[1]University of Geneva, Switzerland
[2]CERN, Geneva, Switzerland

Automated Test Case Generation for CTRL using Pex: Lessons Learned

Stefan Klikovits [1,2]

David PY Lawrence [1]
Manuel Gonzalez-Berges [2]
Didier Buchs [1]

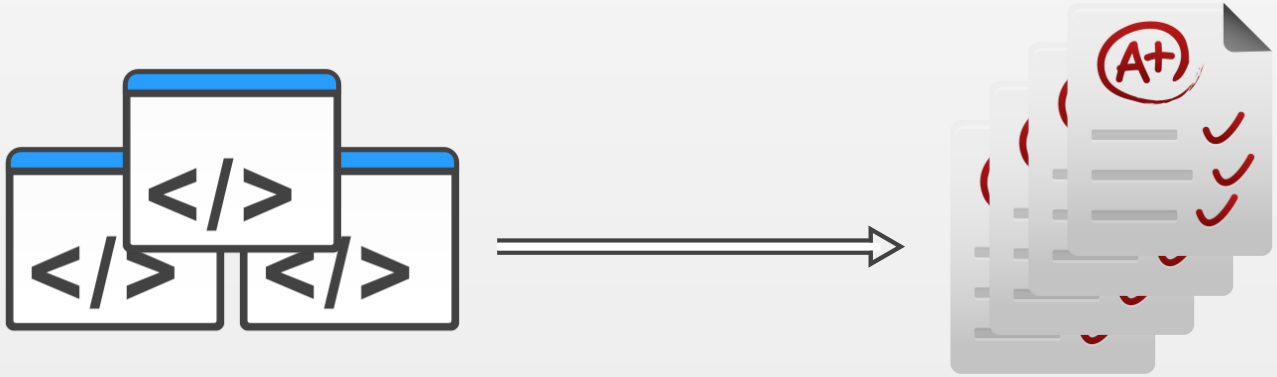
s.k@cern.ch

What are we doing?

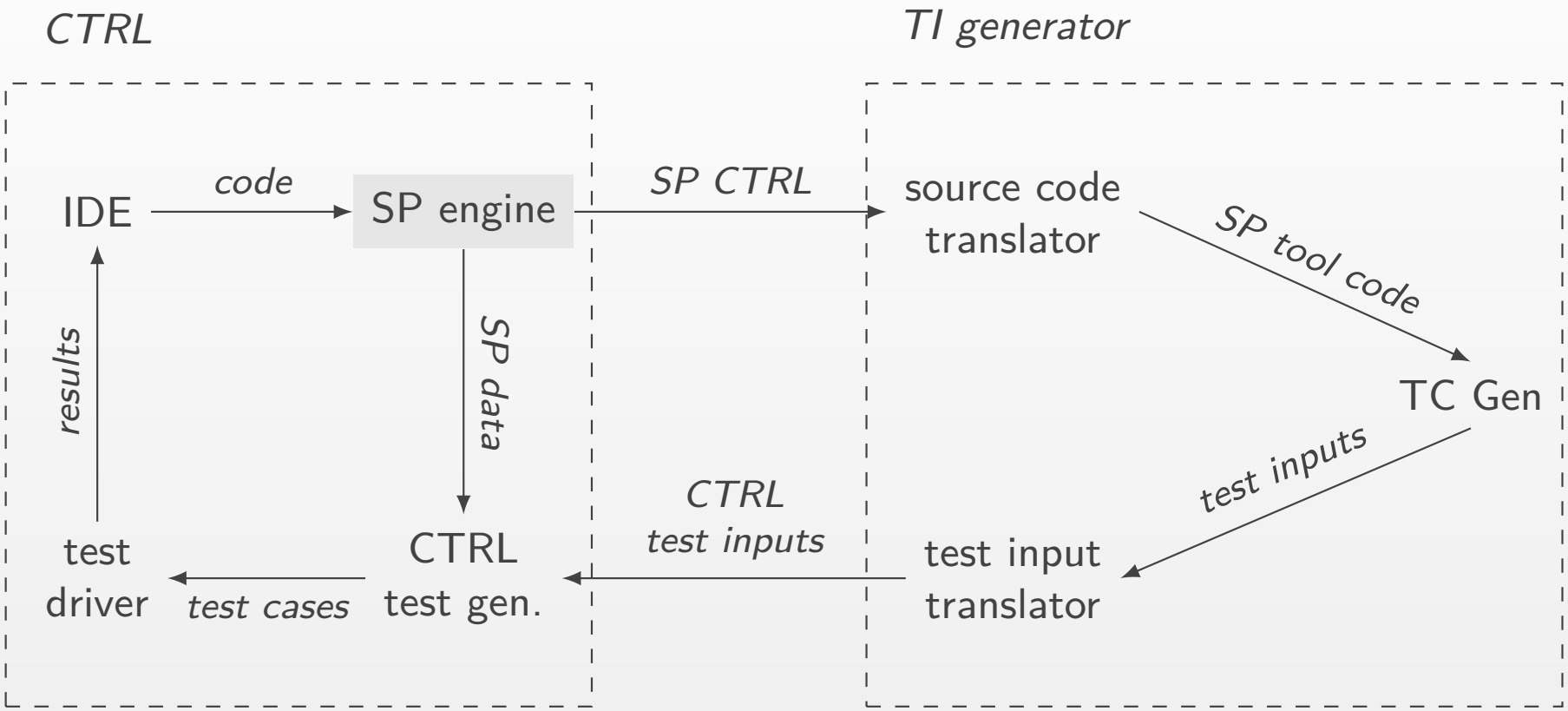
- 1 MLOC code
- no automated unit testing until two years ago
- frequent changes in execution environment
- (mostly) manual verification
- big expenses (time) on QA side

What are we doing?

- 1 MLOC code
- no automated unit testing until two years ago
- frequent changes in execution environment
- (mostly) manual verification
- big expenses (time) on QA side

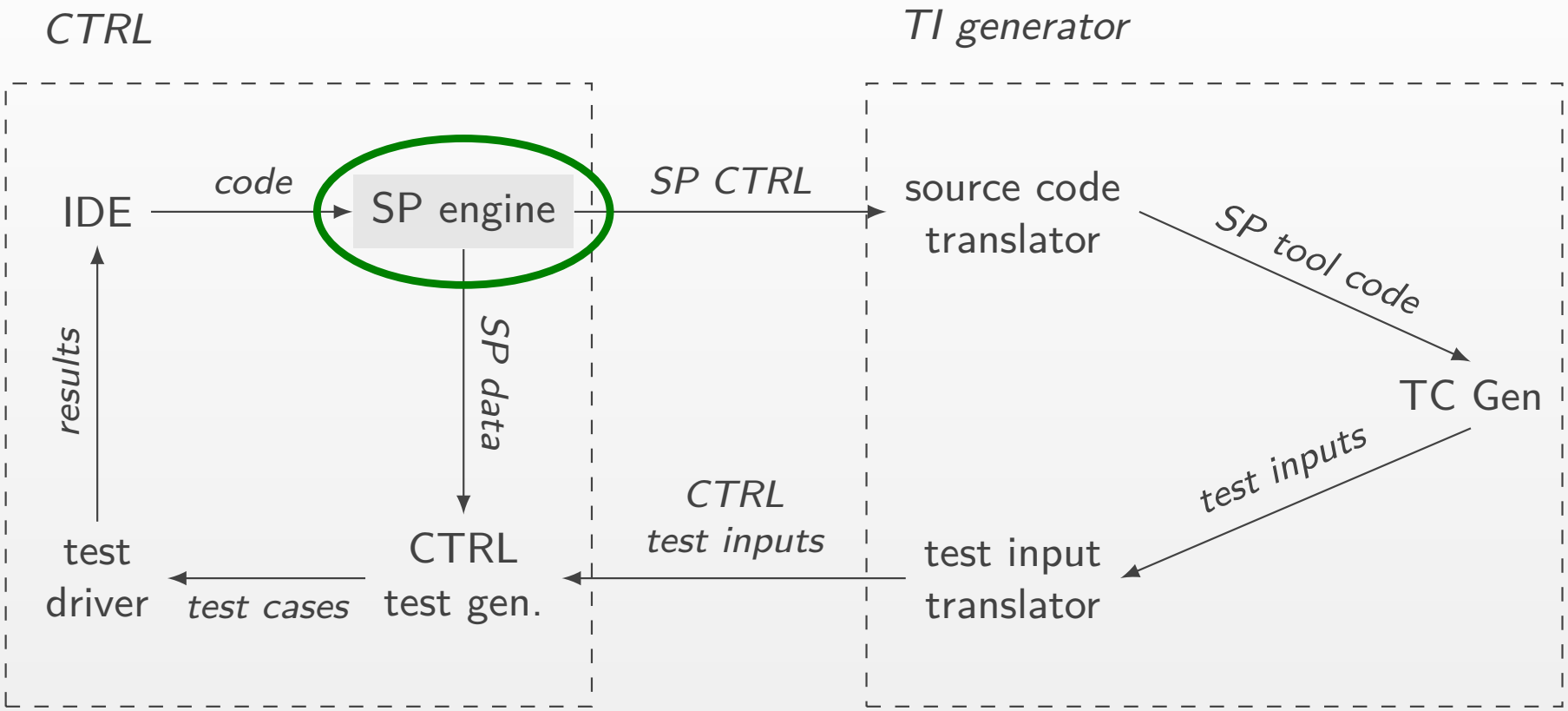


How are we doing it?



ITEC workflow

How are we doing it?



ITEC workflow

Considering Execution Environment Resilience: A White-Box Approach

Klikovits et. al. SERENE 2015

Recap semi-purification

- replace dependencies with parameters

```
f(x){
  if GLOBAL_VAR:
    return dbGet(x)
  else:
    return -1
}
```

A non-pure function

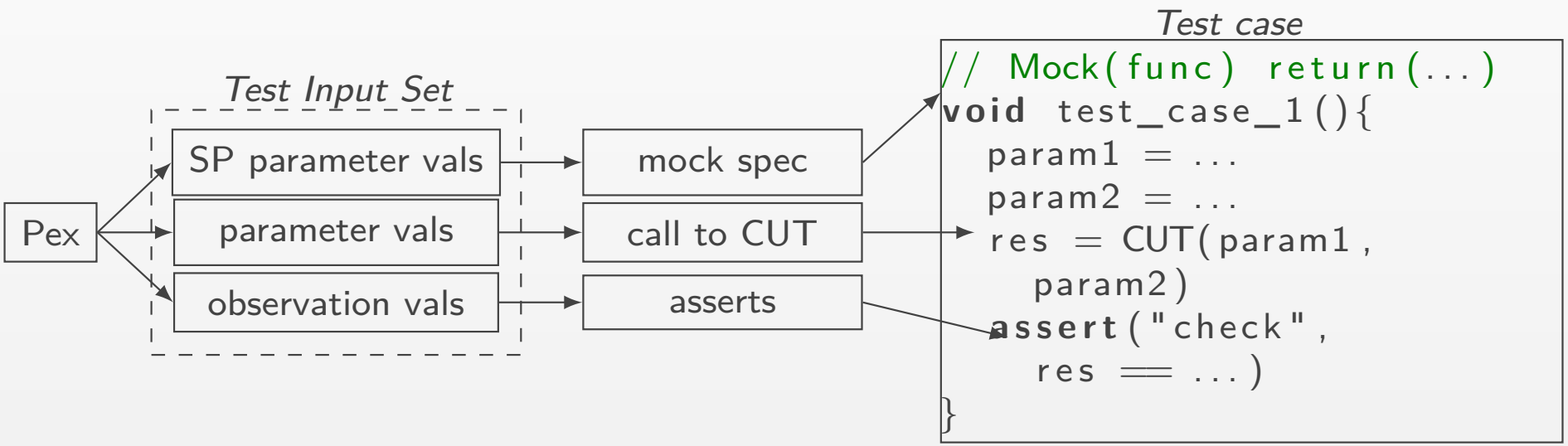
```
f_sp(x,a,b){
  if a:
    return b
  else:
    return -1
}
```

Semi-purified `f(x)`

```
test_f_sp(){
  x = f("test",True,5) //act
  assert(x == 5) //assert
}
```

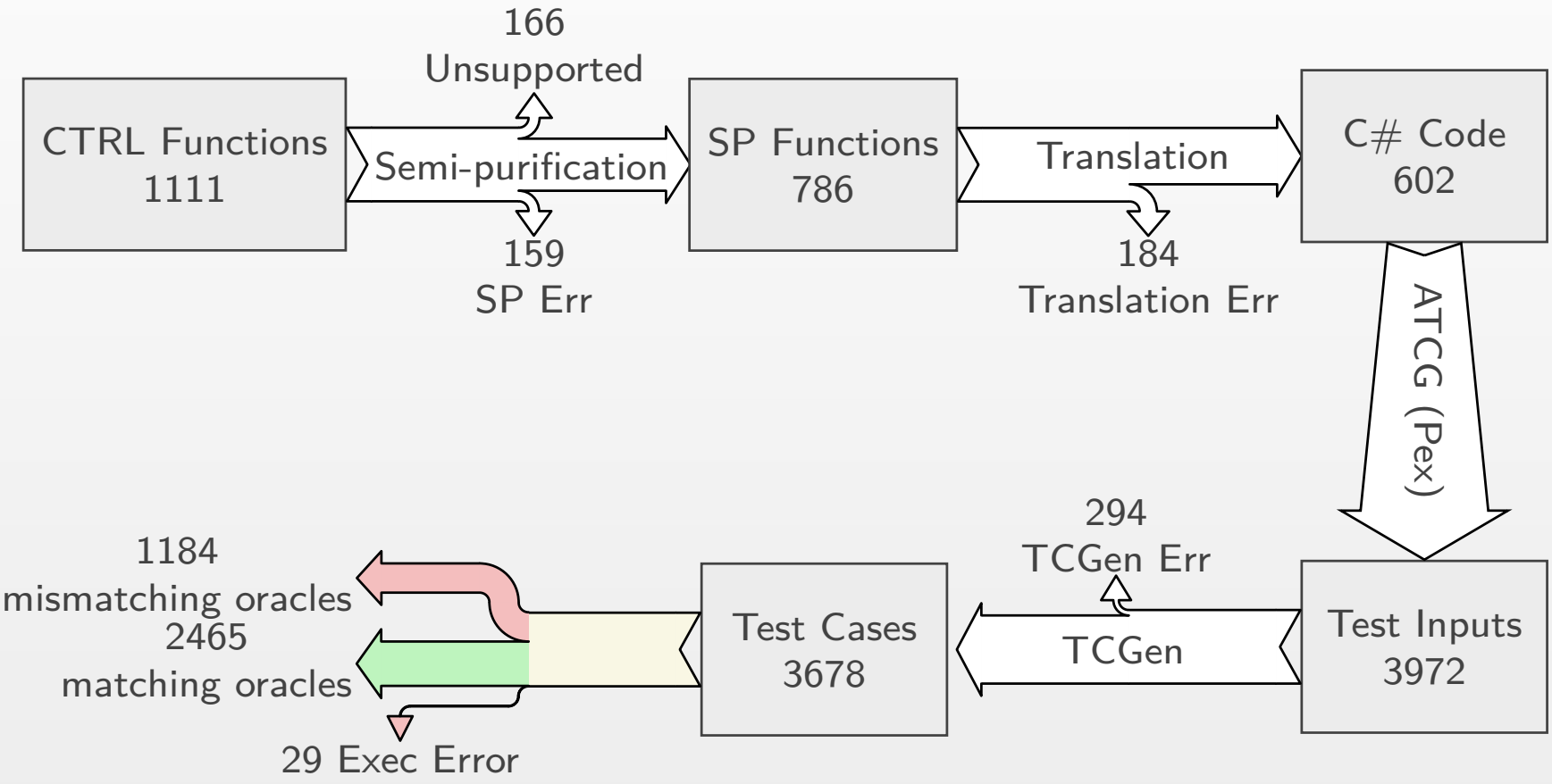
Test case

From Pex to test cases

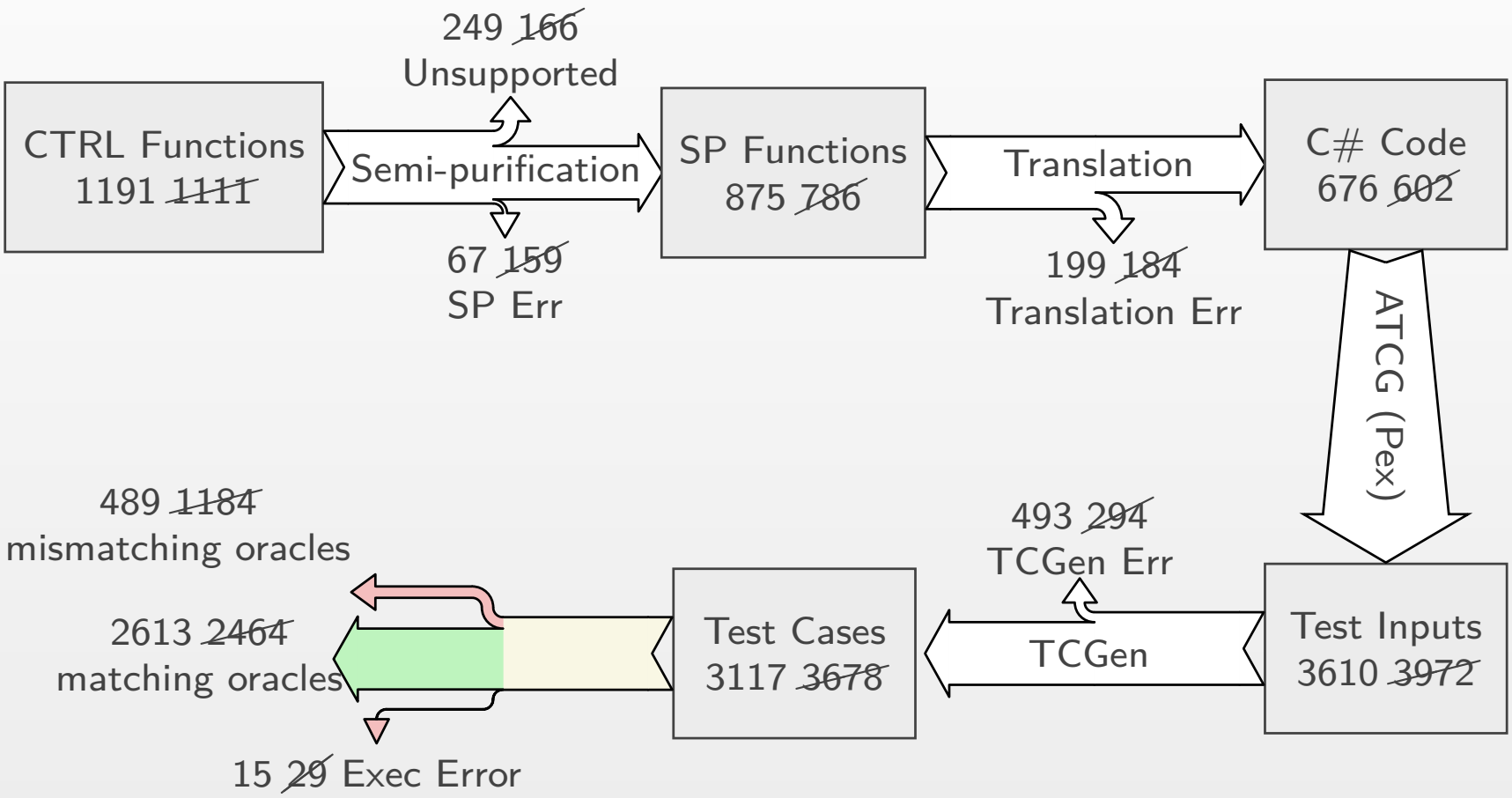


Test case generation from Pex output

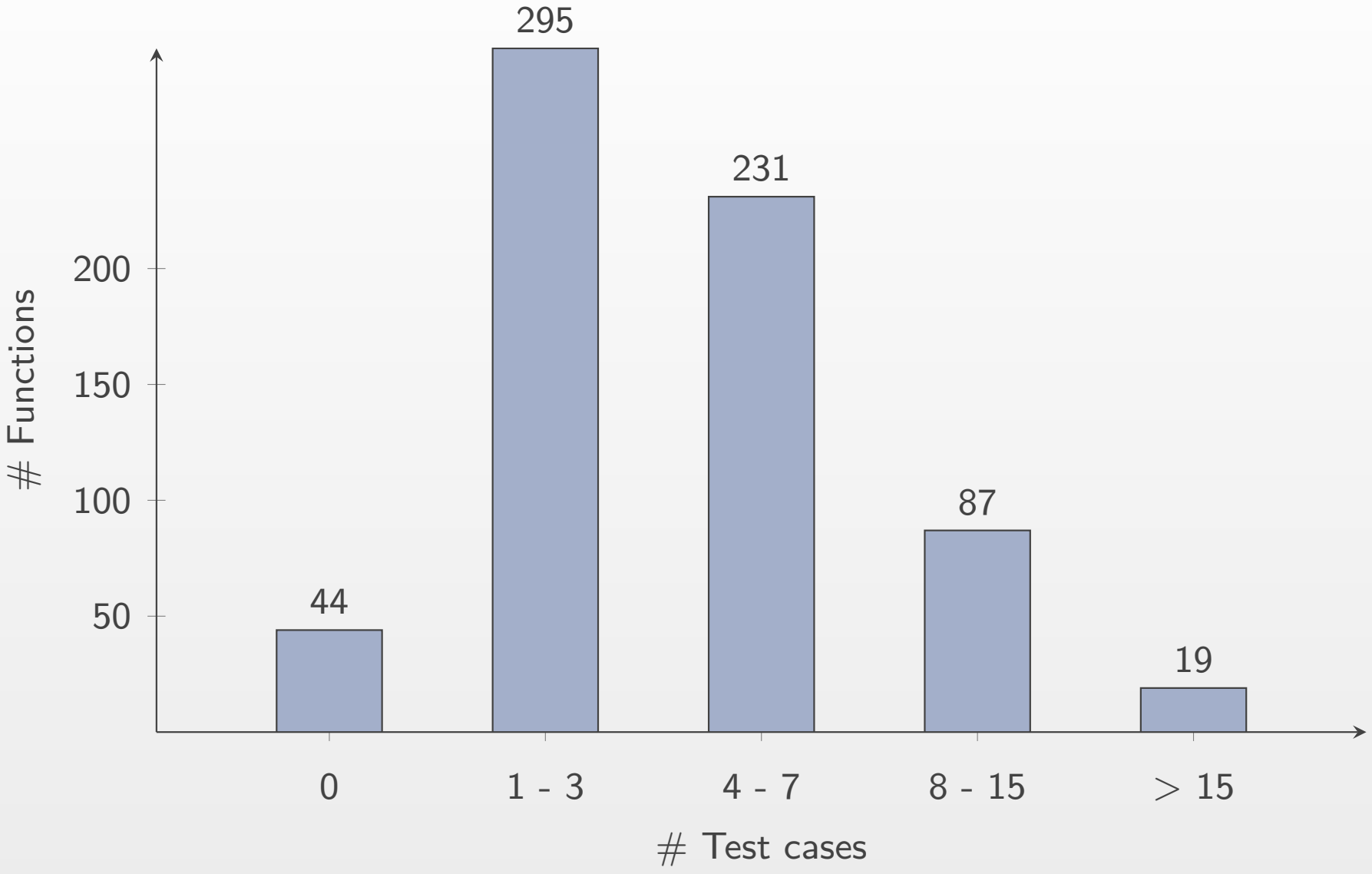
Test case generation: results



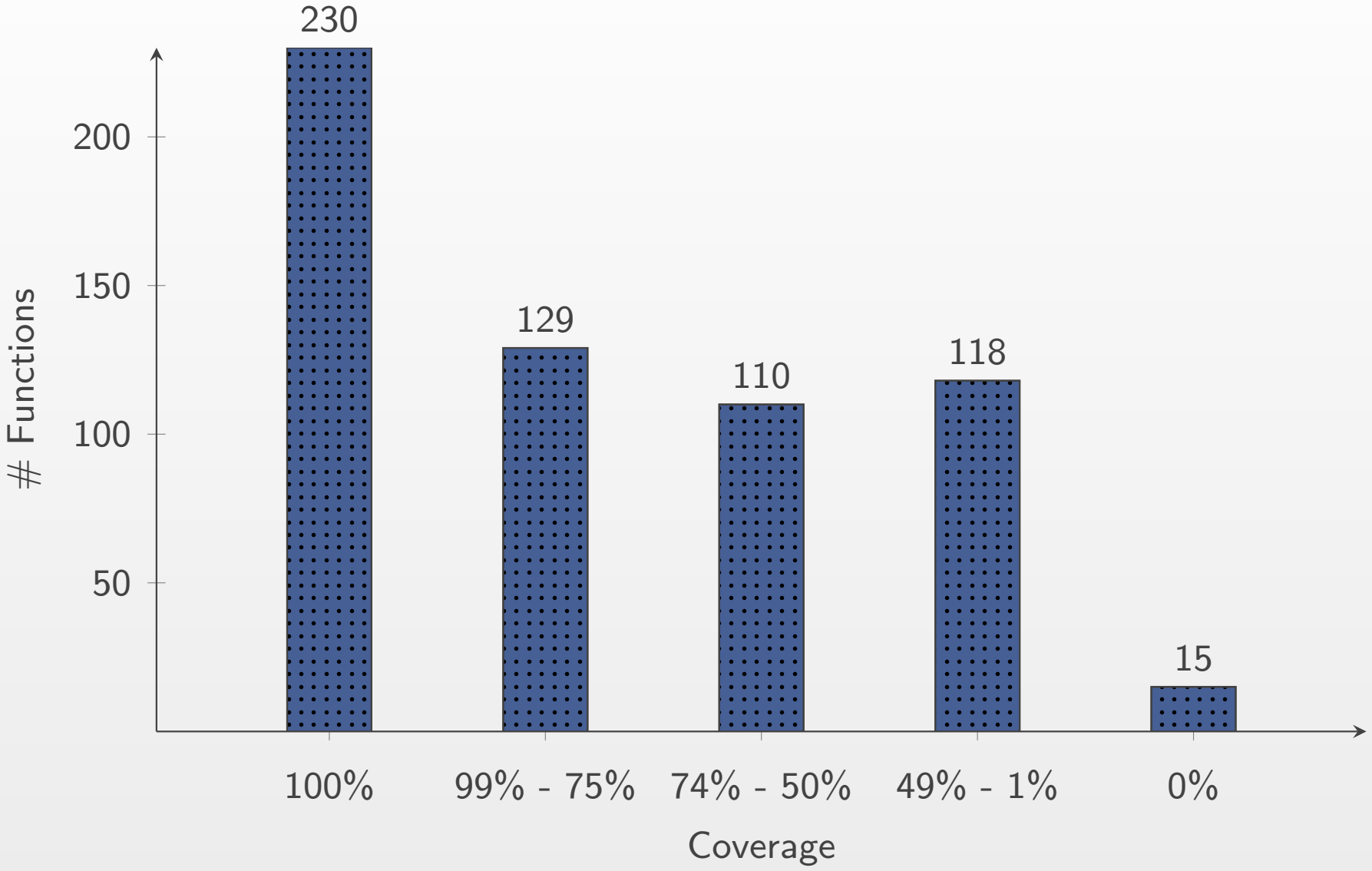
Test case generation: update



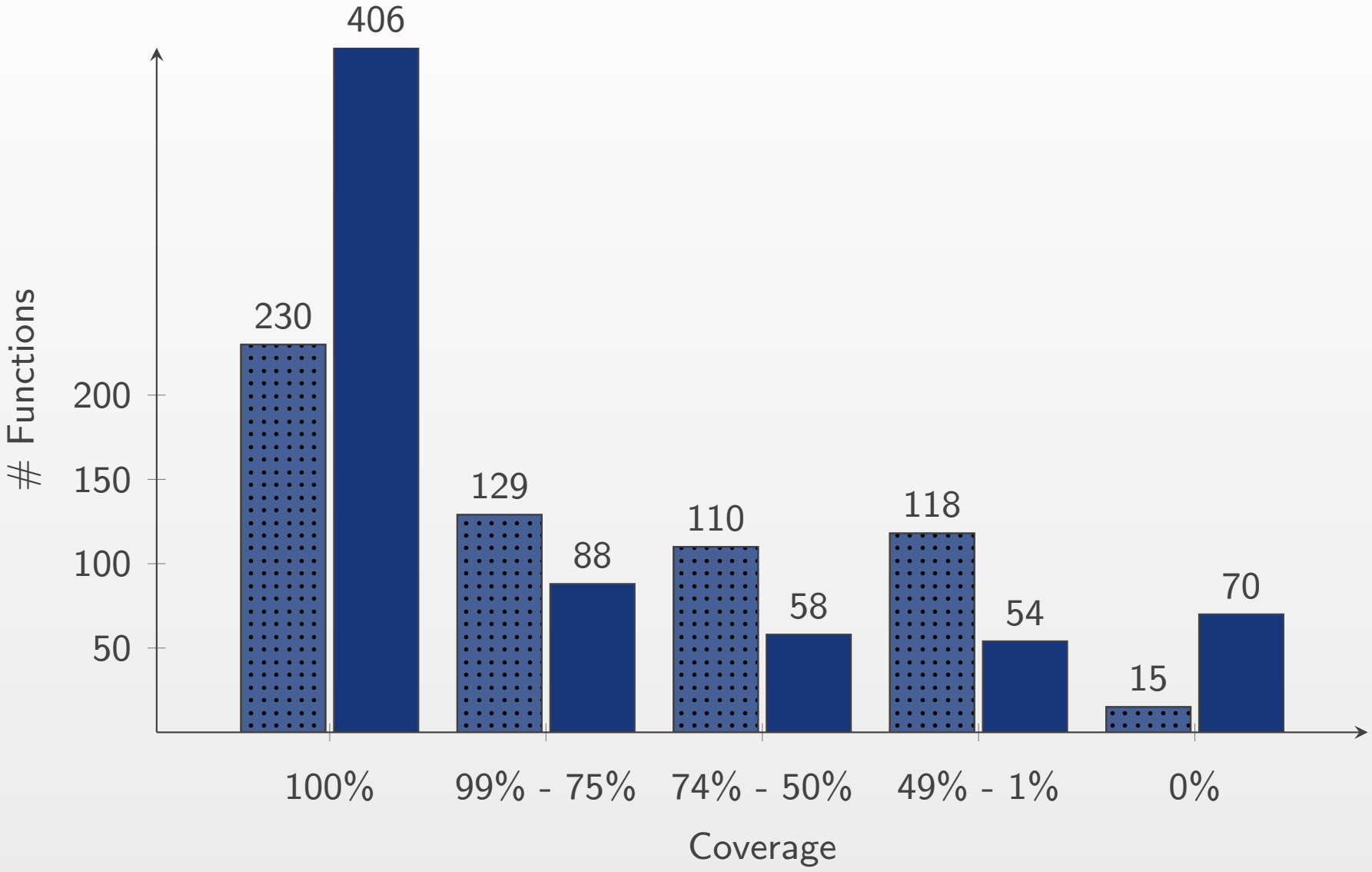
Number of test cases



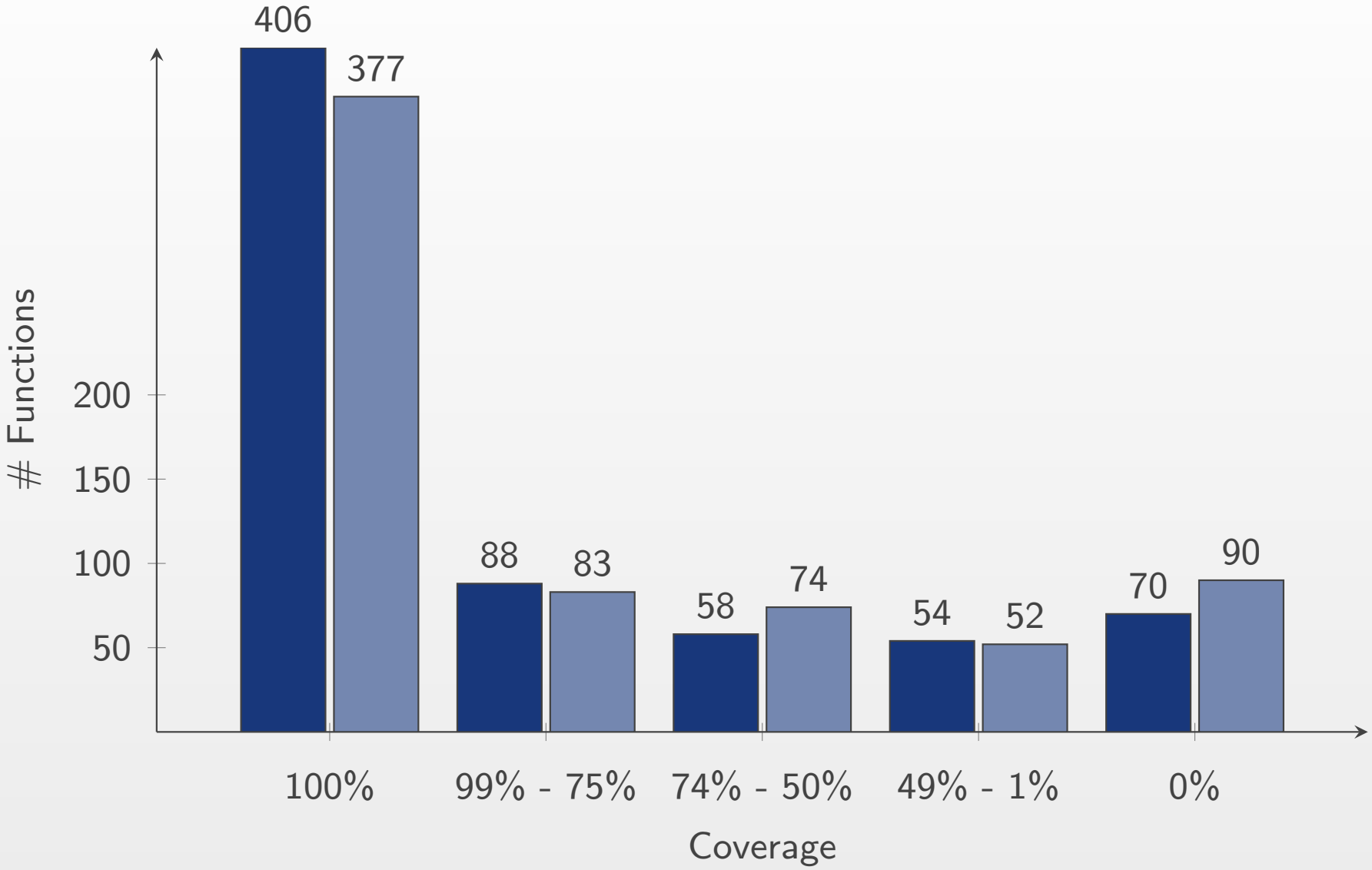
Coverages


230
129
110
118
15
200
150
100
50
Functions
100%
99% - 75%
74% - 50%
49% - 1%
0%
Coverage

Coverages: updated



Coverages: matching oracles



Lessons learned

- not everything can be translated (easily)

Lessons learned

- not everything can be translated (easily)
- not all features should be supported

Lessons learned

- not everything can be translated (easily)
- not all features should be supported
- C# is no silver bullet

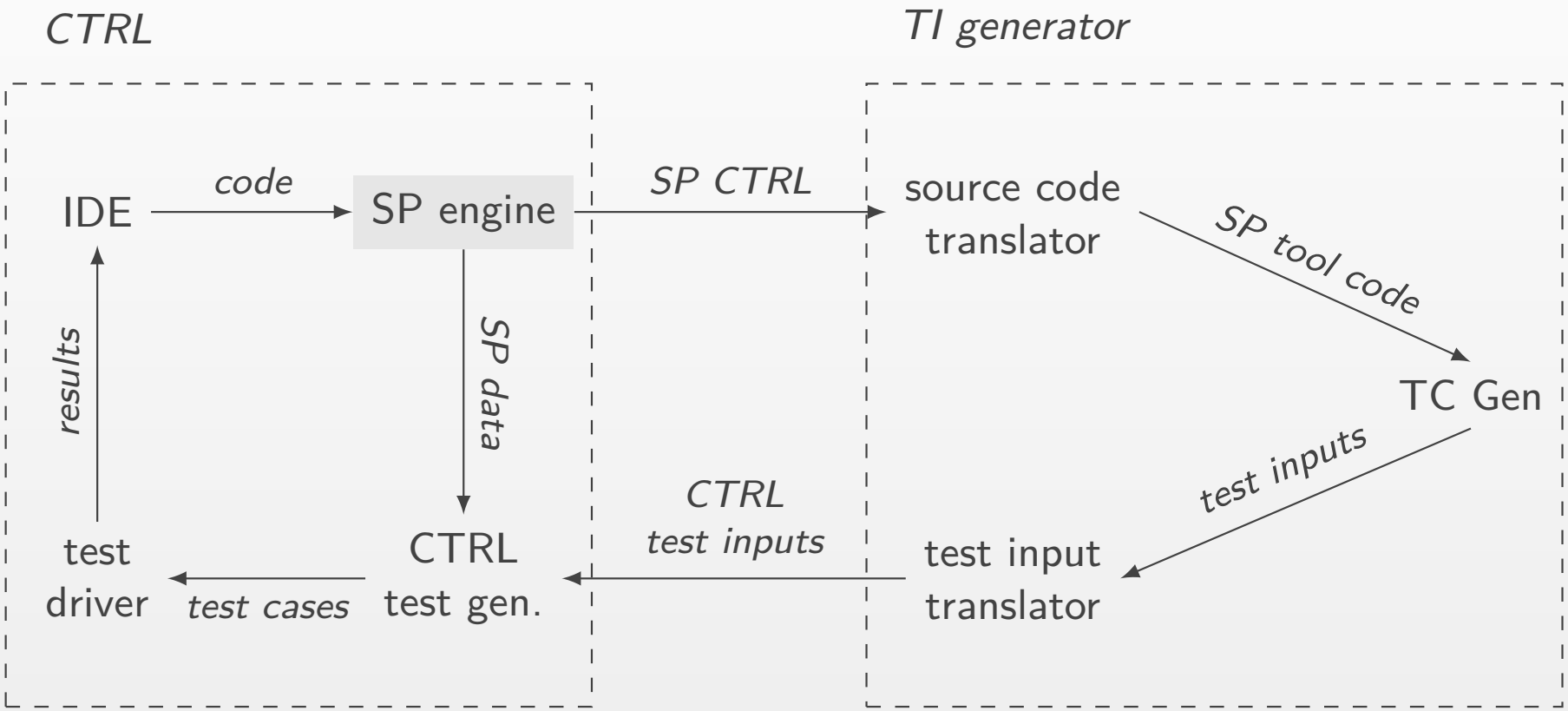
Lessons learned

- not everything can be translated (easily)
- not all features should be supported
- C# is no silver bullet
- improving the quality of test cases ?

Lessons learned

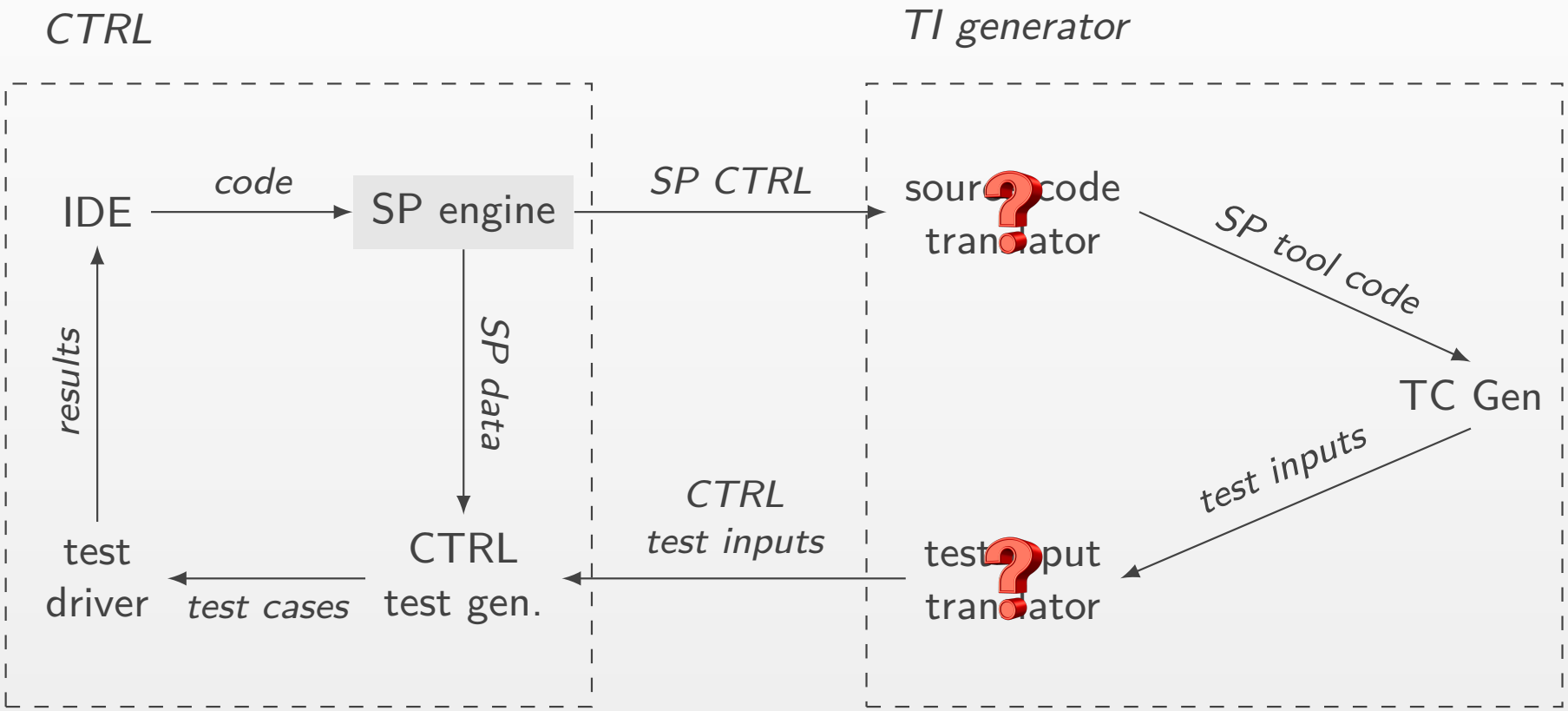
- not everything can be translated (easily)
- not all features should be supported
- C# is no silver bullet
- improving the quality of test cases ?
- tools have “features”

How are we doing it?



ITEC workflow

How are we doing it?



ITEC workflow

Why test the translation?

- DySyEx: execute code, cover max. paths
- small differences – big impacts

Why test the translation?

- DySyEx: execute code, cover max. paths
- small differences – big impacts

http://samcnitt.tumblr.com/

Why not test the translator?

- No formal semantics?
- Changing language?
- Effort to adapt the solution?

Why not test the translator?

- No formal semantics?
- Changing language?
- Effort to adapt the solution?

http://asterix.wikia.com/wiki/Asterix_and_Cleopatra

How to test?

How to test?

Divide

http://chapleau.us/Img/caesar_asterix.gif

How to test?

Divide

http://chapleau.us/Img/caesar_asterix.gif

Anonymise

https://www.youtube.com/watch?v=UF6E-4G4n_M

How to test?

Divide

http://chapleau.us/Img/caesar_asterix.gif

Anonymise

https://www.youtube.com/watch?v=UF6E-4G4n_M

Analyse Blocks

https://en.gamigo.com/game/asterix

How to test?

Divide

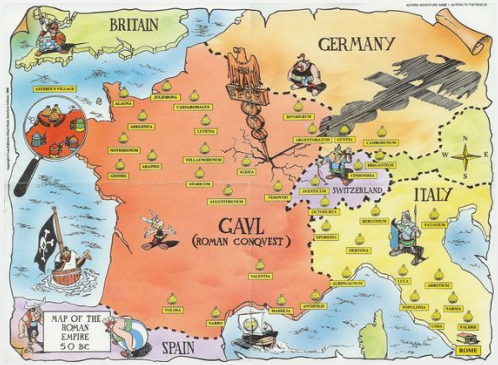
http://chapleau.us/Img/caesar_asterix.gif

Anonymise

https://www.youtube.com/watch?v=UF6E-4G4n_M

Analyse Blocks

https://en.gamigo.com/game/asterix



Conquer

https://www.pinterest.com/pin/336784878358770673/

How to test?

```
int func(int a, int b) {
  a++
  a++
  b = b+2
  if(a > b){
    return a % b
  } else {
    return a + b
  }
}
```

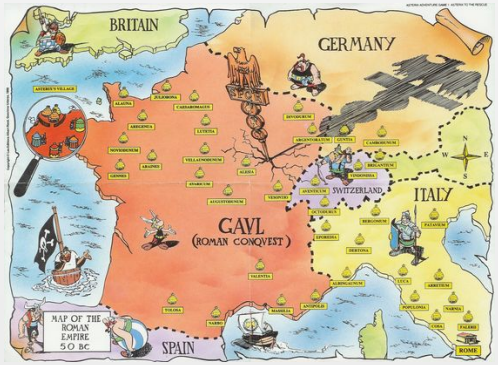
Divide

Anonymise

https://www.youtube.com/watch?v=UF6E-4G4n_M

Analyse Blocks

https://en.gamigo.com/game/asterix



Conquer

https://www.pinterest.com/pin/336784878358770673/

How to test?

```
int func(int a, int b) {
  a++
  a++
  b = b+2
  if(a > b){
    return a % b
  } else {
    return a + b
  }
}
```

Divide

```
int func(int, int){
  int++
  int++
  int = int + int
  if(int > int) {
    return int % int
  } else {
    return int + int
  }
}
```

Anonymise

Analyse Blocks

https://en.gamigo.com/game/asterix

Conquer

https://www.pinterest.com/pin/336784878358770673/

How to test?

```
int func(int a, int b) {
  a++
  a++
  b = b+2
  if(a > b){
    return a % b
  } else {
    return a + b
  }
}
```

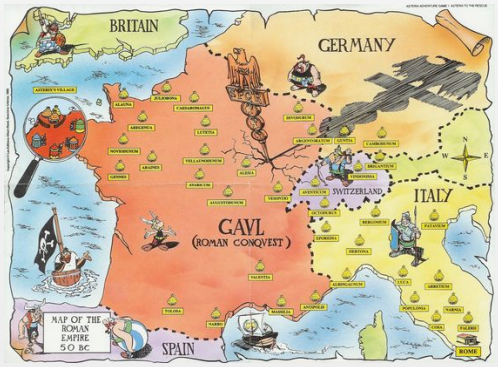
Divide

```
int func(int , int){
  int++
  int++
  int = int + int
  if(int > int) {
    return int % int
  } else {
    return int + int
  }
}
```

Anonymise

```
int func(int a, int b) {
  int++                    1
  int++                    1
  int = int+int            1
  if(int > int){           0
    return int % int       0
  } else {
    return int + int       1
  }
}
```

Analyse Blocks



Conquer

https://www.pinterest.com/pin/336784878358770673/

How to test?

```
int func(int a, int b) {
  a++
  a++
  b = b+2
  if(a > b){
    return a % b
  } else {
    return a + b
  }
}
```

Divide

```
int func(int, int){
  int++
  int++
  int = int + int
  if(int > int) {
    return int % int
  } else {
    return int + int
  }
}
```

Anonymise

```
int func(int a, int b) {
  int++                    1
  int++                    1
  int = int+int            1
  if(int > int){           0
    return int % int       0
  } else {
    return int + int       1
  }
}
```

Analyse Blocks

$$\phi = \frac{\sum \phi(L_i)}{|L|}$$

Conquer

Anonymisation & basic blocks

- Equivalence classes
- exhaustive testing of features
- confidence in basic blocks
- calculate confidence for CUT

The quality metric

For translated source code

$$conf(c) = \frac{\sum_{i=1}^{n} \phi(anon_i)}{n}, anon_i \in \{Anon\}$$

The quality metric

For translated source code

$$conf(c) = \frac{\sum_{i=1}^{n} \phi(anon_i)}{n}, anon_i \in \{Anon\}$$

For test cases

$$conf_{tc}(c, \sigma) = \prod_{\forall s_i \in S|_\sigma} \phi(anonymize(s_i))$$

Example calculation

```
1  int func(int a, int b) {
2    int++                    1
3    int++                    1
4    int = int+int            1
5    if(int > int){           0
6      return int % int       0
7    } else {
8      return int + int       1
9    }
10 }
```

$$\phi(func) = \frac{1_{L2} + 1_{L4} + 0_{L5} + 0_{L6}}{4} = 0.5$$

Example calculation

```
1  int func(int a, int b) {
2    a++                          1
3    a++                          1
4    b = b+2                      1
5    if(a > b){                   0
6      return a  % b              0
7    } else {
8      return a + b               1
9    }
10 }
```

$$\phi(func) = \frac{1_{L2} + 1_{L4} + 0_{L5} + 0_{L6}}{4} = 0.5$$

$$\phi(func, \langle 3, 5 \rangle) = 1_{L2} * 1_{L3} * 1_{L4} * 0_{L5} * 1_{L8} = 0$$

Lessons learned

- how to find exhaustive test suites ?

Lessons learned

- how to find exhaustive test suites ?
- granularity of basic blocks ?

Lessons learned

- how to find exhaustive test suites ?
- granularity of basic blocks ?
- there are thousands (!!) of possibilities

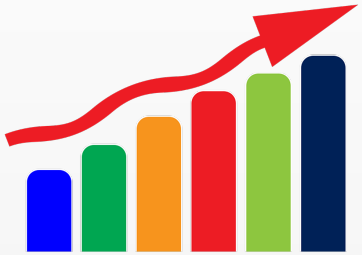
Lessons learned

- how to find exhaustive test suites ?
- granularity of basic blocks ?
- there are thousands (!!) of possibilities
- automation ?

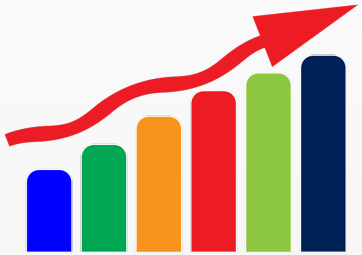
What next?

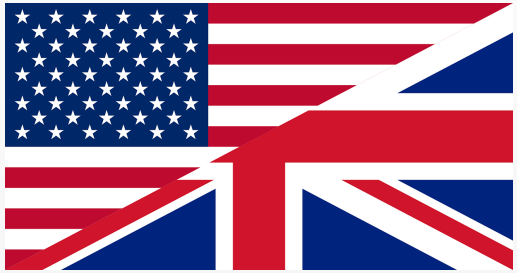
- expand TC generation
- exhaustive testing for basic blocks
- representative study for quality metric
- trade-off complexity vs. usefulness
- research unsupported features

Conclusion

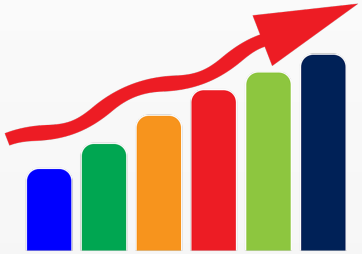


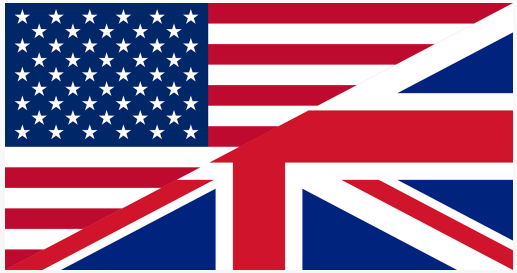
Conclusion



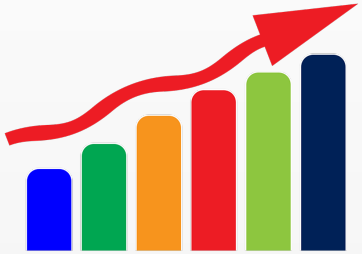


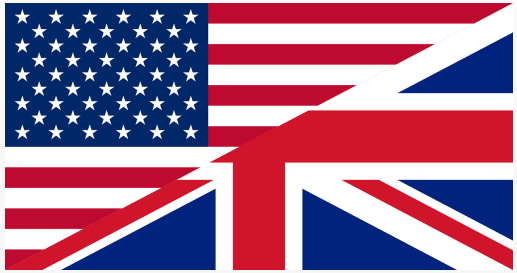
Conclusion





Conclusion






CLACK!


BRITAIN
GERMANY
SWITZERLAND
ITALY
GAUL
(ROMAN CONQUEST)
SPAIN

[1]University of Geneva, Switzerland
[2]CERN, Geneva, Switzerland

Automated Test Case Generation for CTRL using Pex: Lessons Learned

Stefan Klikovits [1,2]

David PY Lawrence [1]
Manuel Gonzalez-Berges [2]
Didier Buchs [1]

s.k@cern.ch