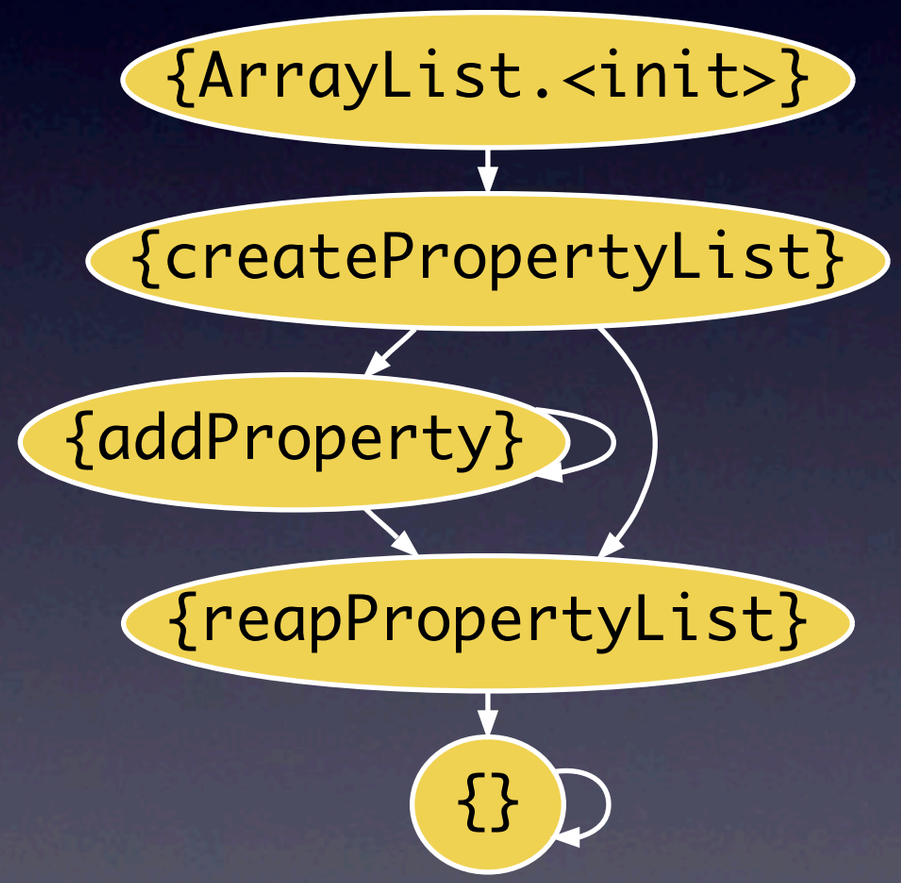
Tikanga


Program
Kripke structures
{ArrayList.<init>}
{createPropertyList}
{addProperty}
{reapPropertyList}
{}
CTL specifications
AF ArrayList.<init>
EF addProperty
EF reapPropertyList
Operational preconditions
AF ArrayList.<init>
EF reapPropertyList
Ranked violations
AF ArrayList.<init>
EF reapPropertyList
AF ArrayList.<init>
EF reapPropertyList

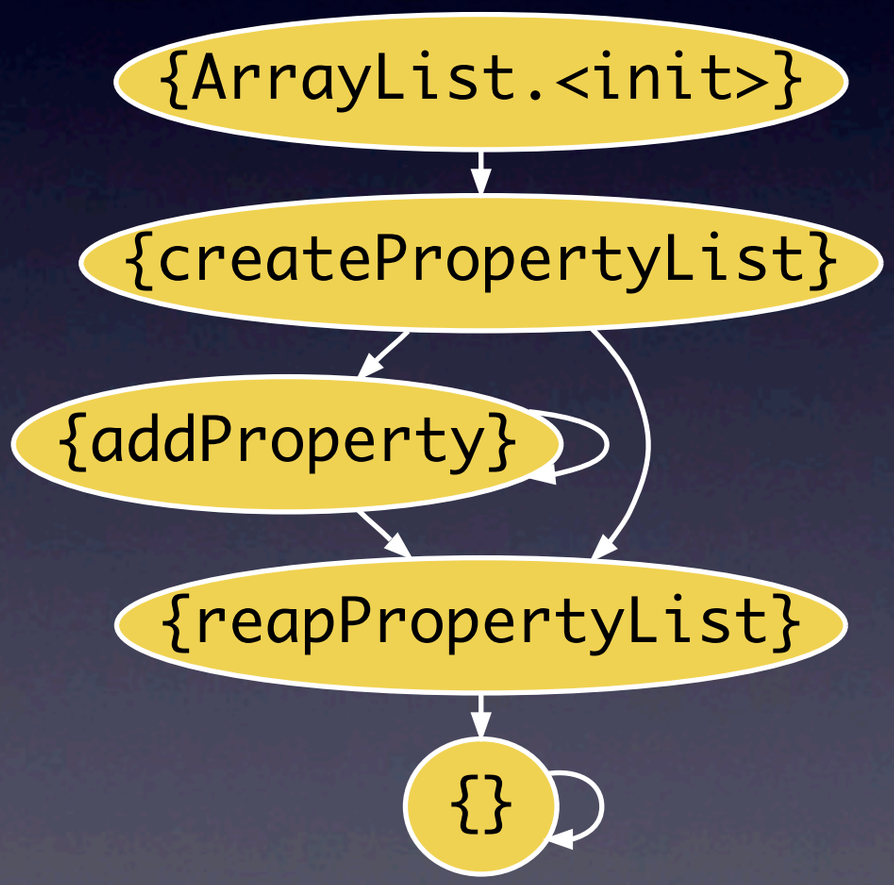
Tikanga


Program
Kripke structures
{ArrayList.<init>}
{createPropertyList}
{addProperty}
{reapPropertyList}
{}
CTL specifications
AF ArrayList.<init>
EF addProperty
EF reapPropertyList
Operational preconditions
AF ArrayList.<init>
EF reapPropertyList
Ranked violations
AF ArrayList.<init>
EF reapPropertyList
AF ArrayList.<init>
EF reapPropertyList

Extracting CTL Specifications



Extracting CTL Specifications



Atomic propositions

ArrayList.<init>
createPropertyList
addProperty
reapPropertyList

Extracting CTL Specifications

Atomic propositions

```
ArrayList.<init>
createPropertyList
addProperty
reapPropertyList
```

Extracting CTL Specifications

Atomic propositions

```
ArrayList.<init>
createPropertyList
addProperty
reapPropertyList
```

CTL templates

```
AF p
AG (p1 => AF p2)
AG (p1 => EF p2)
```

Extracting CTL Specifications

Atomic propositions

```
ArrayList.<init>
createPropertyList
addProperty
reapPropertyList
```

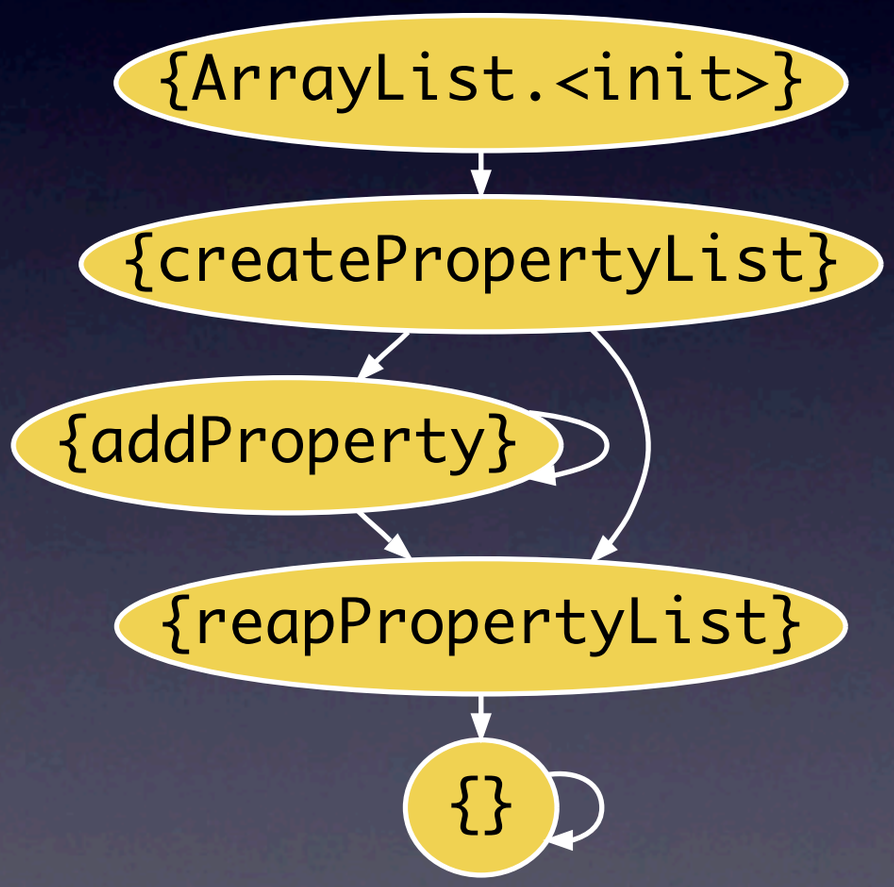
CTL templates

```
AF p
AG (p1 => AF p2)
AG (p1 => EF p2)
```

CTL formulas

```
AF ArrayList.<init>
AF createPropertyList
AF addProperty
AF reapPropertyList
AG (ArrayList.<init> =>
  AF createPropertyList)
AG (ArrayList.<init> =>
  AF addProperty)
AG (ArrayList.<init> =>
  AF reapPropertyList)
AG (createPropertyList =>
  AF addProperty)
…
```

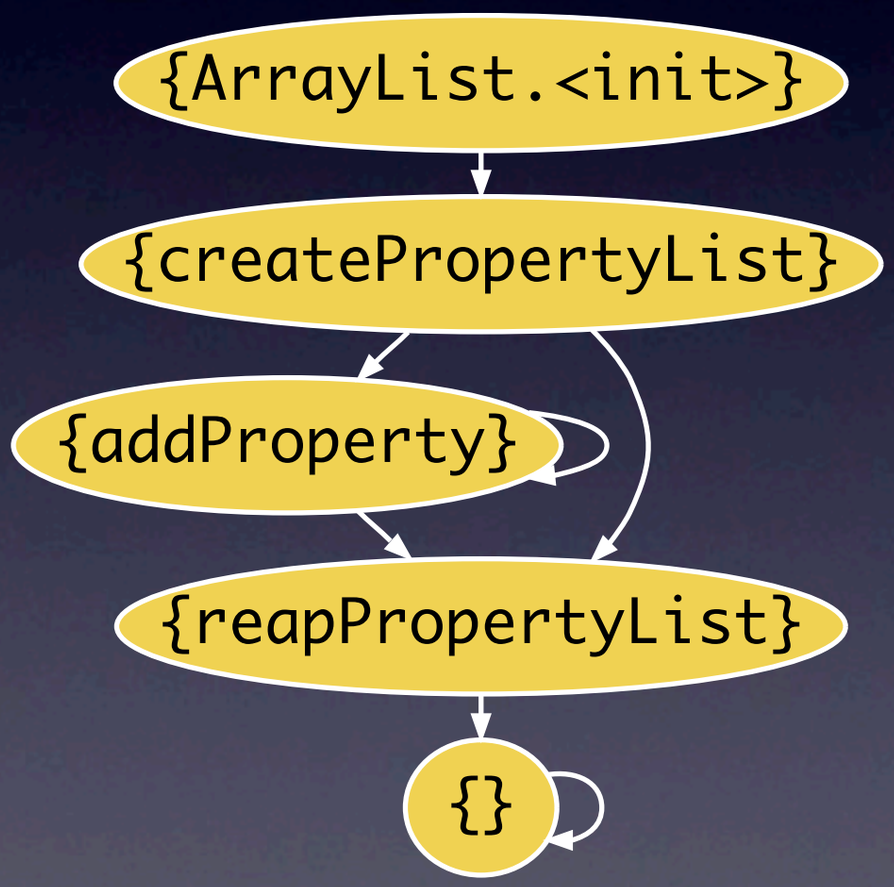
Extracting CTL Specifications



CTL formulas

```
AF ArrayList.<init>
AF createPropertyList
AF addProperty
AF reapPropertyList
AG (ArrayList.<init> =>
  AF createPropertyList)
AG (ArrayList.<init> =>
  AF addProperty)
AG (ArrayList.<init> =>
  AF reapPropertyList)
AG (createPropertyList =>
  AF addProperty)
…
```

Extracting CTL Specifications

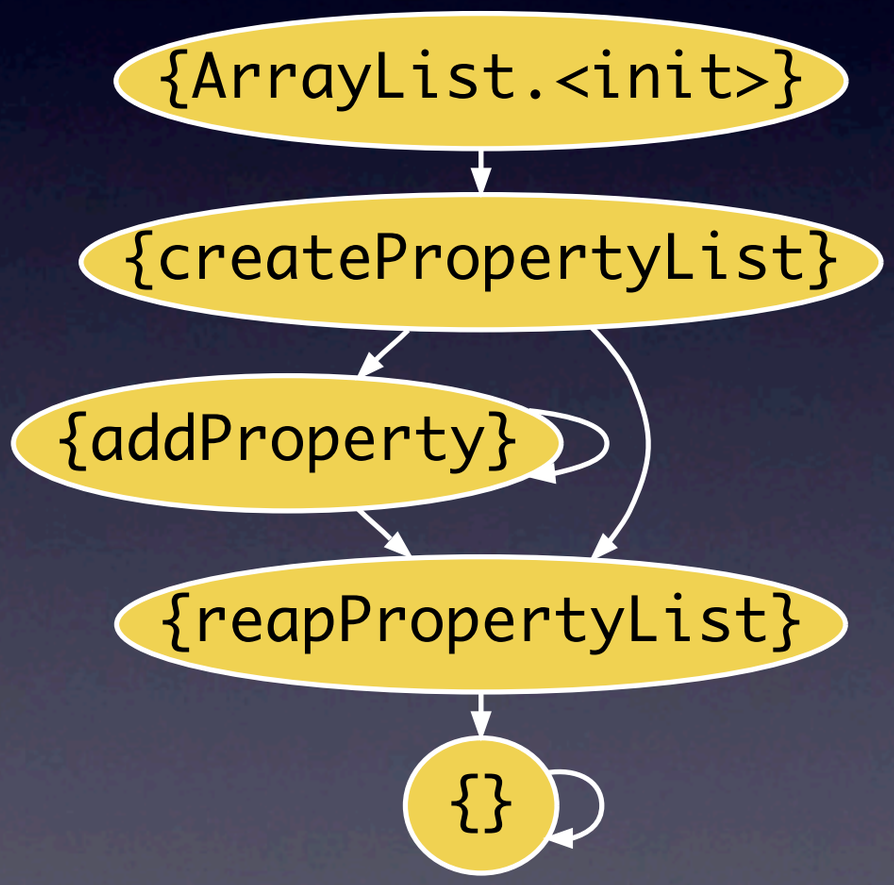


CTL formulas

✓ AF ArrayList.<init>
AF createPropertyList
AF addProperty
AF reapPropertyList
AG (ArrayList.<init> =>
  AF createPropertyList)
AG (ArrayList.<init> =>
  AF addProperty)
AG (ArrayList.<init> =>
  AF reapPropertyList)
AG (createPropertyList =>
  AF addProperty)
…

Clarke, E. M., E. A. Emerson, and A. P. Sistla. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems* 8, no. 2 (April): 244–263.

Extracting CTL Specifications

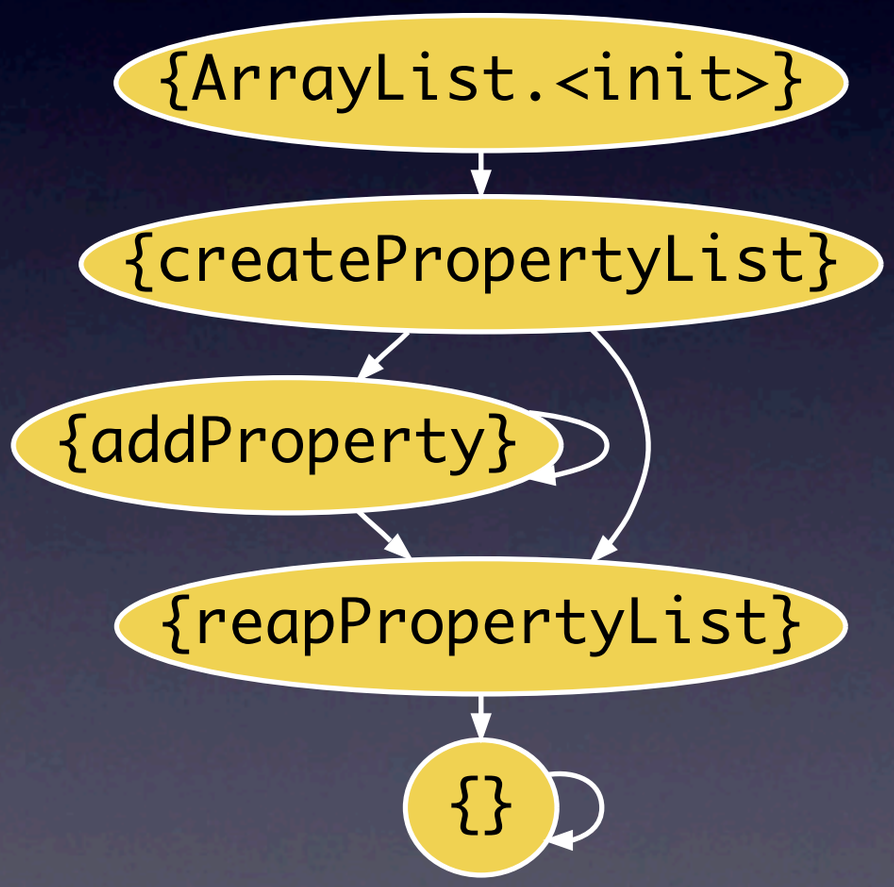


CTL formulas

✓ AF ArrayList.<init>
✓ AF createPropertyList
AF addProperty
AF reapPropertyList
AG (ArrayList.<init> =>
  AF createPropertyList)
AG (ArrayList.<init> =>
  AF addProperty)
AG (ArrayList.<init> =>
  AF reapPropertyList)
AG (createPropertyList =>
  AF addProperty)
…

Clarke, E. M., E. A. Emerson, and A. P. Sistla. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems* 8, no. 2 (April): 244–263.

Extracting CTL Specifications

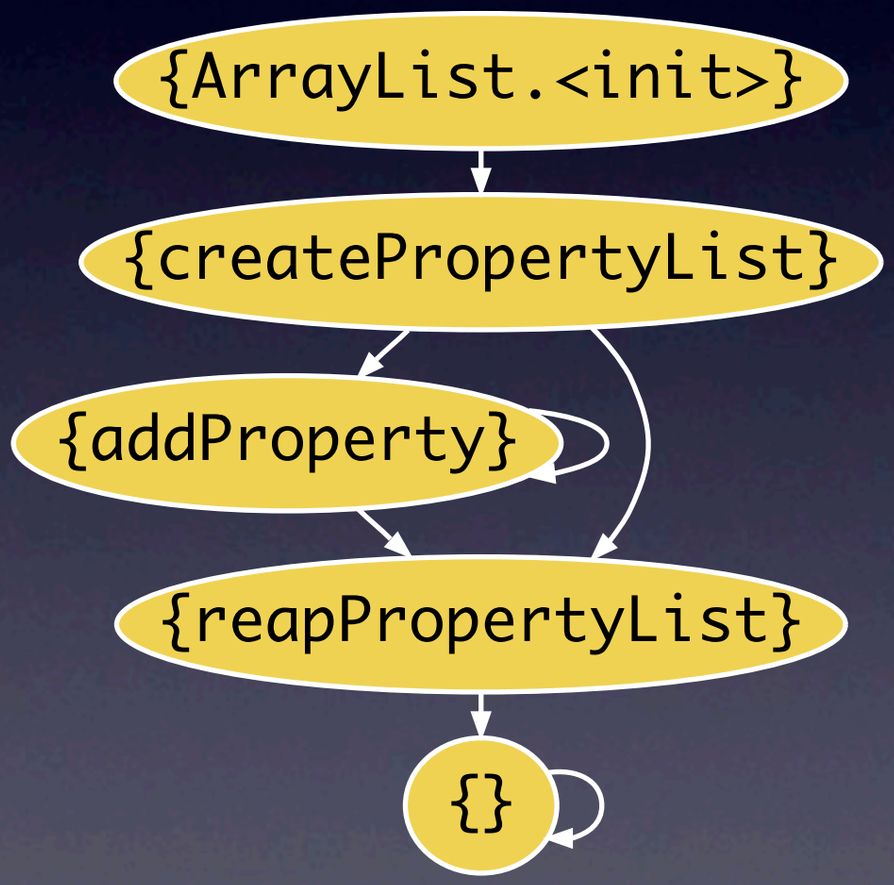


CTL formulas

✓ AF ArrayList.<init>
✓ AF createPropertyList
✗ AF addProperty
AF reapPropertyList
AG (ArrayList.<init> =>
  AF createPropertyList)
AG (ArrayList.<init> =>
  AF addProperty)
AG (ArrayList.<init> =>
  AF reapPropertyList)
AG (createPropertyList =>
  AF addProperty)
…

Clarke, E. M., E. A. Emerson, and A. P. Sistla. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems* 8, no. 2 (April): 244–263.

Extracting CTL Specifications



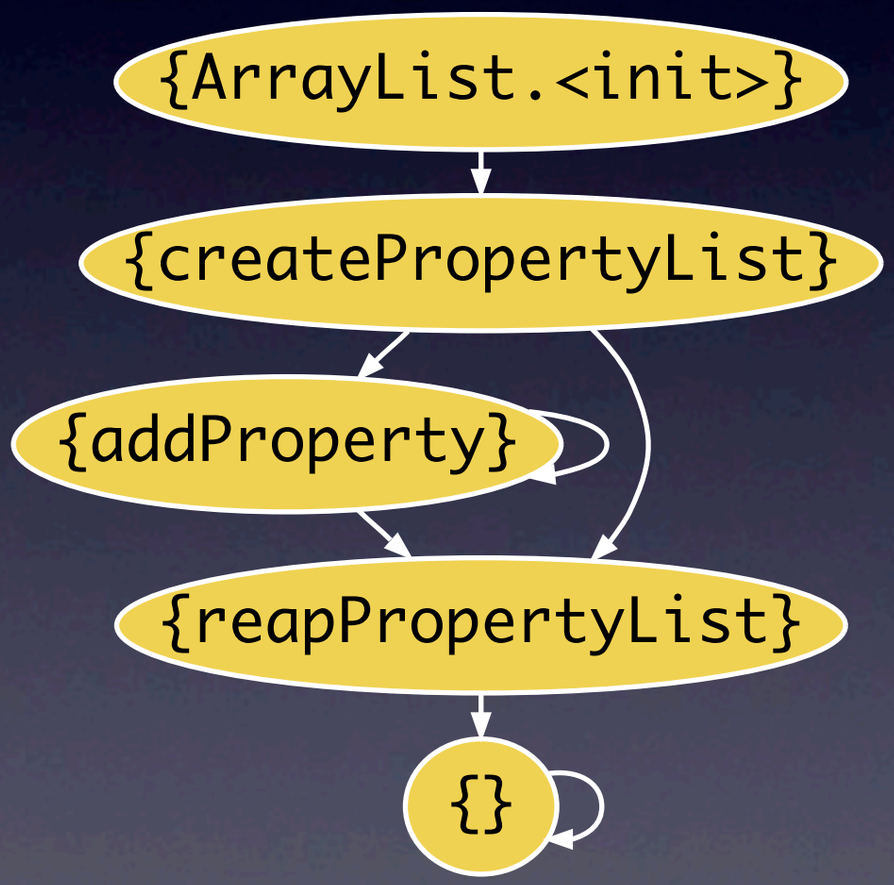
CTL formulas

✓ AF ArrayList.<init>
✓ AF createPropertyList
✗ AF addProperty
✗ AF reapPropertyList
✓ AG (ArrayList.<init> =>
  AF createPropertyList)
✗ AG (ArrayList.<init> =>
  AF addProperty)
✗ AG (ArrayList.<init> =>
  AF reapPropertyList)
✗ AG (createPropertyList =>
  AF addProperty)

…

Clarke, E. M., E. A. Emerson, and A. P. Sistla. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems* 8, no. 2 (April): 244–263.

Extracting CTL Specifications

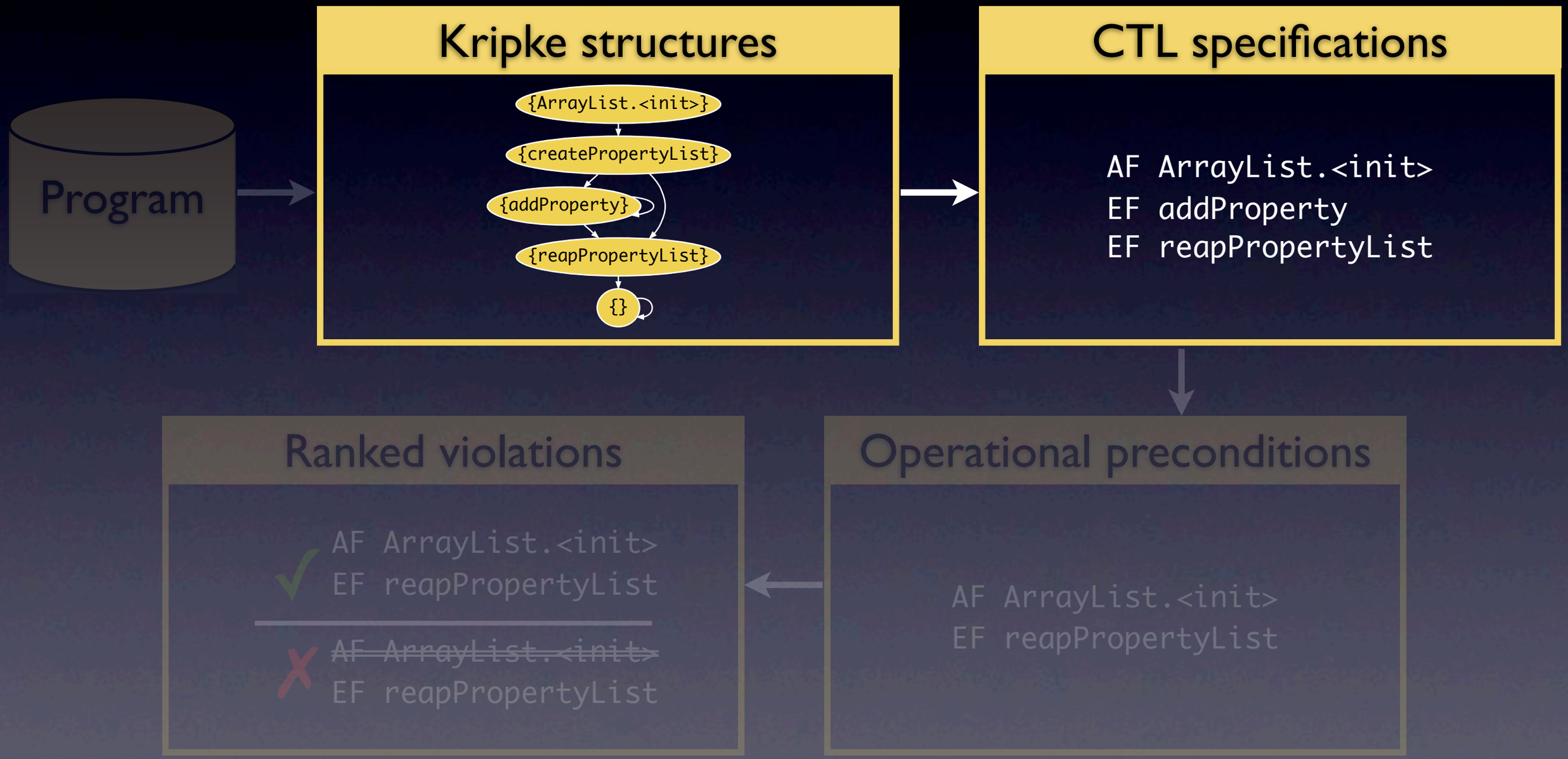


CTL specification

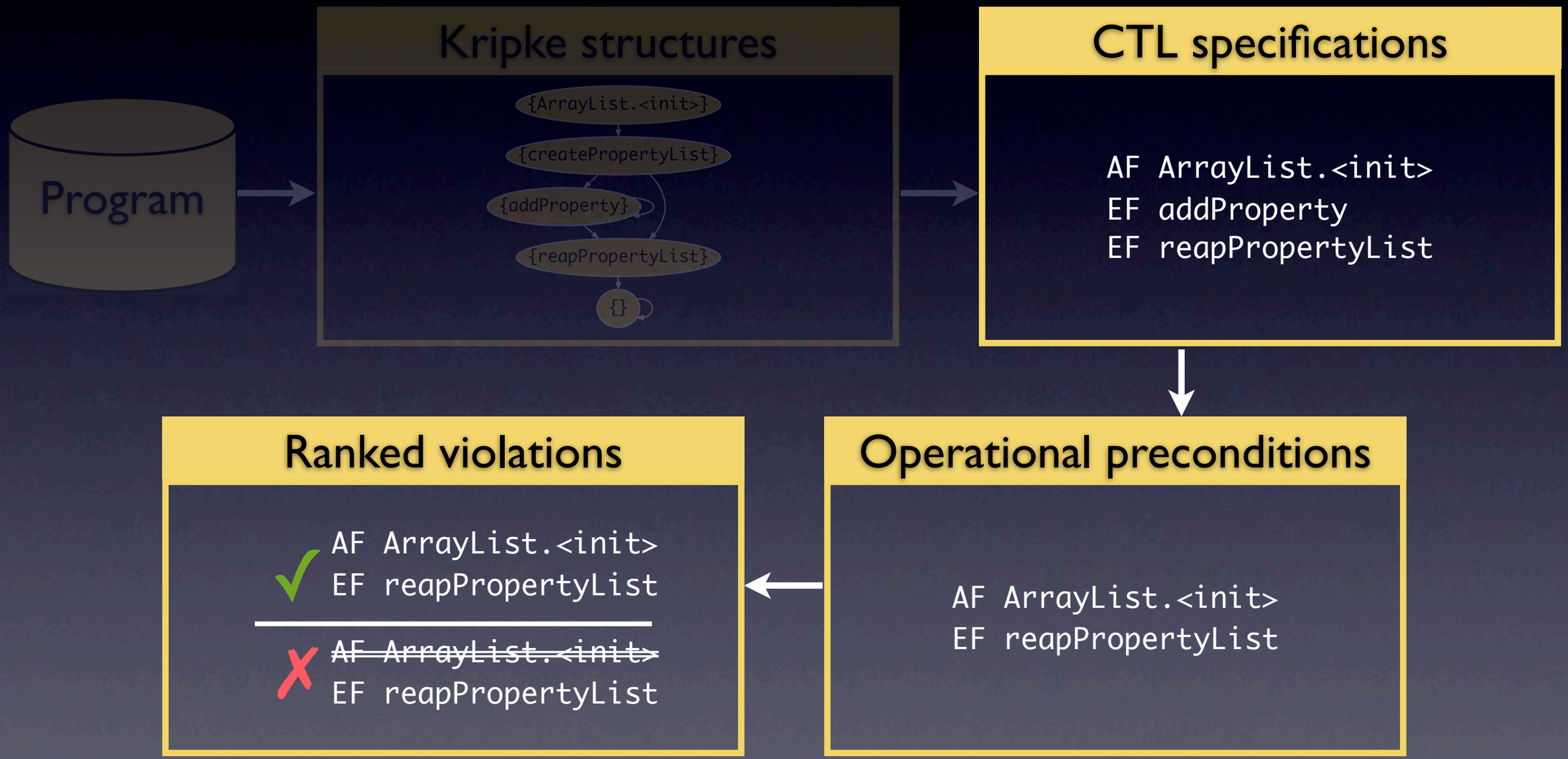
✓ AF ArrayList.<init>
✓ AF createPropertyList
✓ AG (ArrayList.<init> =>
  AF createPropertyList)

…

Tikanga


Program
Kripke structures
{ArrayList.<init>}
{createPropertyList}
{addProperty}
{reapPropertyList}
{}
CTL specifications
AF ArrayList.<init>
EF addProperty
EF reapPropertyList
Operational preconditions
AF ArrayList.<init>
EF reapPropertyList
Ranked violations
AF ArrayList.<init>
EF reapPropertyList
AF ArrayList.<init>
EF reapPropertyList

Tikanga


Program
Kripke structures
{ArrayList.<init>}
{createPropertyList}
{addProperty}
{reapPropertyList}
{}
CTL specifications
AF ArrayList.<init>
EF addProperty
EF reapPropertyList
Operational preconditions
AF ArrayList.<init>
EF reapPropertyList
Ranked violations
AF ArrayList.<init>
EF reapPropertyList
AF ArrayList.<init>
EF reapPropertyList

Operational Preconditions

```
AF countTokens
AF hasMoreTokens
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
…

AF hasMoreTokens
AG (hasMoreTokens =>
  AF countTokens)
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
…
```

```
AF StringTokenizer.<init>
AF hasMoreTokens
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
…

…
```

Operational Preconditions

```
AF countTokens
AF hasMoreTokens
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
```

…

```
AF hasMoreTokens
AG (hasMoreTokens =>
  AF countTokens)
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
```

…

```
AF StringTokenizer.<init>
AF hasMoreTokens
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
```

…

Lindig, Christian. 2007. Mining patterns and violations using concept analysis. Technical report, Saarland University. http://www.st.cs.uni-saarland.de/publications/files/lindig-tr-2007.pdf

Operational Preconditions

```
AF countTokens
AF hasMoreTokens
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =
  EF hasMoreTok
…

AF hasMoreToker
AG (hasMoreToke
  AF countToker
AG (hasMoreToke
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
…
```

```
AF StringTokenizer.<init>
AF hasMoreTokens
AG (hasMoreTokens =>
                    >
                  ens)
```

Operational Precondition

```
AF hasMoreTokens
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
…
```

Lindig, Christian. 2007. Mining patterns and violations using concept analysis. Technical report, Saarland University. http://www.st.cs.uni-saarland.de/publications/files/lindig-tr-2007.pdf

Violations

Operational Precondition

```
AF hasMoreTokens
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
…
```

```
AF StringTokenizer.<init>
AF hasMoreTokens
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
…
```

```
AF hasMoreTokens
AG (hasMoreTokens =>
  AF countTokens)
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
…
```

```
AF hasMoreTokens
AG (hasMoreTokens =>
  AF countTokens)
AG (hasMoreTokens =>
  EF nextToken)
…
```

Lindig, Christian. 2007. Mining patterns and violations using concept analysis. Technical report, Saarland University. http://www.st.cs.uni-saarland.de/publications/files/lindig-tr-2007.pdf

Violations

Operational Precondition

```
AF hasMoreTokens
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
…
```

```
AF StringTokenizer.<init>
AF hasMoreTokens
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
…

AF hasMoreTokens
AG (hasMoreTokens =>
  AF countTokens)
AG (hasMoreTokens =>
  EF nextToken)
…
```

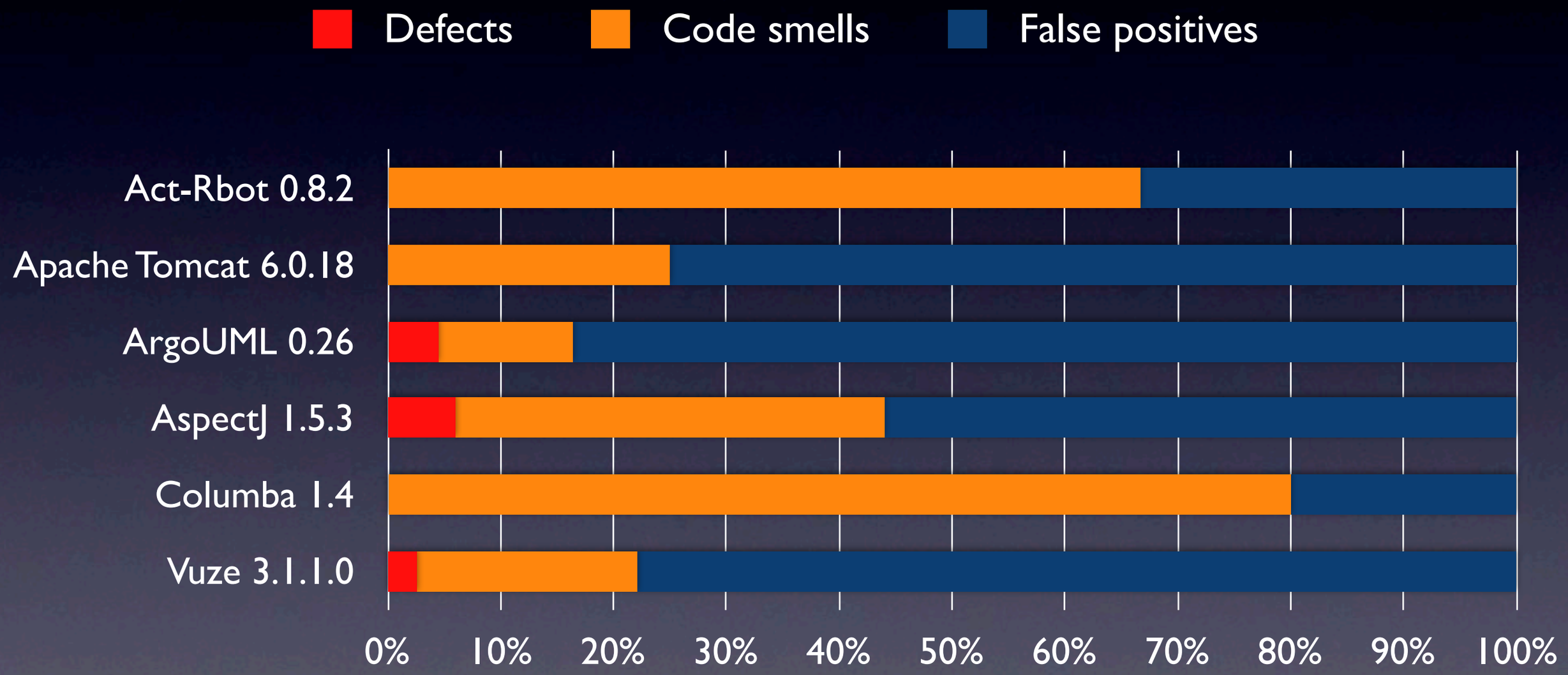
```
  AF countTokens)
AG (hasMoreTokens =>
  EF nextToken)
AG (nextToken =>
  EF hasMoreTokens)
…
```

Lindig, Christian. 2007. Mining patterns and violations using concept analysis. Technical report, Saarland University. http://www.st.cs.uni-saarland.de/publications/files/lindig-tr-2007.pdf

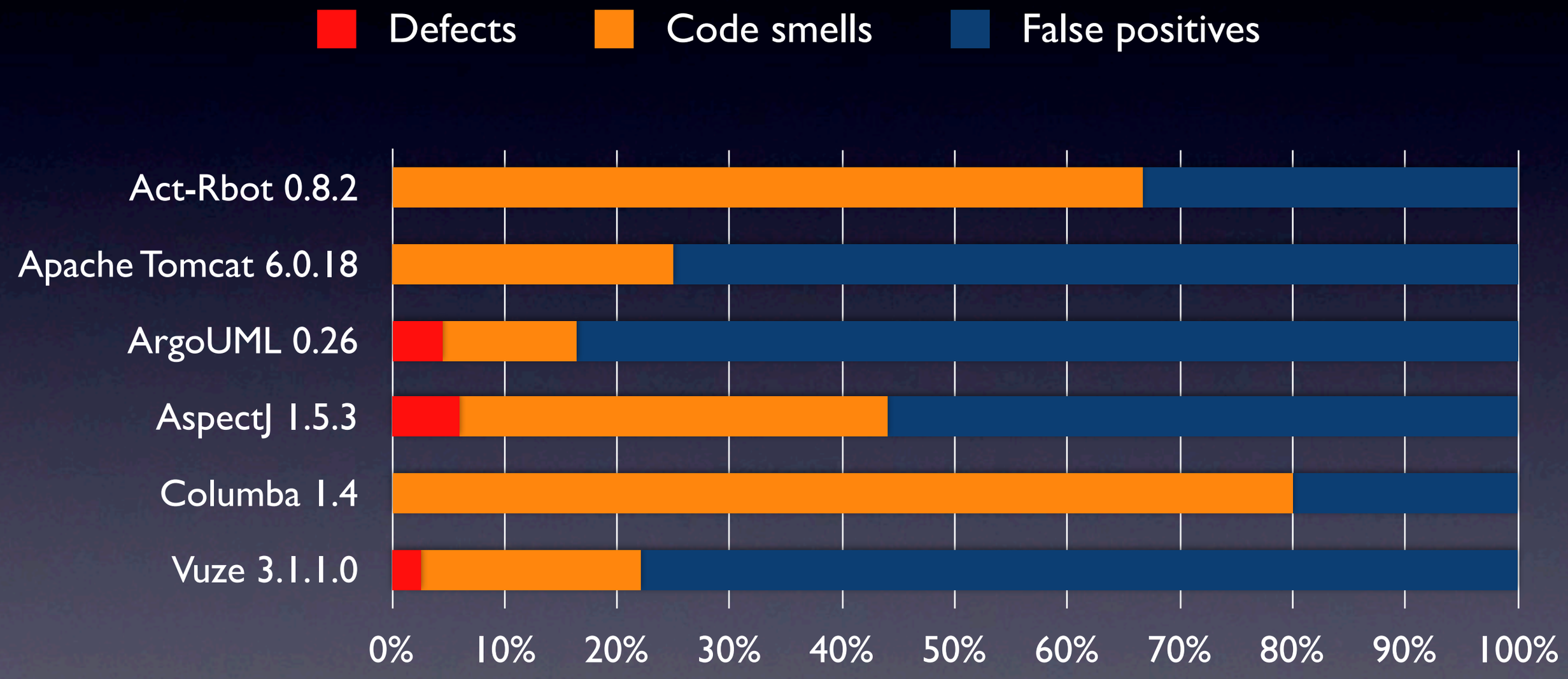
Evaluation

- 7 projects examined
- Analysis time < 15 minutes / project
- 747 violations found
- Examined top 25% for each project

Violations


Defects
Code smells
False positives
Act-Rbot 0.8.2
Apache Tomcat 6.0.18
ArgoUML 0.26
AspectJ 1.5.3
Columba 1.4
Vuze 3.1.1.0
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%

Violations



Average true positive rate per project: 42%

An AspectJ Defect

```
for (Iterator it = c1.iterator(); it.hasNext();) {
  E e1 = (E) it.next();
  …
  for (Iterator it2 = c2.iterator(); it.hasNext();) {
    E e2 = (E) it2.next();
    …
  }
}
```

An AspectJ Defect

```
for (Iterator it = c1.iterator(); it.hasNext();) {
  E e1 = (E) it.next();
  …
  for (Iterator it2 = c2.iterator(); it.hasNext();) {
    E e2 = (E) it2.next();
    …
  }
}
```

should be `it2`

An AspectJ Code Smell

```
List propertyList = new ArrayList(1);
createPropertyList(ReferencePointcut.class, propertyList);
return reapPropertyList(propertyList);
```

An AspectJ Code Smell

```
List propertyList = new ArrayList(1);
createPropertyList(ReferencePointcut.class, propertyList);
// add the type thingy here
return reapPropertyList(propertyList);
```

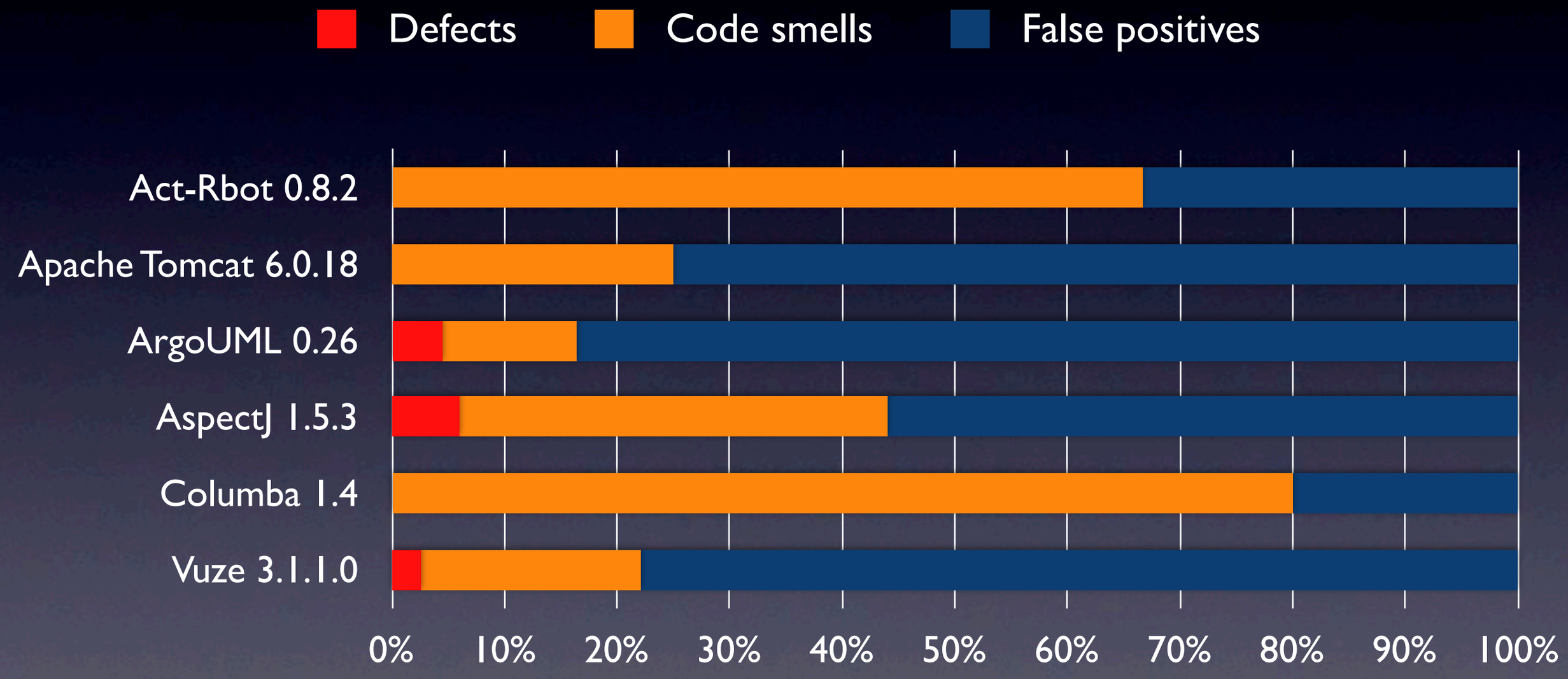
An ArgoUML False Positive

```
public Object clone() {
  FigObject figClone = (FigObject) super.clone();
  Iterator it = figClone.getFigs().iterator();
  figClone.setBigPort((FigRect) it.next());
  figClone.cover = (FigRect) it.next();
  figClone.setNameFig((FigText) it.next());
  return figClone;
}
```

An ArgoUML Alternative

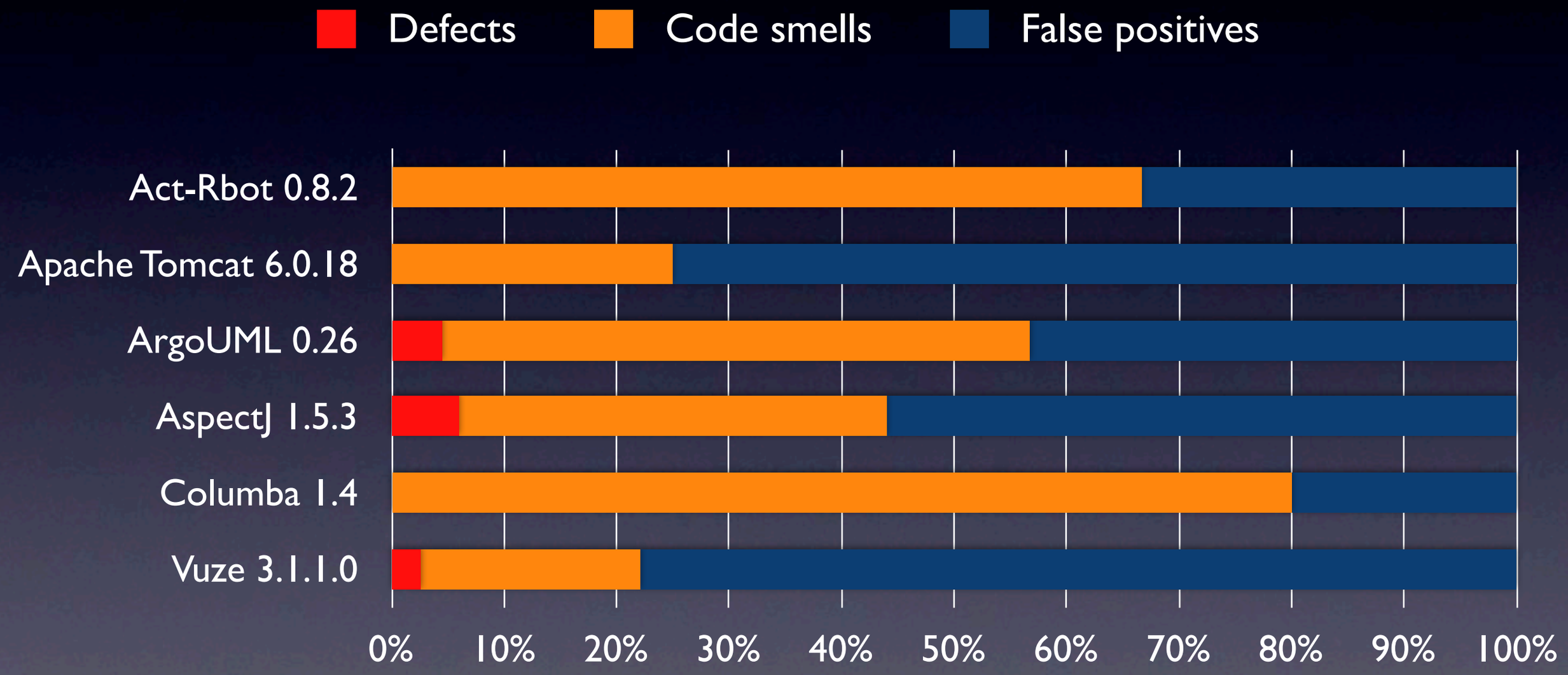
```
public Object clone() {
    final FigNodeModelElement clone = …

    final Iterator cloneIter = clone.getFigs().iterator();
    for (Object thisFig : getFigs()) {
        final Fig cloneFig = (Fig) cloneIter.next();
        if (thisFig == getBigPort()) {
            clone.setBigPort(cloneFig);
        }
        if (thisFig == nameFig) {
            clone.nameFig = (FigSingleLineText) thisFig;
        }
        …
    }
    return clone;
}
```

Violations



Average true positive rate per project: 42%

Violations



Average true positive rate per project: 49%

Operational Preconditions

```
public List getPL (Set ps) {
  List l = new ArrayList ();
  createPropertyList (this.cl, l);
  Iterator it = ps.iterator ();
  while (it.hasNext ()) {
    Property p = (Property) it.next ();
    addProperty (p, l);
  }
  reapPropertyList (l);
  if (l.size () == 1)
    Debug.log ("Empty property list");
  return l;
}
```

Operational Precondition

```
AF ArrayList.<init>
AF createPropertyList
AG (ArrayList.<init> =>
  AF createPropertyList)
AG (createPropertyList =>
  EF addProperty)
…
```

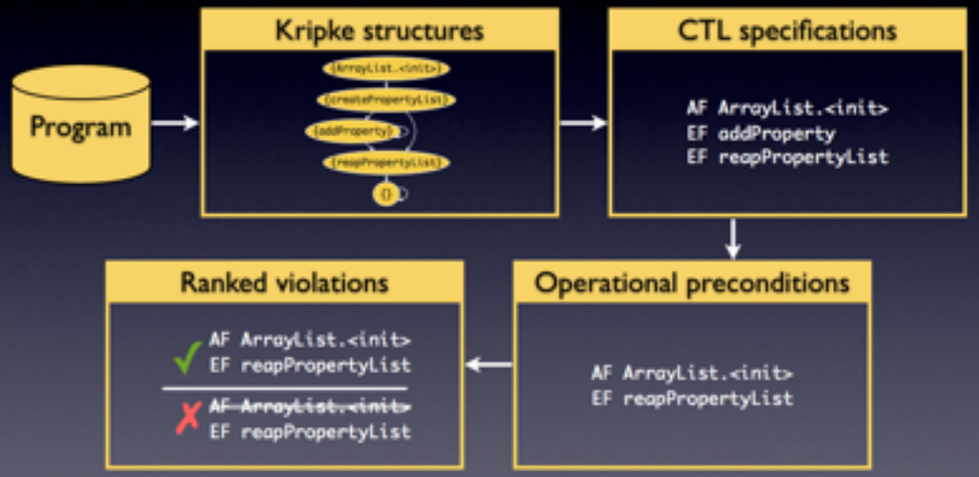
Extracting CTL Specifications

CTL specification

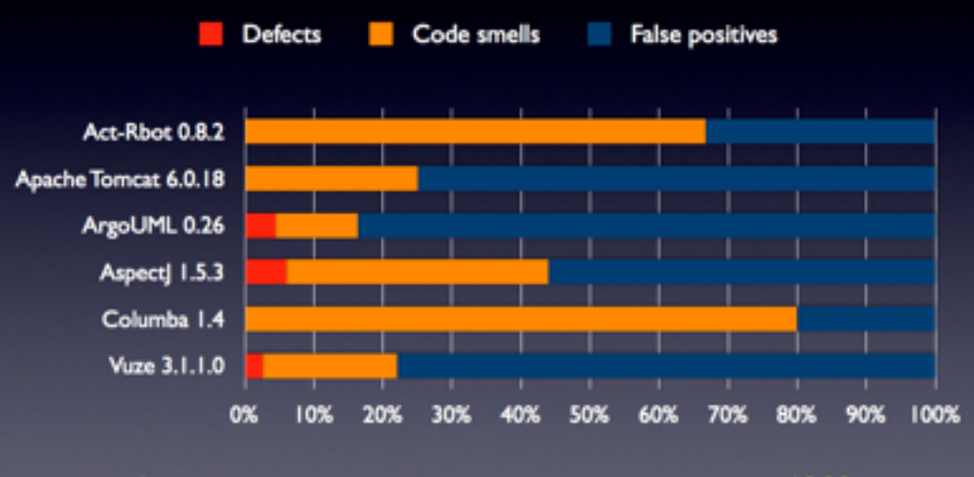
✓ AF ArrayList.<init>
✓ AF createPropertyList
✓ AG (ArrayList.<init> =>
  AF createPropertyList)
…

Summary

Tikanga



Violations



Average true positive rate per project: 42%