Model Based testing
+
Approcci Combinatori
+
Misura della Copertura

Fabrizio Pastore
www.fabriziopastore.com

Esecuzione Automatica di Casi di Test (di Unità)

- Basato su librerie come
  - JUnit (Java), NUnit (C#), CppUnit(C/C++)
  - forniscono utility methods per fare il setup dei test, definire asserzioni, elencare test che falliscono/passano

- Esempio: SimpleExamples/AgeValidatorTest

Misurare la Copertura dei Test

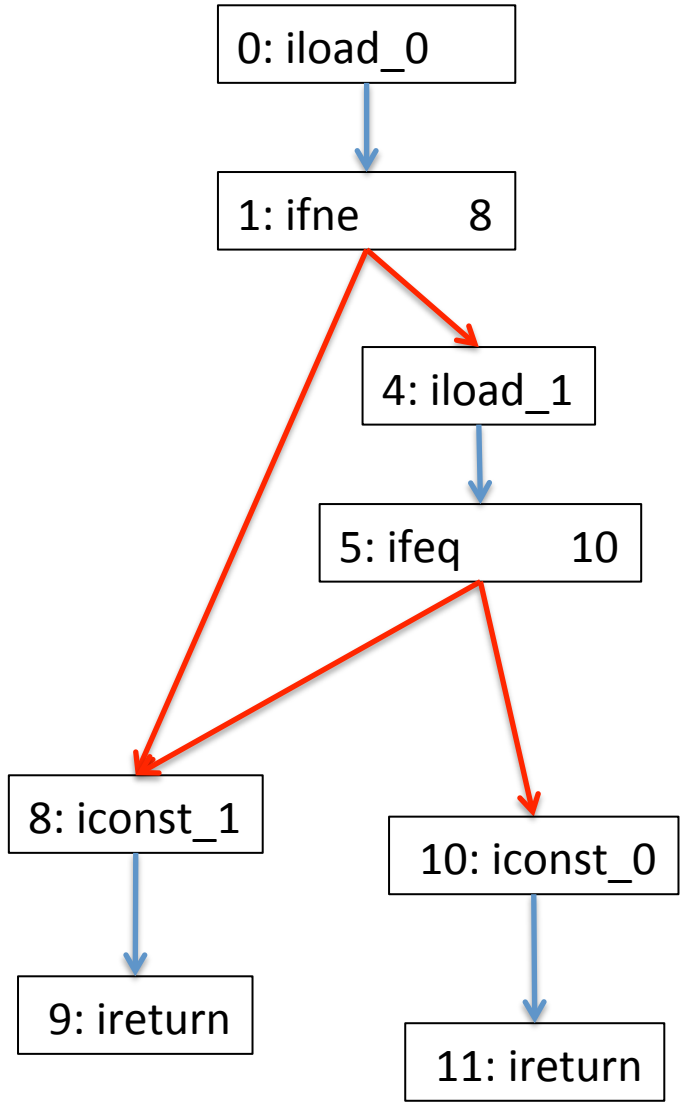
- Strumenti che registrano informazioni sulle istruzioni coperte a runtime
  - basati su instrumentazione / aggiunta di istruzioni ad hoc
  - Java:
    - EclEmma: opensource (bytecode instrumentation)
    - Clover: pay (source instrumentation)
  - C/C++:
    - gcov: disponibile con gcc

EclEmma: www.eclemma.org

- Plug-in di Eclipse basato su libreria JaCoCo
- Misure di copertura
  - instruction coverage
  - line coverage
  - branch coverage
    - considera i branch nel bytecode, granularità più fine
  - cyclomatic coverage
    - Formal definition v(G) = E - N + 2
      - **E**dges, **N**odes
    - Jacoco version v(G) = B - D + 1
      - **B**ranches, **D**ecision points

EclEmma: Bytecode Branch Coverage

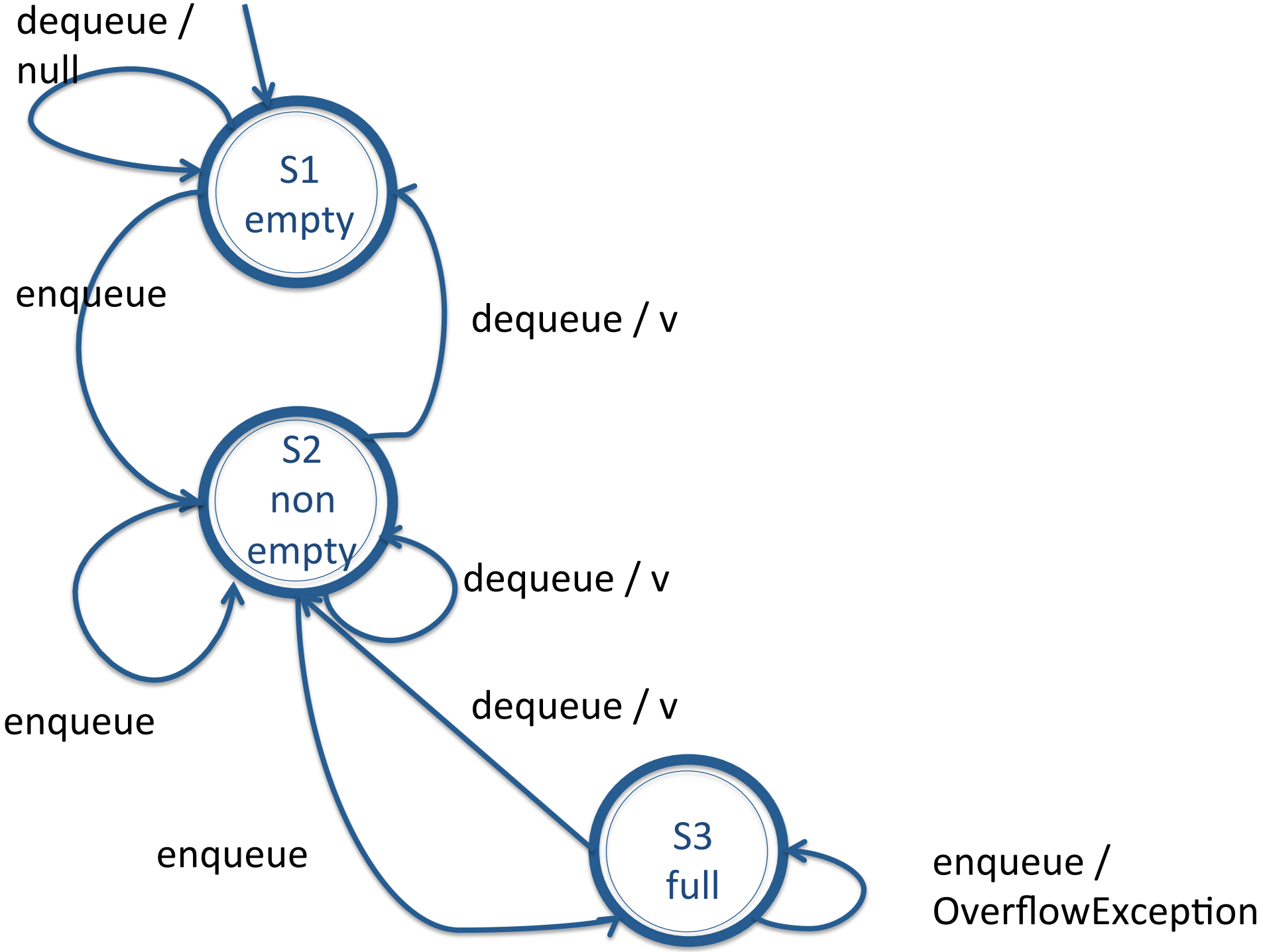
```
public static boolean or(boolean a, boolean b){
        if ( a || b ){
                return true;
        }
        return false;
}

public static boolean or(boolean, boolean);
        0: iload_0
        1: ifne         8
        4: iload_1
        5: ifeq         10
        8: iconst_1
        9: ireturn
       10: iconst_0
       11: ireturn
     LineNumberTable:
       line 21: 0
       line 22: 8
       line 24: 10
```

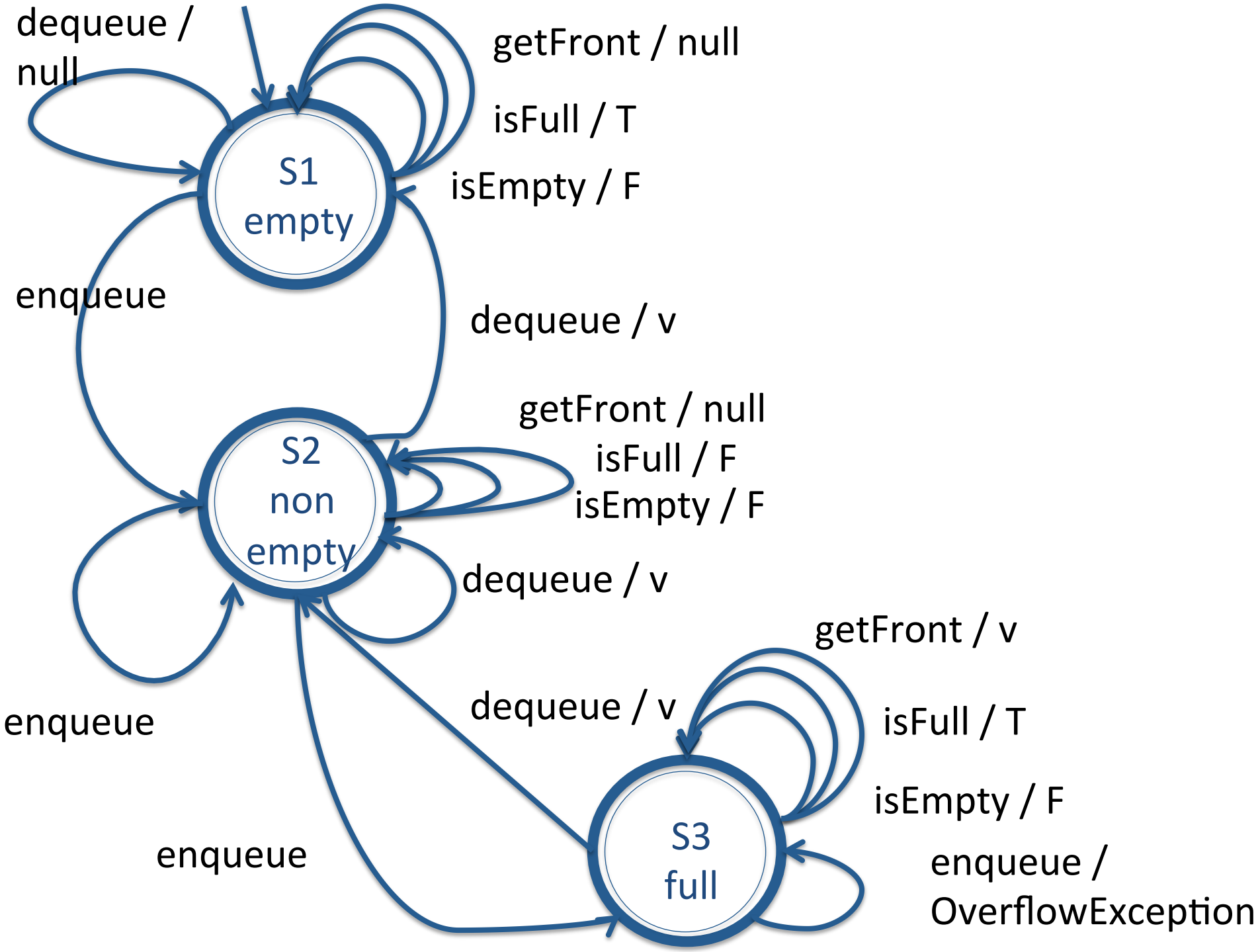


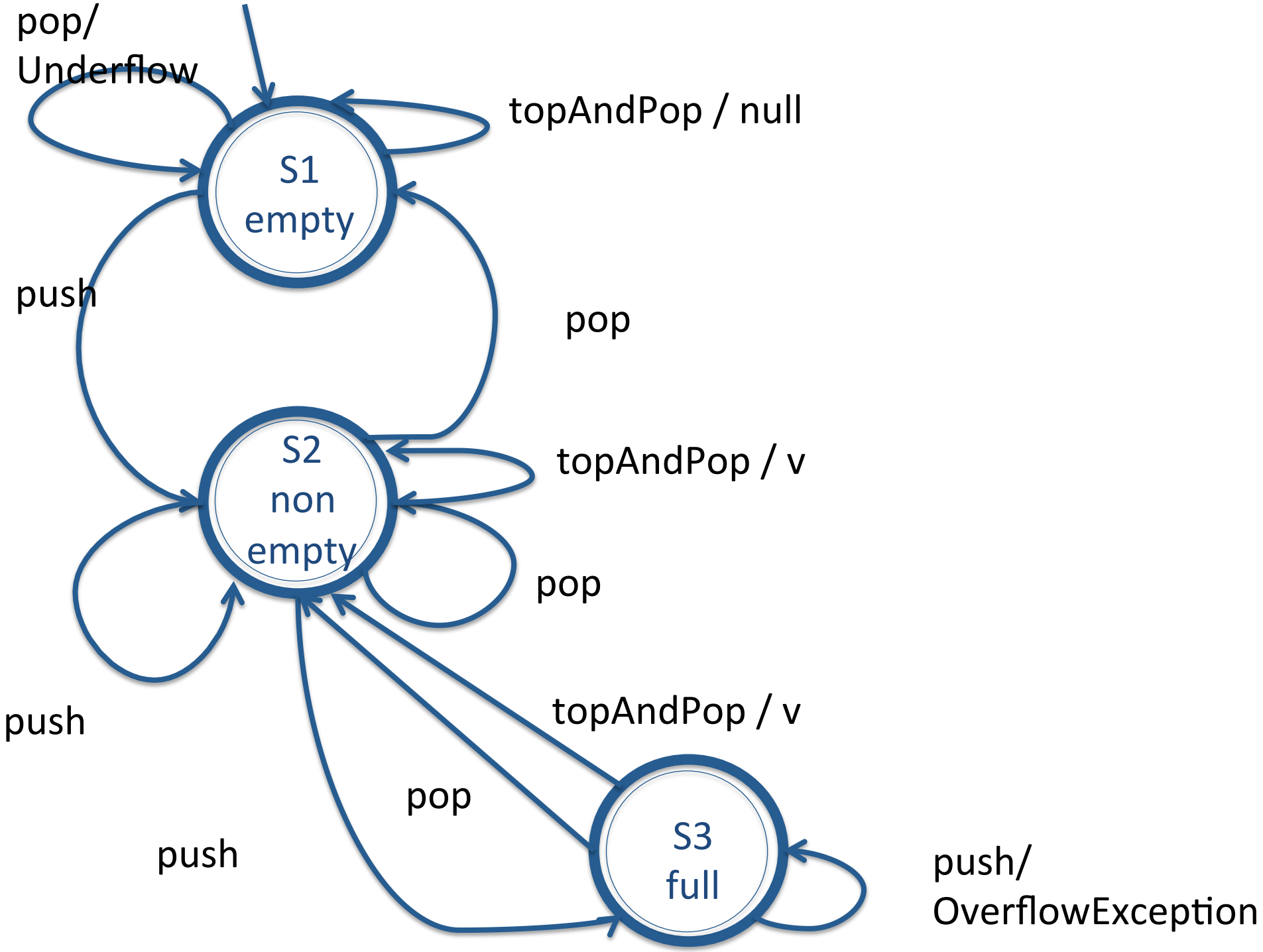
Altre Opzioni di EclEmma

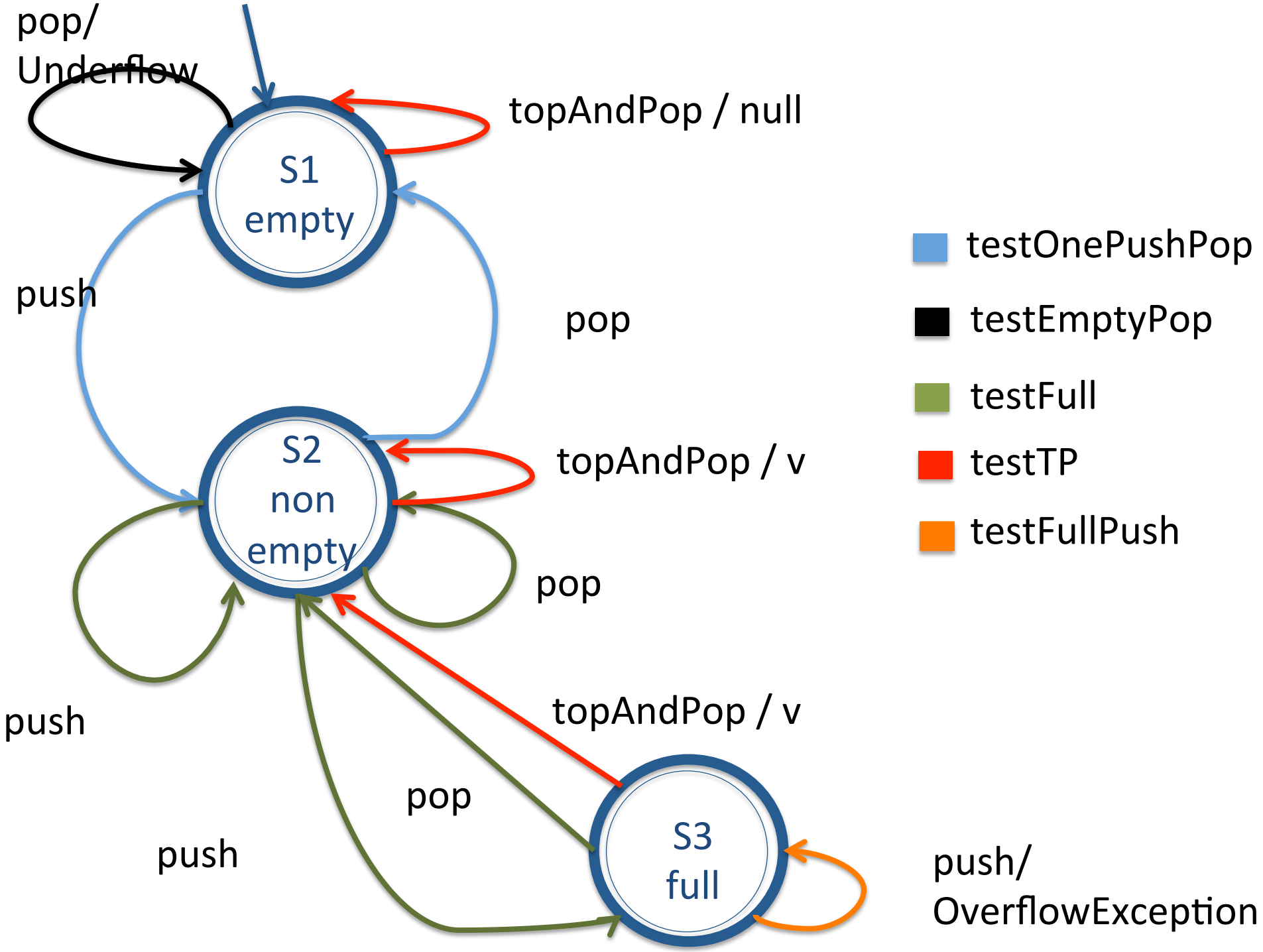
- Visualizzazione per package/class
- Unione di copertura di test run differenti

Identificazione Implicita di Test a partire da Modelli tramite Model Based Testing


dequeue /
null
S1
empty
enqueue
dequeue / v
S2
non
empty
dequeue / v
enqueue
dequeue / v
enqueue
S3
full
enqueue /
OverflowException


dequeue /
null
getFront / null
isFull / T
isEmpty / F
S1
empty
enqueue
dequeue / v
getFront / null
isFull / F
isEmpty / F
S2
non
empty
dequeue / v
getFront / v
dequeue / v
isFull / T
isEmpty / F
enqueue
enqueue
S3
full
enqueue /
OverflowException


pop/
Underflow
topAndPop / null
S1
empty
push
pop
S2
non
empty
topAndPop / v
pop
push
topAndPop / v
pop
push
S3
full
push/
OverflowException


pop/
Underflow
S1
empty
topAndPop / null
push
pop
S2
non
empty
topAndPop / v
pop
push
topAndPop / v
pop
push
S3
full
push/
OverflowException
testOnePushPop
testEmptyPop
testFull
testTP
testFullPush

Tool per Generare Test in Automatico

- http://graphwalker.org

Identificazione Implicita di Test a partire **da Modelli** tramite **Model Based Testing**

Identificazione Implicita di Test
a partire **da Modelli** tramite
Model Based Testing

Combinazione Esplicita di
Valori Significativi
tramite
Approcci Combinatori

Category Partition

Category partition (manual steps)

1. Decompose the specification into independently testable features
   - for each **feature** identify
     - **parameters**
     - **environment elements**
   - for each parameter and environment element identify **elementary** characteristics (**categories**)
2. Identify relevant values
   - for each characteristic (category) identify (**classes of**) **values**
     - normal values
     - boundary values
     - special values
     - error values
3. Introduce constraints

Esercizio

- javadoc/org/apache/commons/math/utils/TranformerMap.html

Category Partition, esempio banale, generiamo i test per semplice una form di ricerca

- Parameter form
  - Nome da cercare
    - nome vuoto [single]
    - nome lunghissimo [error]
    - nome di prodotto nel DB [if !empty]
    - nome di prodotto non nel DB
- Environment
  - No elementi nel Database
    - zero [property empty]
    - 1 [single]
    - N

Category Partition con TSL

- Strumento per la generazione a partire da file di testo visti a lezione
  - /opt/categoryPartition/tsl
- Documentazione
  - /opt/categoryPartition/TSL.pdf

Esercizio

- Generare casi di test di unità adottando category partition per il metodo TransformerMap.transform(Object)

TransformersMap: esempio d'uso

```
public class ListTransformer implements NumberTransformer, Serializable {
        public double transform(Object o) throws MathException{
                if ( ! (o instanceof List) )
                        throw new MathException("Object is not List");
                return ((List)o).size();
        }
}
```

```
public class CollectionsSizeCounter{
        TransformerMap tm = new TransformerMap();

        public CollectionsSizeCounter(){
                NumbersTransformer lt = new ListTransformer();
                tm.putTransformer(ArrayList.class,lt);
                tm.putTransformer(LinkedList.class, lt);
                tm.putTransformer(HashMap.class, new MapTransformer());
                //atri transformer potrebbero seguire
        }

        public void size(Object collection) {
                return tm.transform(list)
        }
```

TransformersMap: esempio di test

```
public class ListTransformer implements NumberTransformer, Serializable {
	public double transform(Object o) throws MathException{
		return 2.0;
	}
}
```

```
public class TransformerMapTest {
	@Test
	public void test () throws MathException {
		TransformerMap tm = new TransformerMap();
		tm.putTransformer(ArrayList.class, new ListTransformer());
		ArrayList<Object> list = new ArrayList<Object>();
		Double result = tm.transform(list)
		assertEquals( 2.0, result , 0.001 );
	}
}
```

Pairwise Testing

Testare Display Control

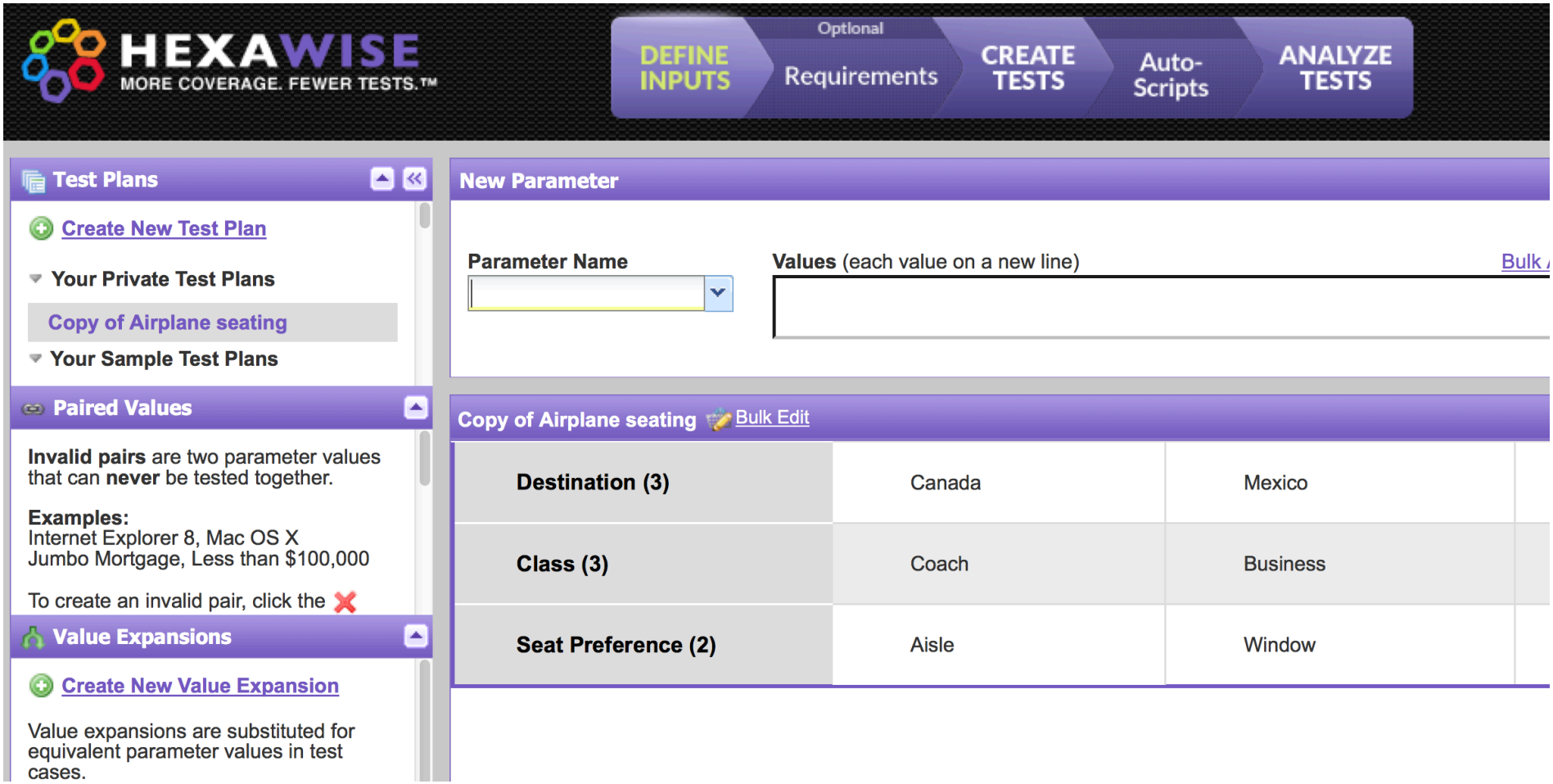
Supponiamo che presenti 5 parametri di configurazione

Display Mode	Language	Fonts	Color	Screen size
full-graphics	English	Minimal	Monochrome	Hand-held
text-only	French	Standard	Color-map	Laptop
limited-bandwidth	Spanish	Document-loaded	16-bit	Full-size
	Portuguese		True-color	

Pairwise combinations: 17 test cases

Language	Color	Display Mode	Fonts	Screen Size
English	Monochrome	Full-graphics	Minimal	Hand-held
English	Color-map	Text-only	Standard	Full-size
English	16-bit	Limited-bandwidth	-	Full-size
English	True-color	Text-only	Document-loaded	Laptop
French	Monochrome	Limited-bandwidth	Standard	Laptop
French	Color-map	Full-graphics	Document-loaded	Full-size
French	16-bit	Text-only	Minimal	-
French	True-color	-	-	Hand-held
Spanish	Monochrome	-	Document-loaded	Full-size
Spanish	Color-map	Limited-bandwidth	Minimal	Hand-held
Spanish	16-bit	Full-graphics	Standard	Laptop
Spanish	True-color	Text-only	-	Hand-held
Portuguese	-	-	Monochrome	Text-only
Portuguese	Color-map	-	Minimal	Laptop
Portuguese	16-bit	Limited-bandwidth	Document-loaded	Hand-held
Portuguese	True-color	Full-graphics	Minimal	Full-size
Portuguese	True-color	Limited-bandwidth	Standard	Hand-held

Pairwise Testing Tool
www.hexawise.com
create un account su questo sito



Funzionalità di Hexawise

- Definizione parametri / valori
- Definizione di vincoli
  - combinazioni non valide + married pairs
- Definizioni requisiti
  - combinazioni di valori che devono essero coperte per motivi strategici
- Visualizzazione della copertura delle coppie
  - prioritizzazione dei test
- Esportazione dei test script
  - è possibile specificare l'output atteso per alcune combinazioni di input
  - spesso i "valori" rappresentano classi di valori, possibile inserire valori concreti da suggerire in autmatico negli script per i tester

Adding constraints

- Simple constraints

  example: color monochrome not compatible with screen laptop and full size

  can be handled by considering the case in separate tables