3.– 6. September 2012
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Treppenschach(t)

Bessere Testfälle mit ScalaCheck

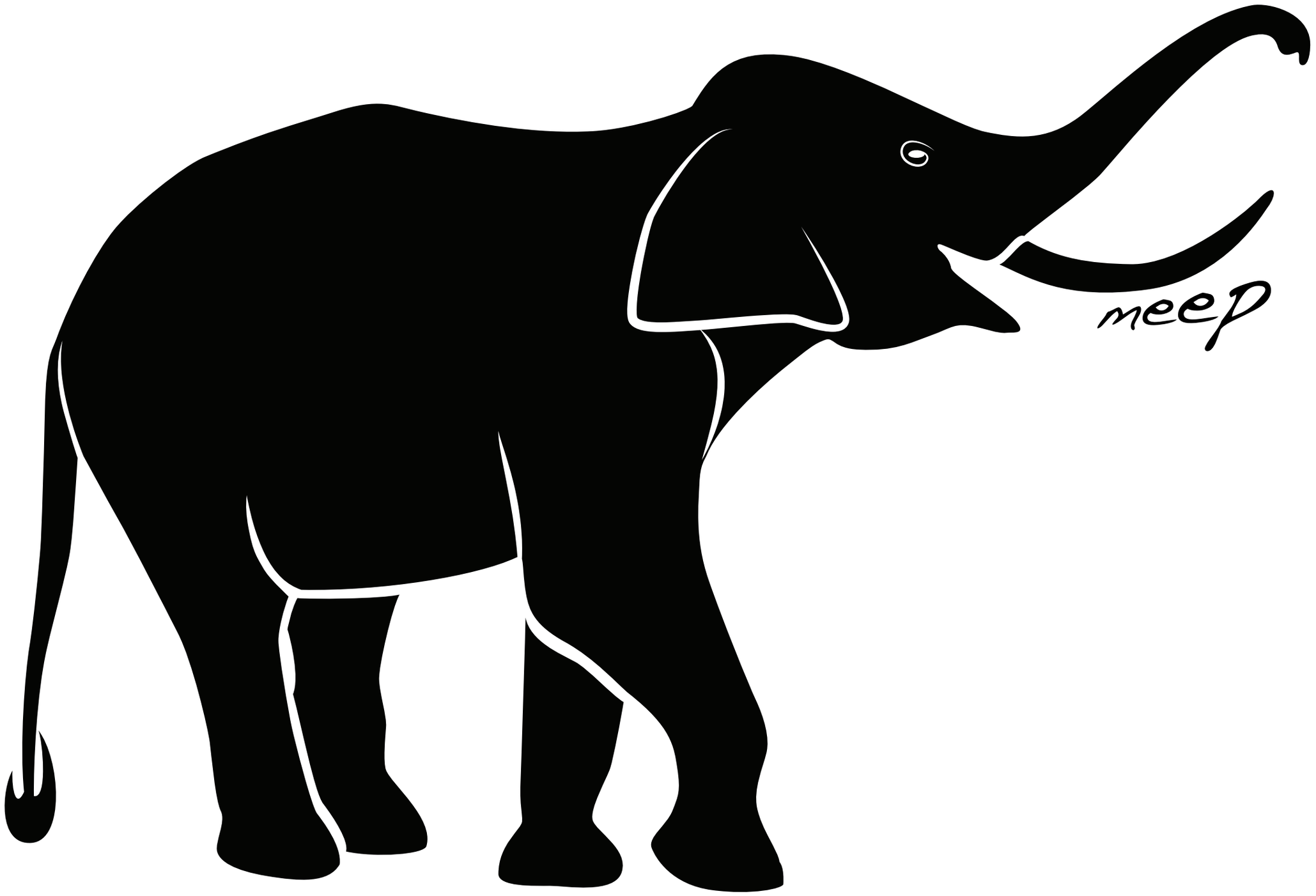
Andreas Flierl

imbus AG

@asflierl

http://blog.flierl.eu

CC BY SA



CC BY SA

CC BY SA

CC BY SA

```
"An elephant" should {
  "be identifiable" in {
    Elephant(42).id must be equalTo 42
  }
}
```

```
case class Elephant(id: Int)
```


CC BY SA

```
"An elephant" should {
  "be identifiable" in {
    Elephant(42, "Mona").id must be equalTo 42
  }

  "have a name" in {
    Elephant(42, "Mona").name must be equalTo "Mona"
  }
}
```

```
case class Elephant(id: Int, name: String)
```


cc BY SA

```
"An elephant" should {
  "be identifiable" in {
    Elephant(42, "Mona").id must be equalTo 42
  }

  "be identified only by its ID" in {
    Elephant(42, "Mona") must be equalTo Elephant(42, "Functo")
  }

  "have a name" in {
    Elephant(42, "Mona").name must be equalTo "Mona"
  }
}
```


cc BY SA

```scala
final case class Elephant(id: Int, name: String) {
  override def equals(other: Any): Boolean =
    other match {
      case Elephant(i, _) => id == i
      case _ => false
    }
}
```


CC BY SA

```
"An elephant" should {
  val id = ID(42)

  "be identifiable" in {
    Elephant(id, "Mona").id must be equalTo id
  }

  "be identified only by its ID" in {
    Elephant(id, "Mona") must be equalTo Elephant(id, "Functo")
  }

  "have a name" in {
    Elephant(id, "Mona").name must be equalTo "Mona"
  }
}

"An ID" should {
  "always be positive" in {
    ID(-42).code must be equalTo 42
  }
}
```


CC BY SA

```scala
final case class Elephant(id: ID, name: String) {
  override def equals(other: Any): Boolean =
    other match {
      case Elephant(i, _) => id == i
      case _ => false
    }
}

final class ID private (val code: Int)
object ID {
  def apply(code: Int): ID = new ID(math abs code)
}
```


CC BY SA

```
"An elephant" should {
  //...

  "have an abbreviated name" in {
    "name"                || "abbreviated"  |
    //--------------------||----------------|
    ""                    !! "<unnamed>"     |
    "Mona"                !! "Mona"          |
    "0123456789"          !! "0123456789"    |
    "0123456789A"         !! "012345678…"    |
    "Eyjafjallajökull"    !! "Eyjafjall…"    |> {

      (name, abbrev) =>
        Elephant(id, name).abbreviatedName must be equalTo abbrev
      }
    }

    //...
}

//...
```

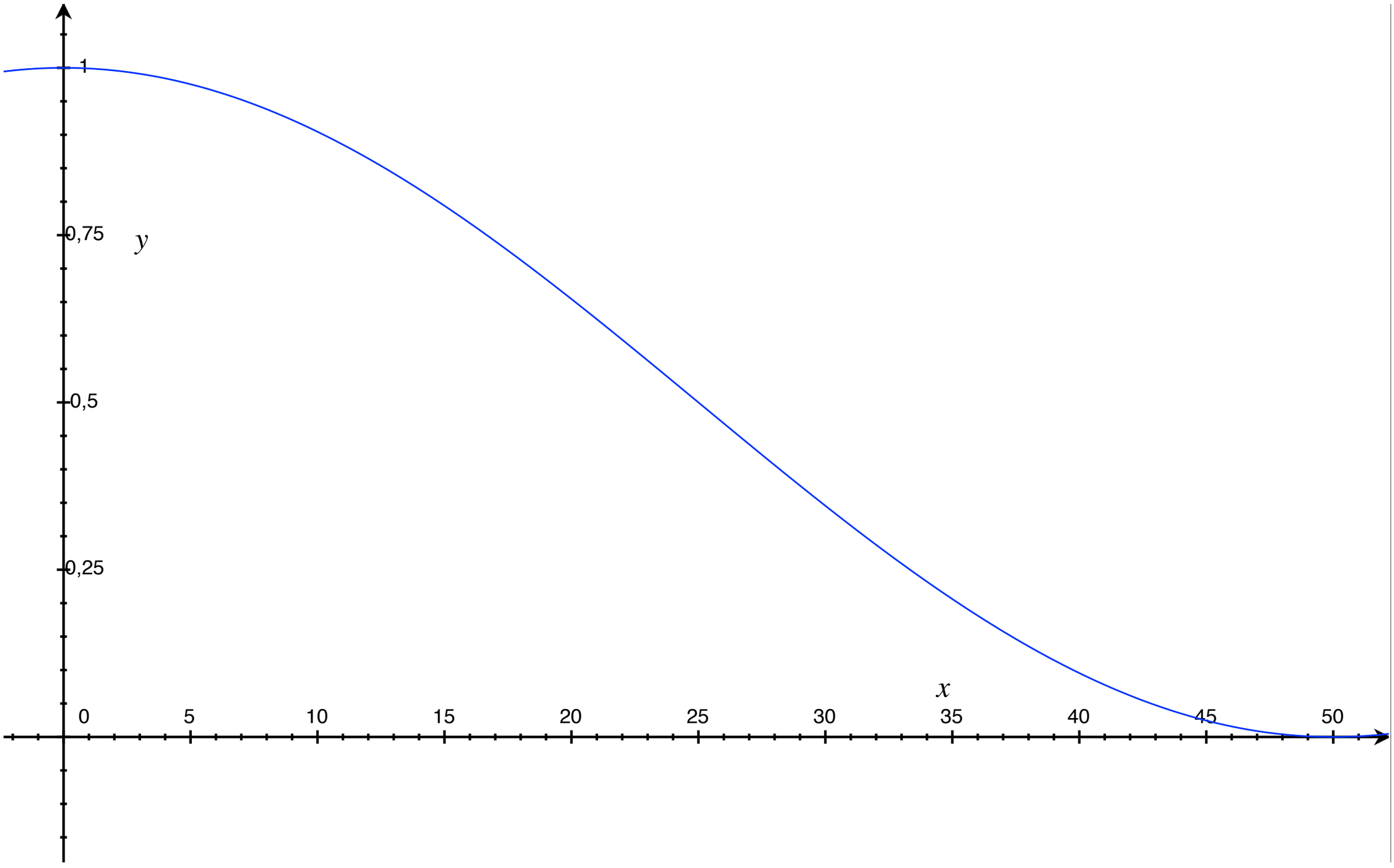

CC BY SA

```scala
final case class Elephant(id: ID, name: String) {
  override def equals(other: Any): Boolean =
    other match {
      case Elephant(i, _) => id == i
      case _ => false
    }

  val abbreviatedName =
    if (name.isEmpty) "<unnamed>"
    else if (name.length <= 10) name
    else name.substring(0, 9) + "…"
}

final class ID private (val code: Int)
object ID {
  def apply(code: Int): ID = new ID(math abs code)
}
```


cc BY SA

```
"An elephant" should {
  //...

  val memory = List(Squee(now), Squee(now - 1.year))

  "remember the squees" in {
    Elephant(id, "Mona", memory).memory must be equalTo memory
  }

  "have a popularity rating" in {
    Elephant(id, "Mona", memory).popularity must be equalTo 1d
  }

  //...
}

//...
```


CC BY SA



CC BY SA

```scala
final case class Elephant(id: ID, name: String, memory: Seq[Squee]) {
  override def equals(other: Any): Boolean = other match {
    case Elephant(i, _, _) => id == i
    case _ => false
  }

  val abbreviatedName =
    if (name.isEmpty) "<unnamed>"
    else if (name.length <= 10) name
    else name.substring(0, 9) + "…"

  def popularity = memory map age filter recent map weight reduce (_ + _)

  def age(squee: Squee) = weeksBetween(squee moment, now)
  def recent(span: Weeks) = span isLessThan weeks(50)
  def weight(span: Weeks) = (cos(span.getWeeks * Pi / 50d) + 1d) / 2d
}

final class ID private (val code: Int)
object ID {
  def apply(code: Int): ID = new ID(math abs code)
}

case class Squee(moment: DateTime)
```


CC BY SA

```
8  final case class Elephant(id: ID, name: String, memory: Seq[Squee]) {
9    override def equals(other: Any): Boolean = other match {
10     case Elephant(i, _, _) => id == i
11     case _ => false
12   }
13
14   val abbreviatedName =
15     if (name.isEmpty) "<unnamed>"
16     else if (name.length <= 10) name
17     else name.substring(0, 9) + "…"
18
19   def popularity = memory map age filter recent map weight reduce (_ + _)
20
21   def age(squee: Squee) = weeksBetween(squee moment, now)
22   def recent(span: Weeks) = span isLessThan weeks(50)
23   def weight(span: Weeks) = (cos(span.getWeeks * Pi / 50d) + 1d) / 2d
24 }
25
26 final class ID private (val code: Int)
27 object ID {
28   def apply(code: Int): ID = new ID(math abs code)
29 }
30
31 case class Squee(moment: DateTime)
```


CC BY SA

```
"An elephant" should {
  //...

  "not equal something else" in {
    Elephant(id, "Mona", Nil) == "fluffy cloud" must beFalse
  }

  //...
}

//...
```


CC BY SA

eu/flierl/treppenschacht/**Elephant.scala** 100 %

Elephant	100 %
ID	100 %
ID	100 %
Squee	100 %

CC BY SA

CC BY SA

```
final case class Elephant(id: ID, name: String, memory: Seq[Squee]) {
  override def equals(other: Any): Boolean = other match {
    case Elephant(i, _, _) => id == i
    case _ => false
  }

  val abbreviatedName =
    if (name.isEmpty) "<unnamed>"
    else if (name.length <= 10) name
    else name.substring(0, 9) + "…"

  def popularity = memory map age filter recent map weight reduce (_ + _)

  def age(squee: Squee) = weeksBetween(squee moment, now)
  def recent(span: Weeks) = span isLessThan weeks(50)
  def weight(span: Weeks) = (cos(span.getWeeks * Pi / 50d) + 1d) / 2d
}

final class ID private (val code: Int)
object ID {
  def apply(code: Int): ID = new ID(math abs code)
}

case class Squee(moment: DateTime)
```

hashCode

unicode

equals,
hashCode,
toString

empty list

future dates

Int.MinValue

CC BY SA

Alter Hut #1

Tests zeigen die
Anwesenheit von Fehlern,
nicht deren **Ab**wesenheit.

(Negativbeweis)

CC BY SA

Alter Hut #2

Testabdeckung (code coverage) ist **kein** Maß für die Güte der Testfälle.

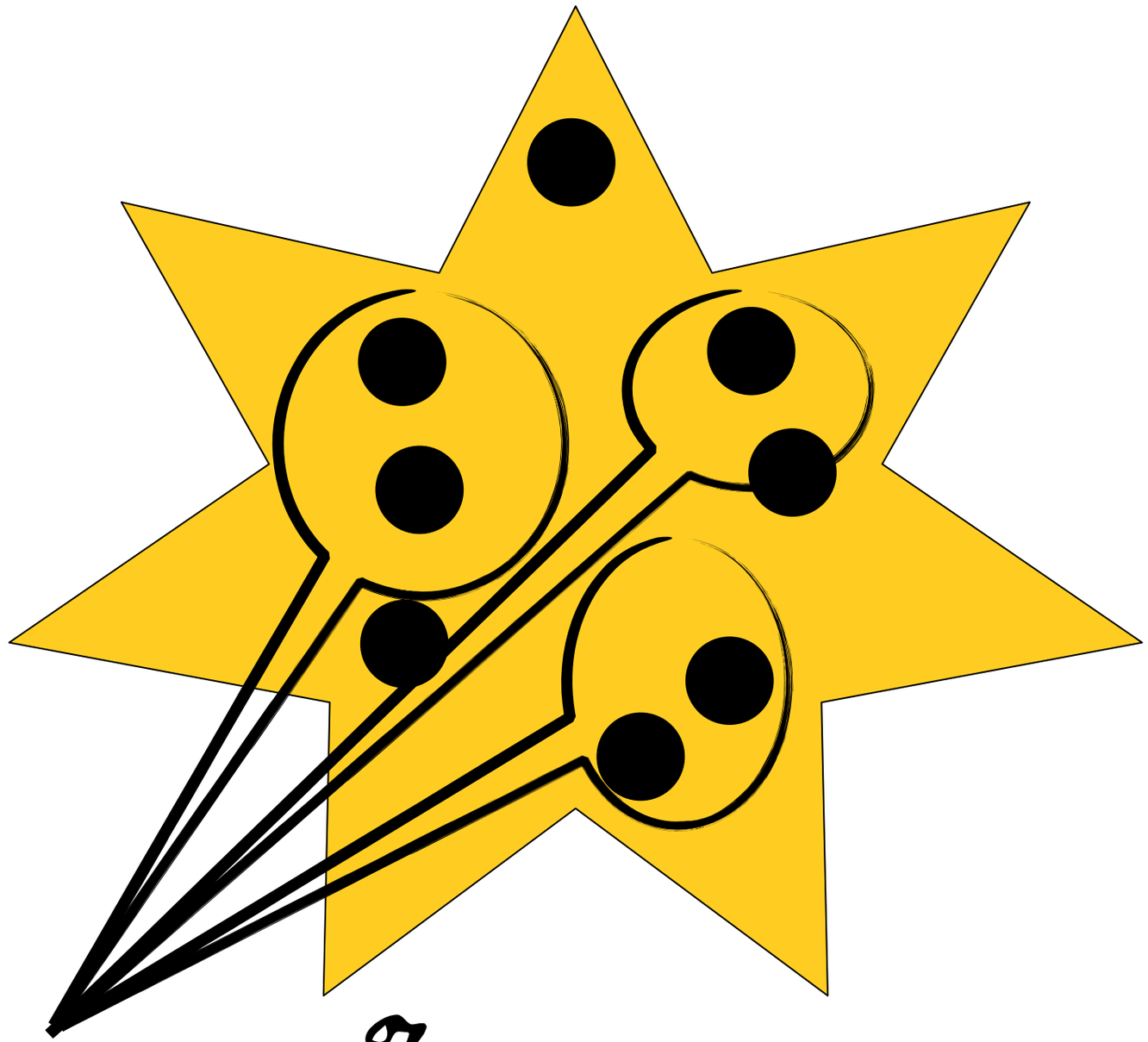
CC BY SA

… oder anders

100% Abdeckung
→ keine Aussage

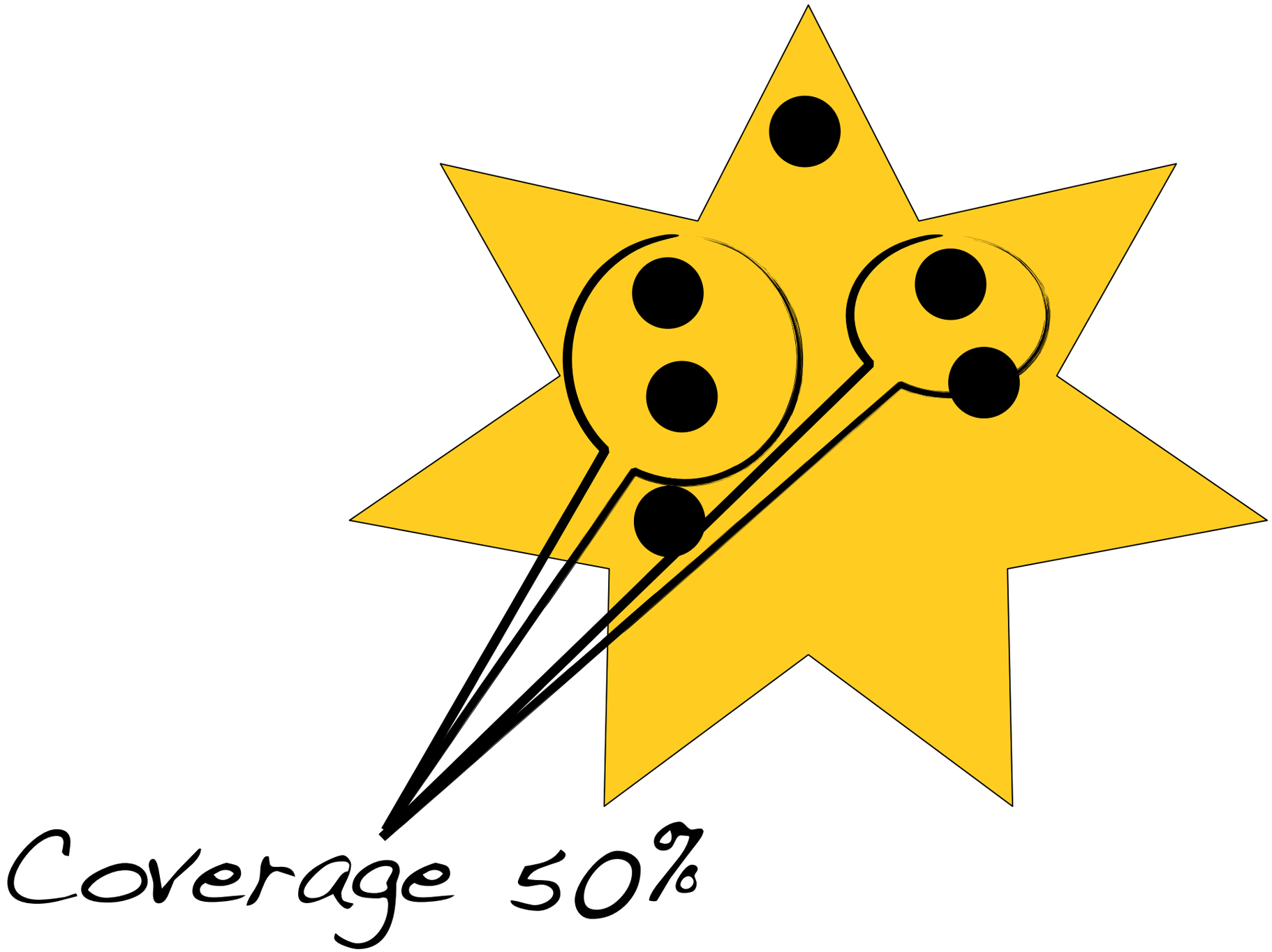
25% Abdeckung
→ Tests unzureichend

CC BY SA

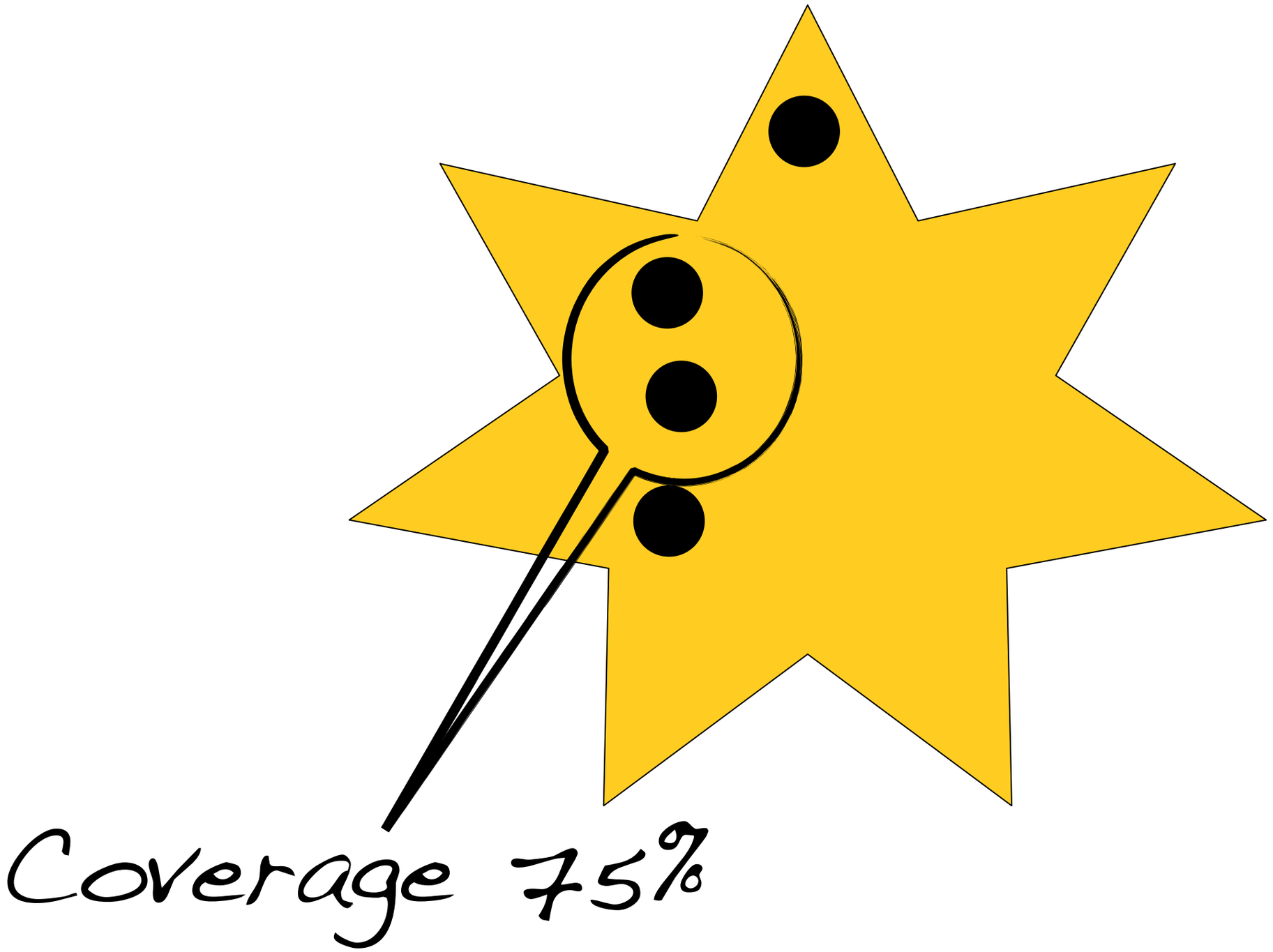


Coverage 25%

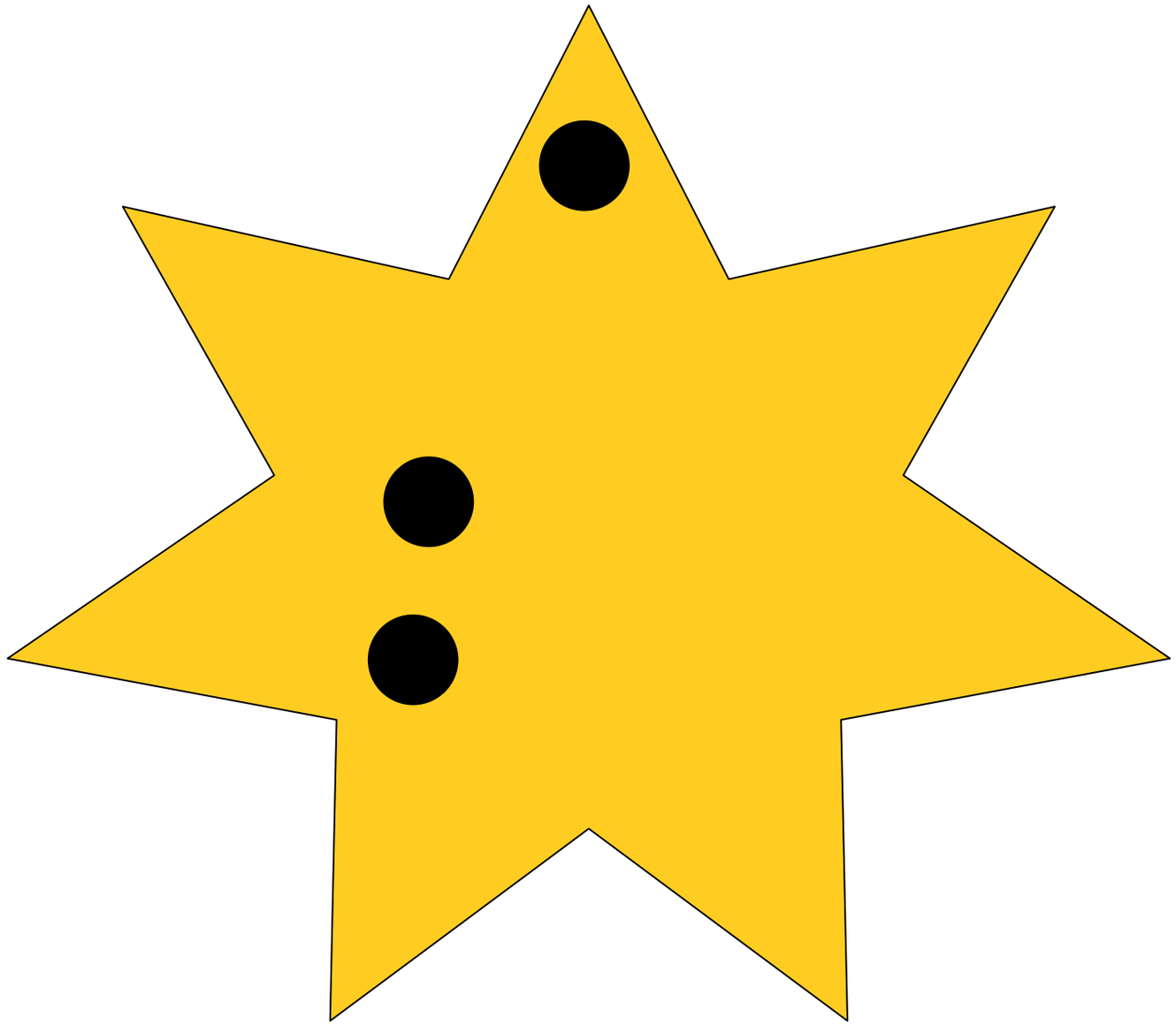
cc BY SA



CC BY SA

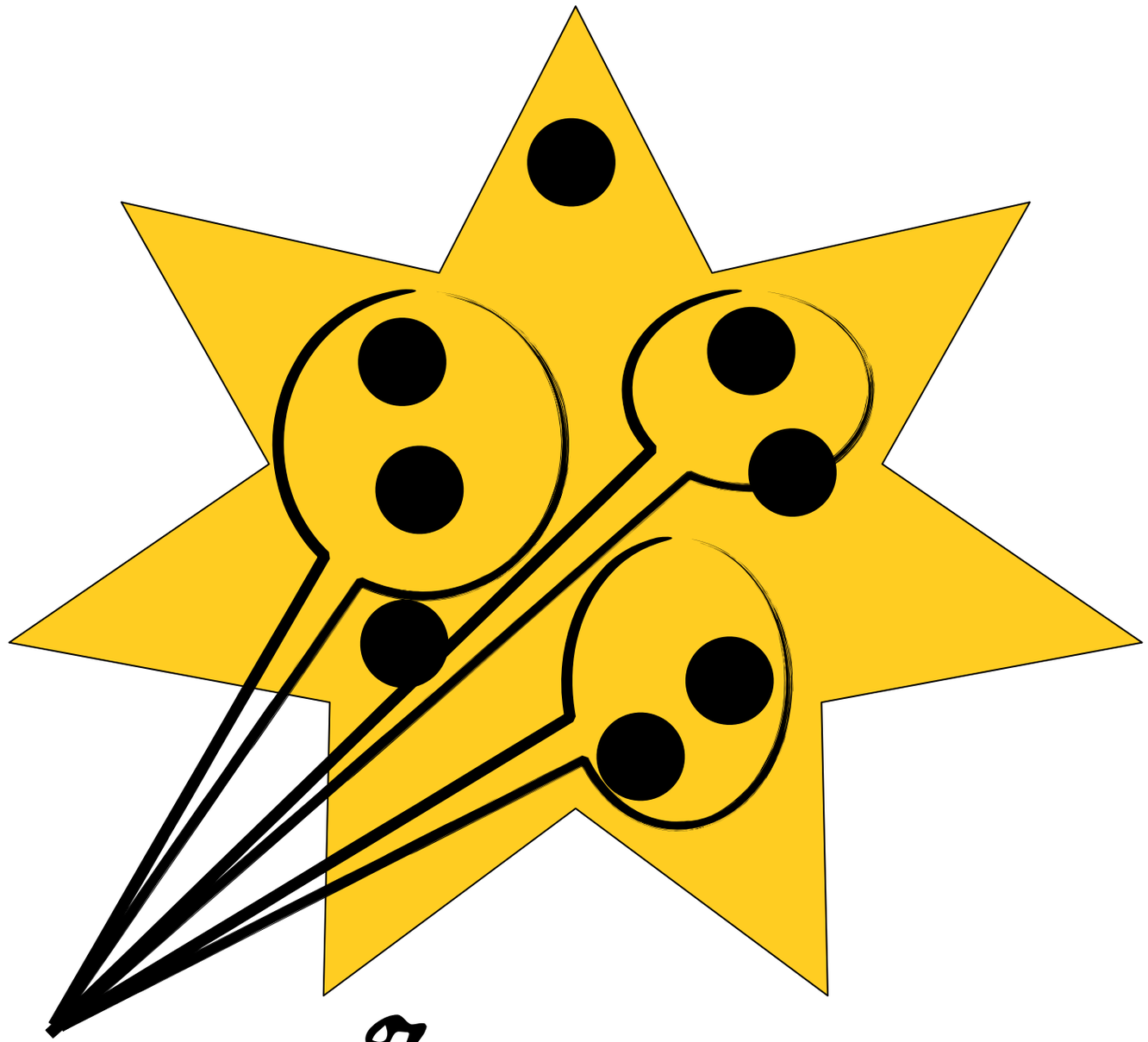


CC BY SA



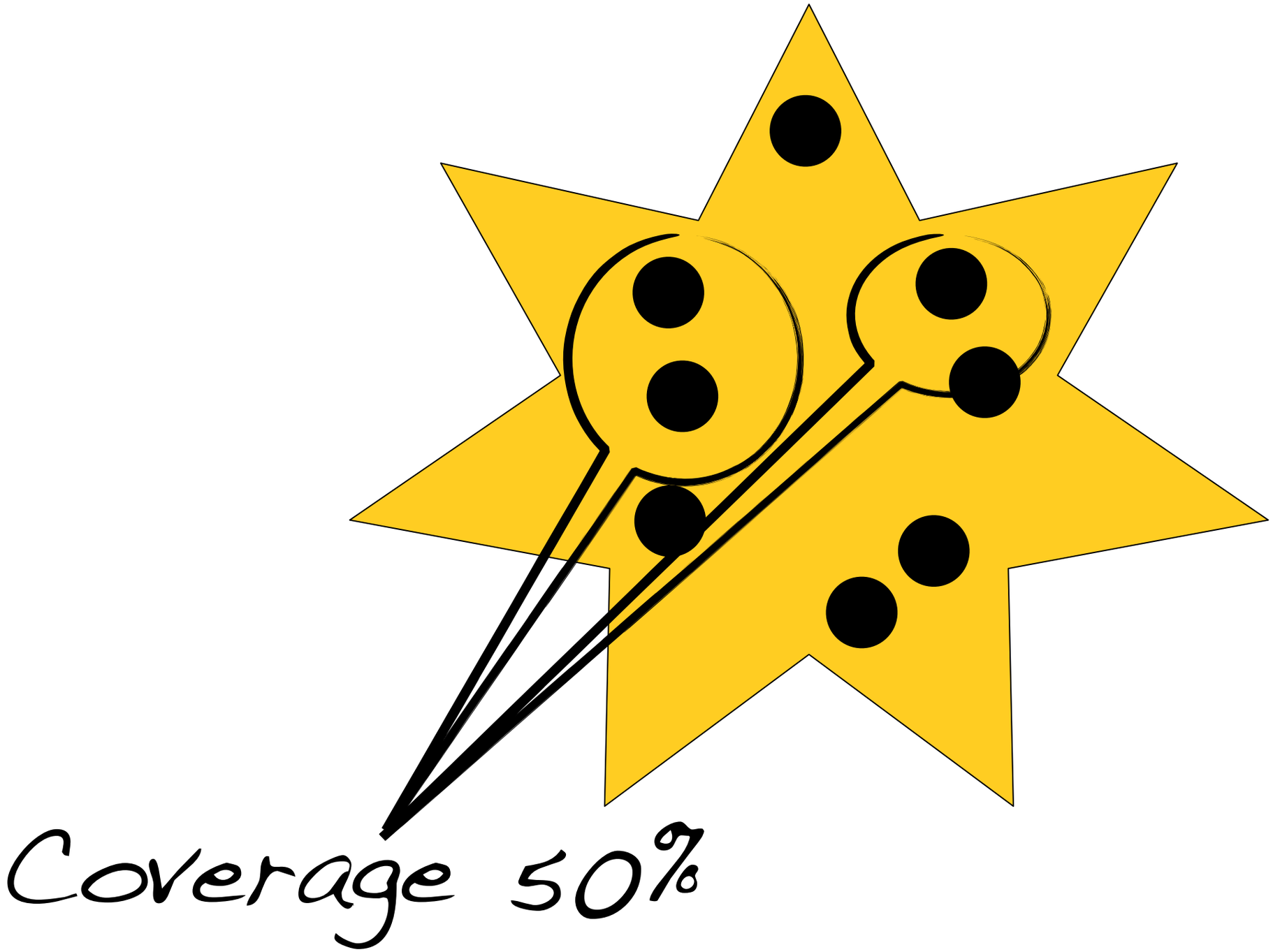
Coverage 100%

CC BY SA

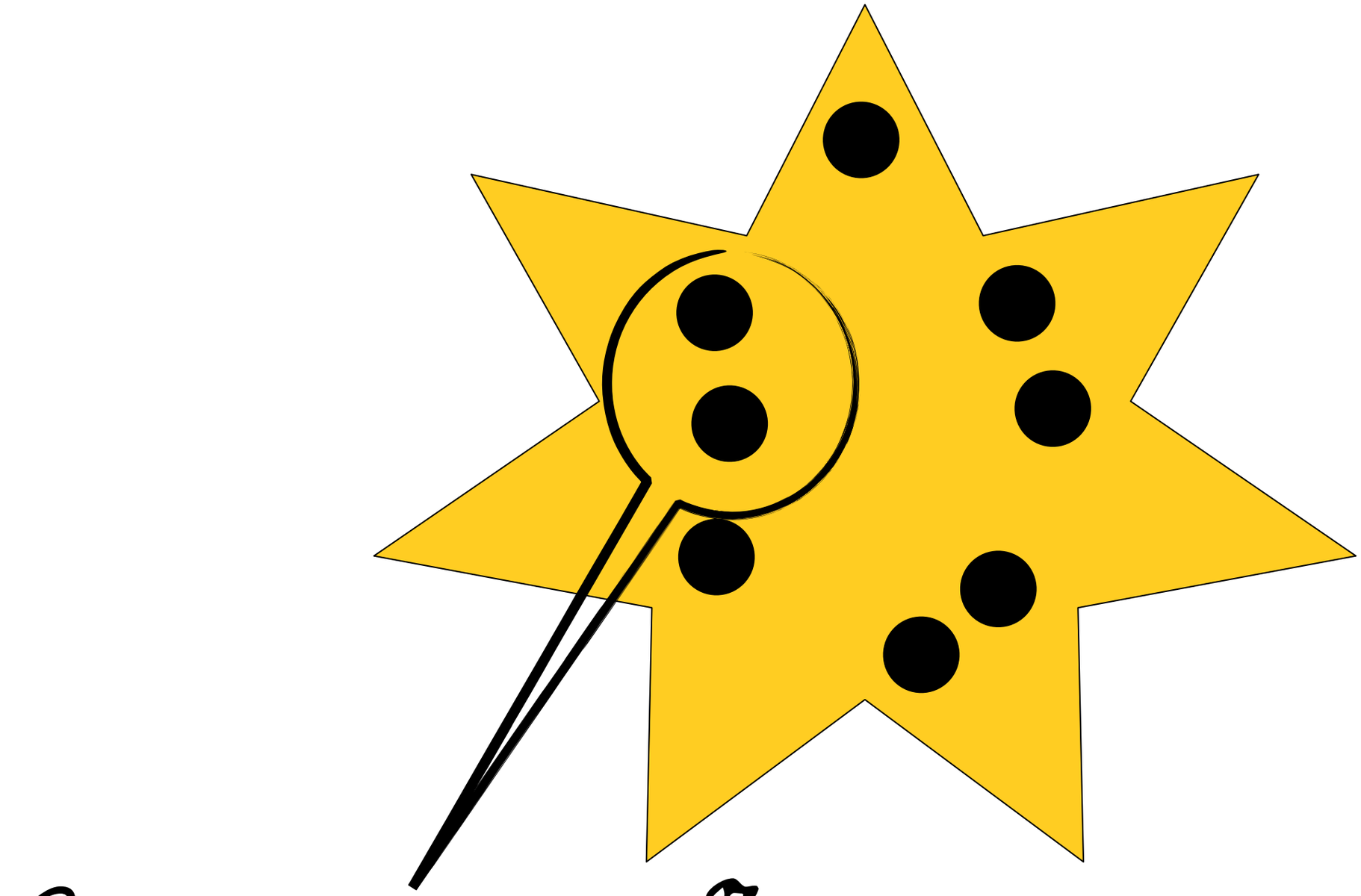


Coverage 25%

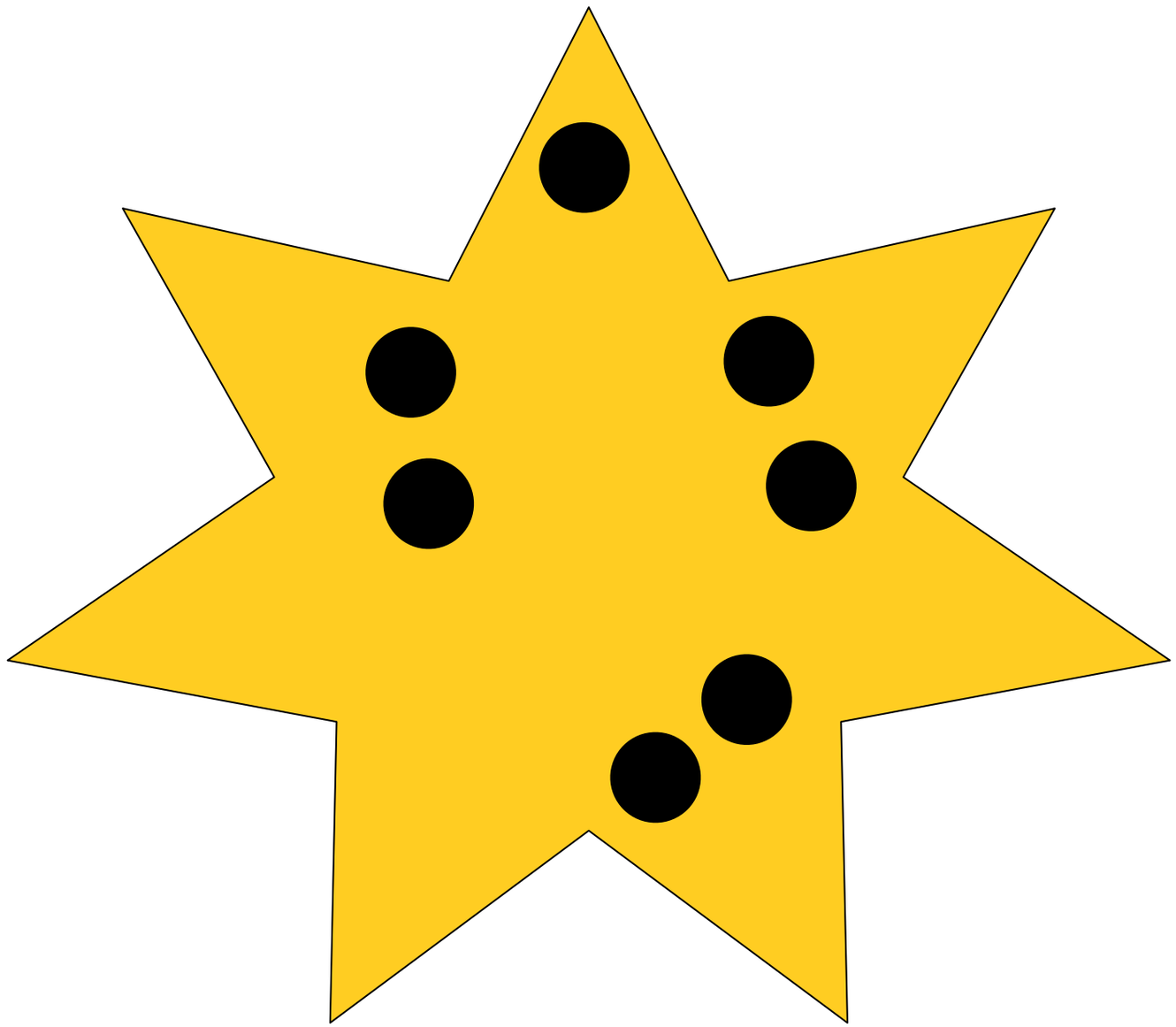
cc BY SA



CC BY SA



CC BY SA

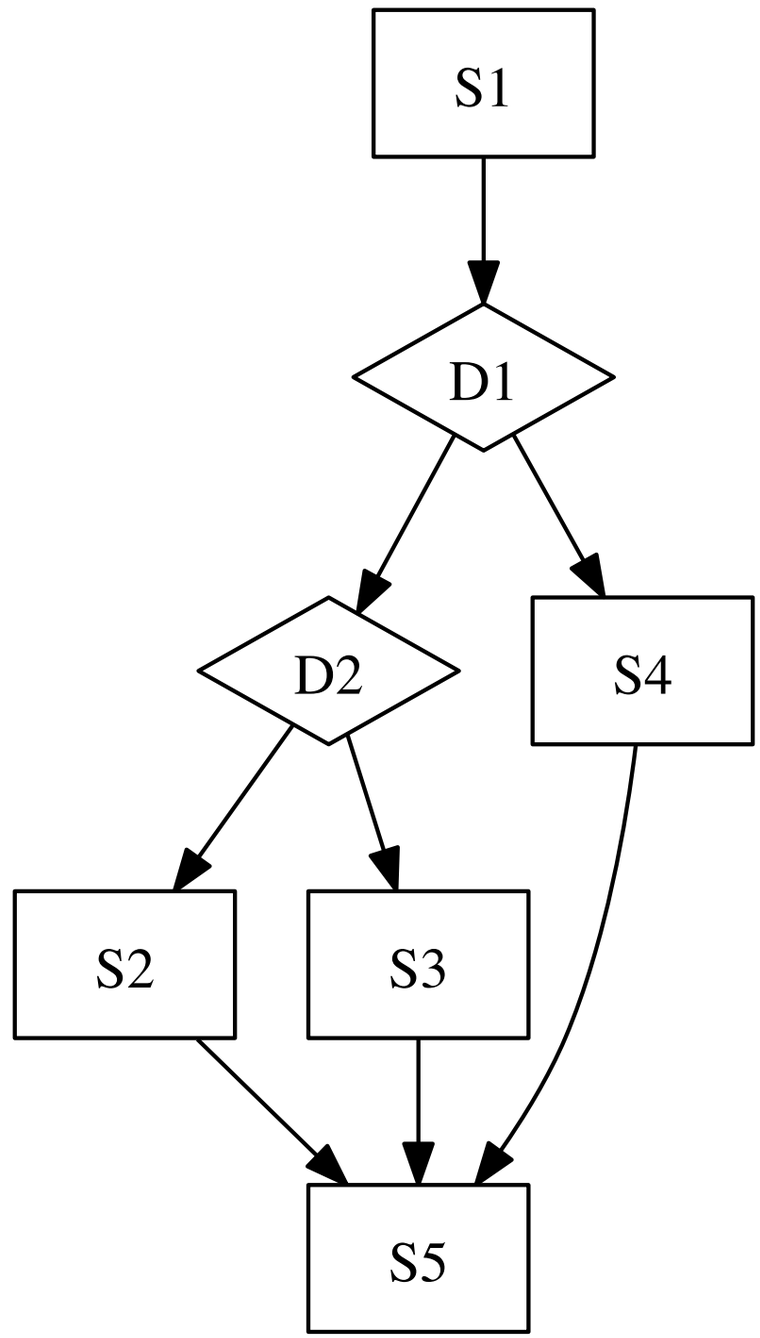


Coverage 100%

CC BY SA

Testabdeckung

```
def f(a: Int, b: Int, c: Int): Int = {
  val x = a + b                    // S1
  val y =
    if (x < c)                     // D1
      if (x % 2 == 0)              // D2
        x                          // S2
      else
        x + 1                      // S3
    else c                         // S4
  x + y                            // S5
}
```



cc BY SA

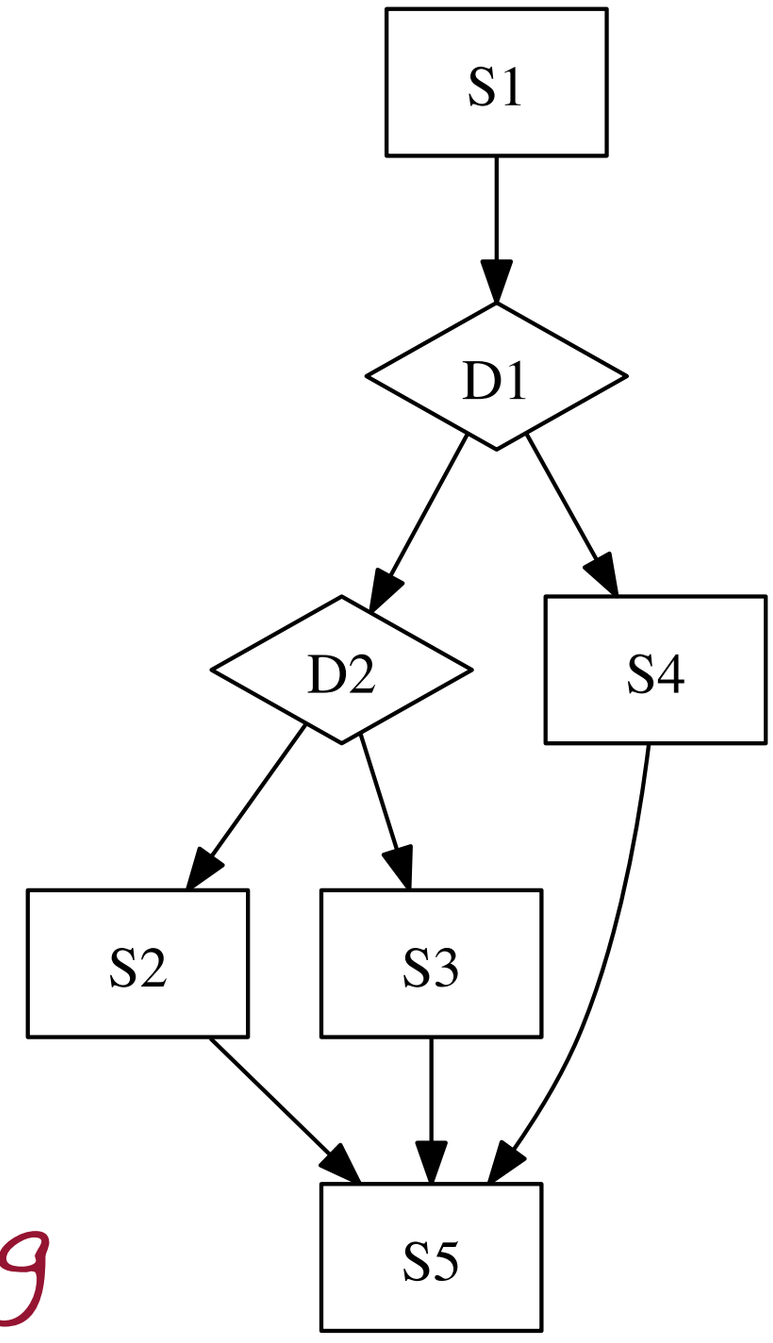
Testabdeckung

C0: Anweisungen

C1: Zweige

C2(a-c): Pfade

C3(a-c): Bedingungen



Achtung!

uneinheitliche Benennung

CC BY SA

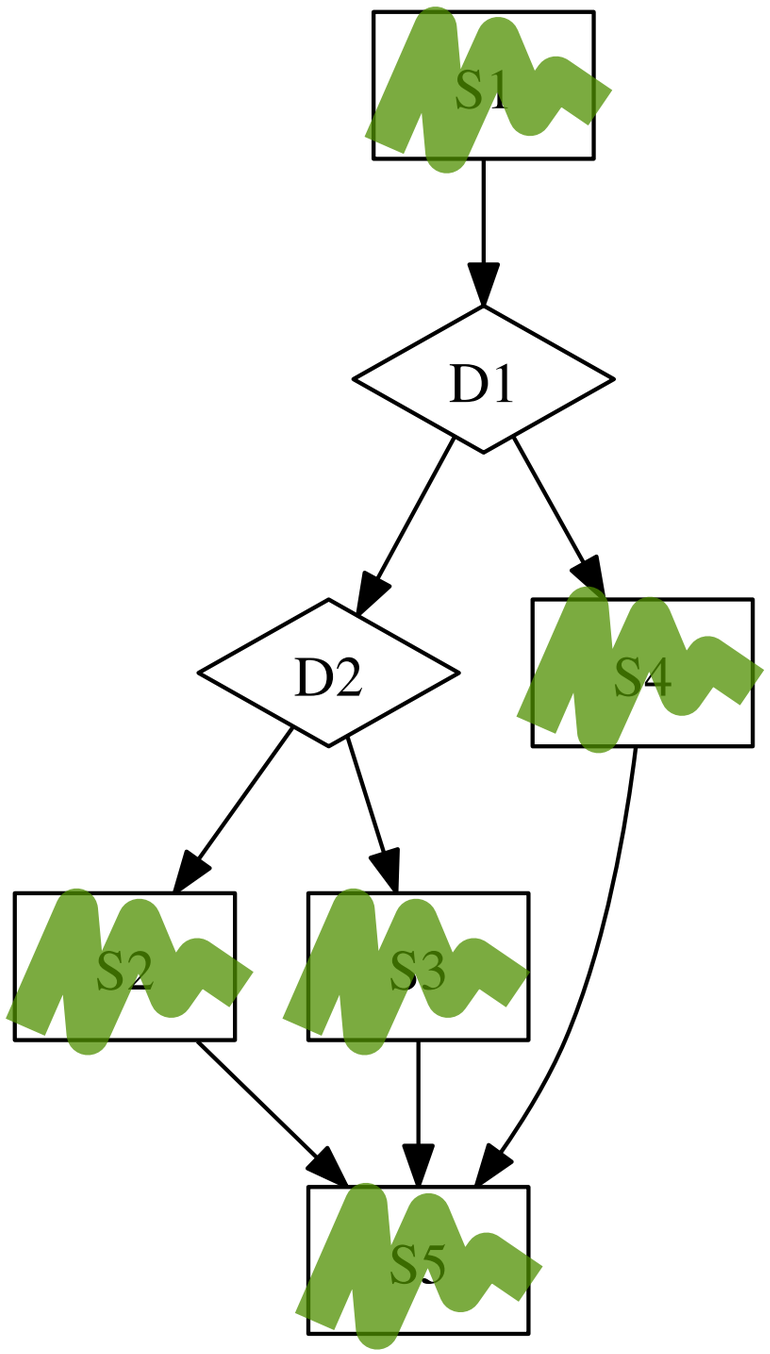
Testabdeckung

C0: Anweisungen

C1: Zweige

C2(a-c): Pfade

C3(a-c): Bedingungen



cc BY SA

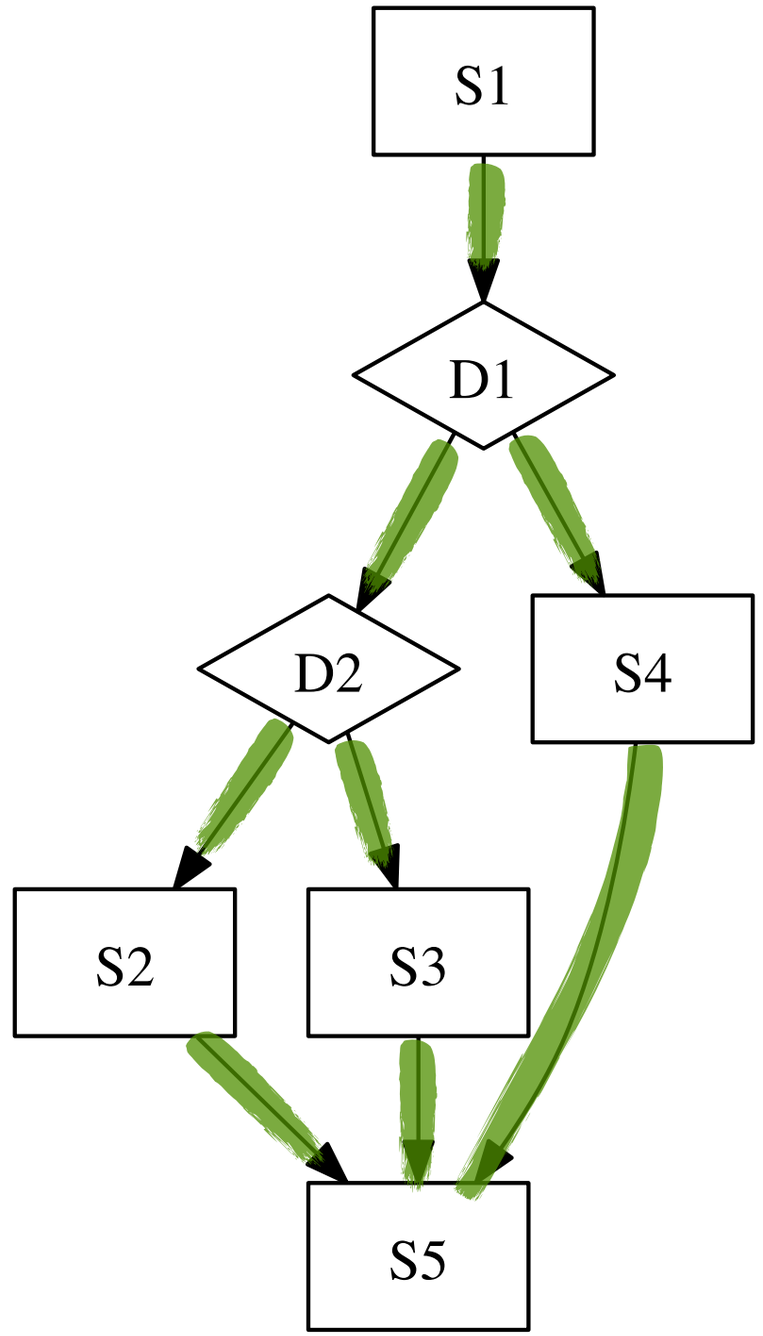
Testabdeckung

C0: Anweisungen

C1: Zweige

C2(a-c): Pfade

C3(a-c): Bedingungen



cc BY SA

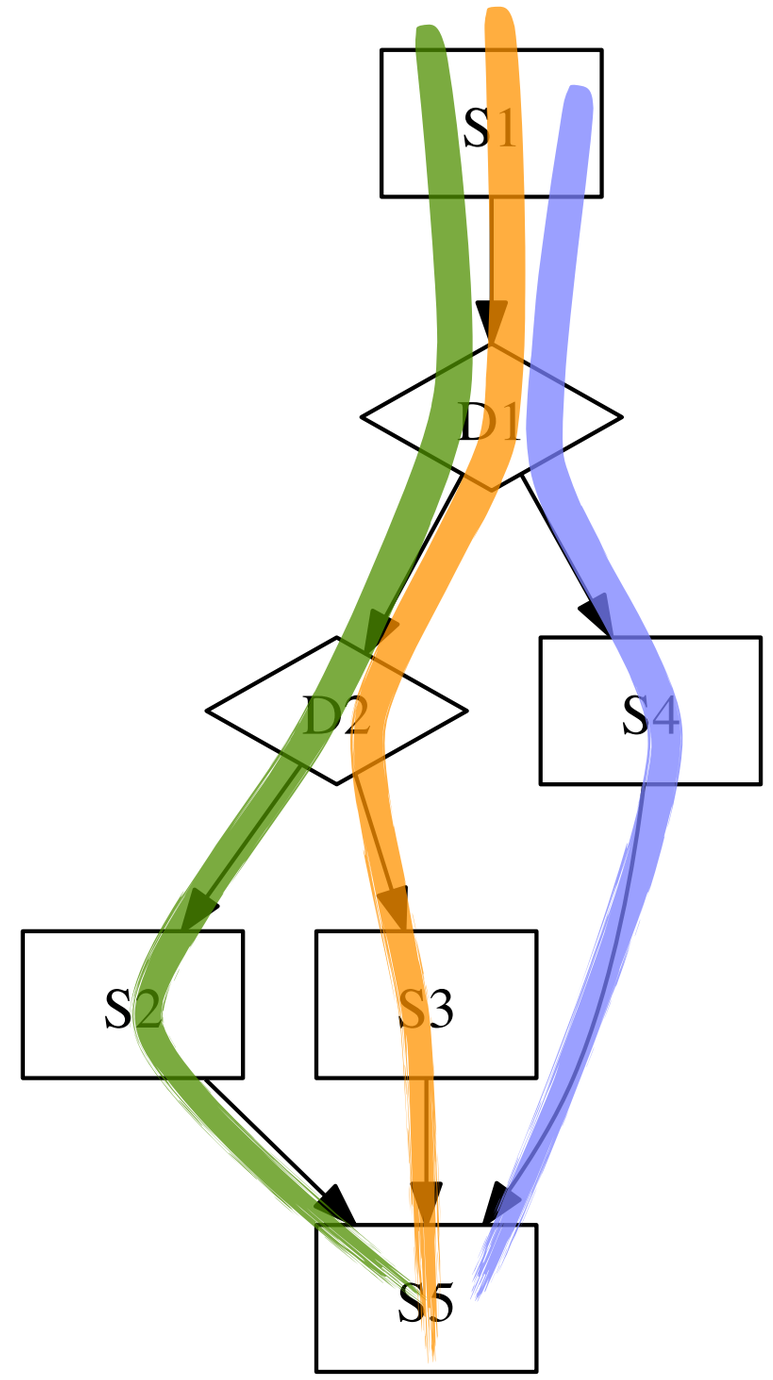
Testabdeckung

C0: Anweisungen

C1: Zweige

C2(a-c): Pfade

C3(a-c): Bedingungen



cc BY SA

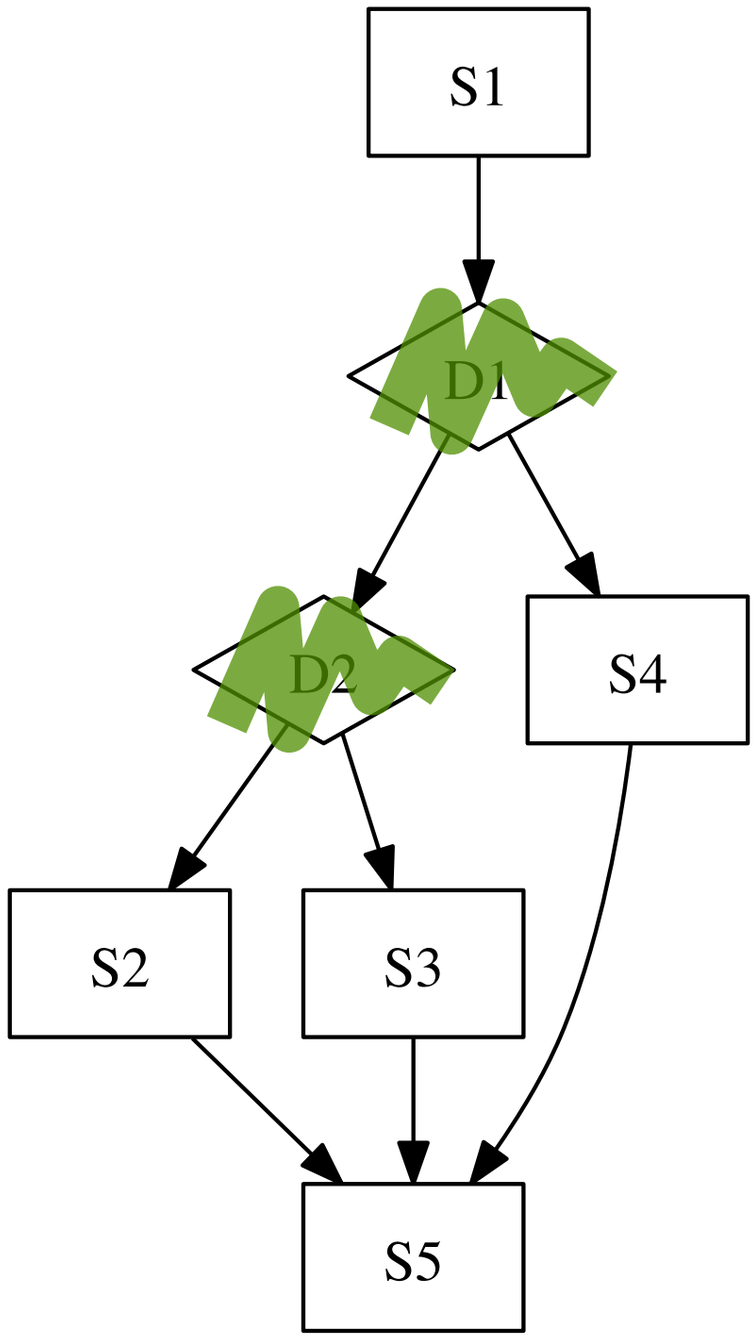
Testabdeckung

C0: Anweisungen

C1: Zweige

C2(a-c): Pfade

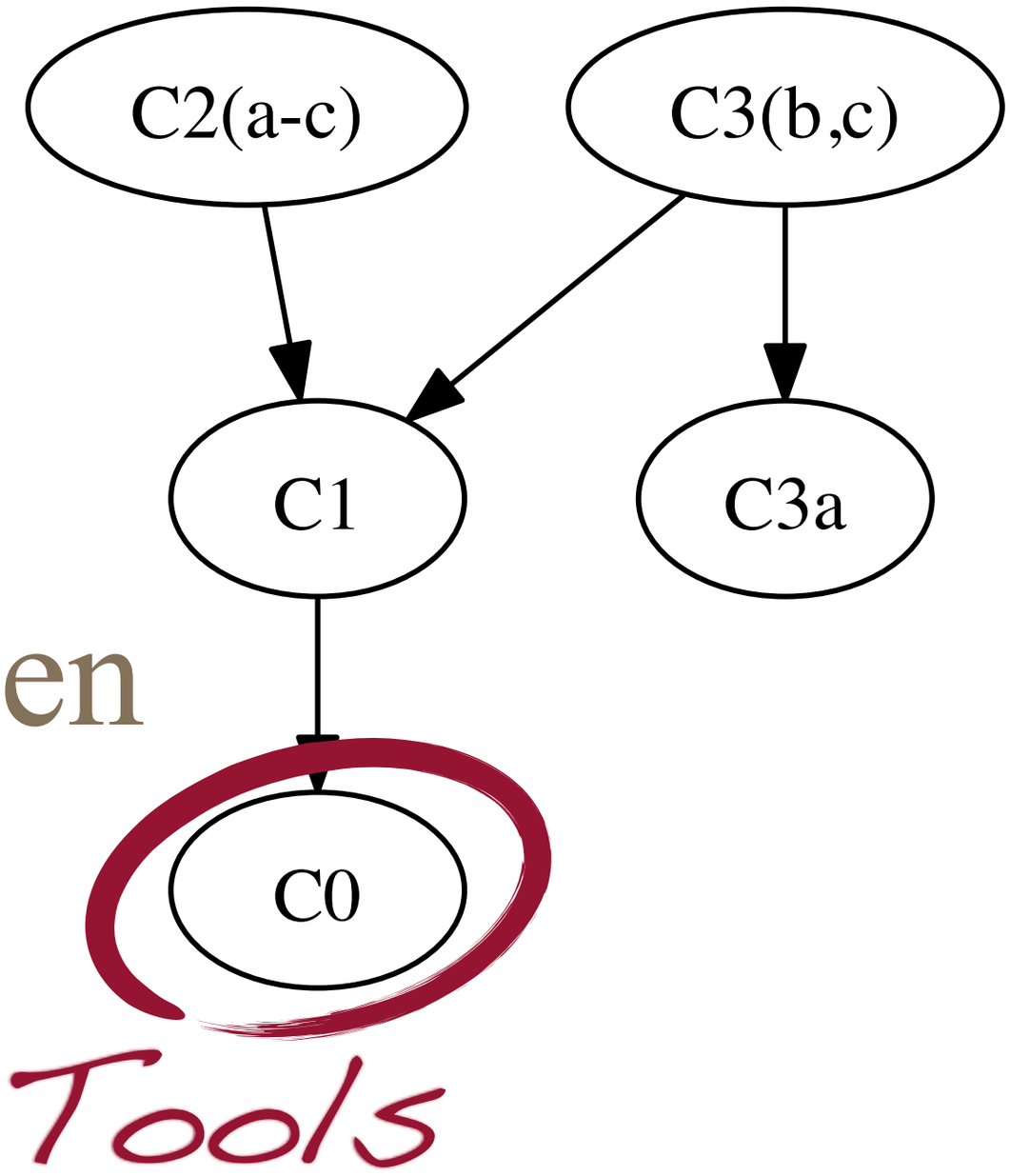
C3(a-c): Bedingungen



cc BY SA

Testabdeckung

C0: Anweisungen
C1: Zweige
C2(a-c): Pfade
C3(a-c): Bedingungen



CC BY SA

Akzeptanzkriterien?

CC BY SA

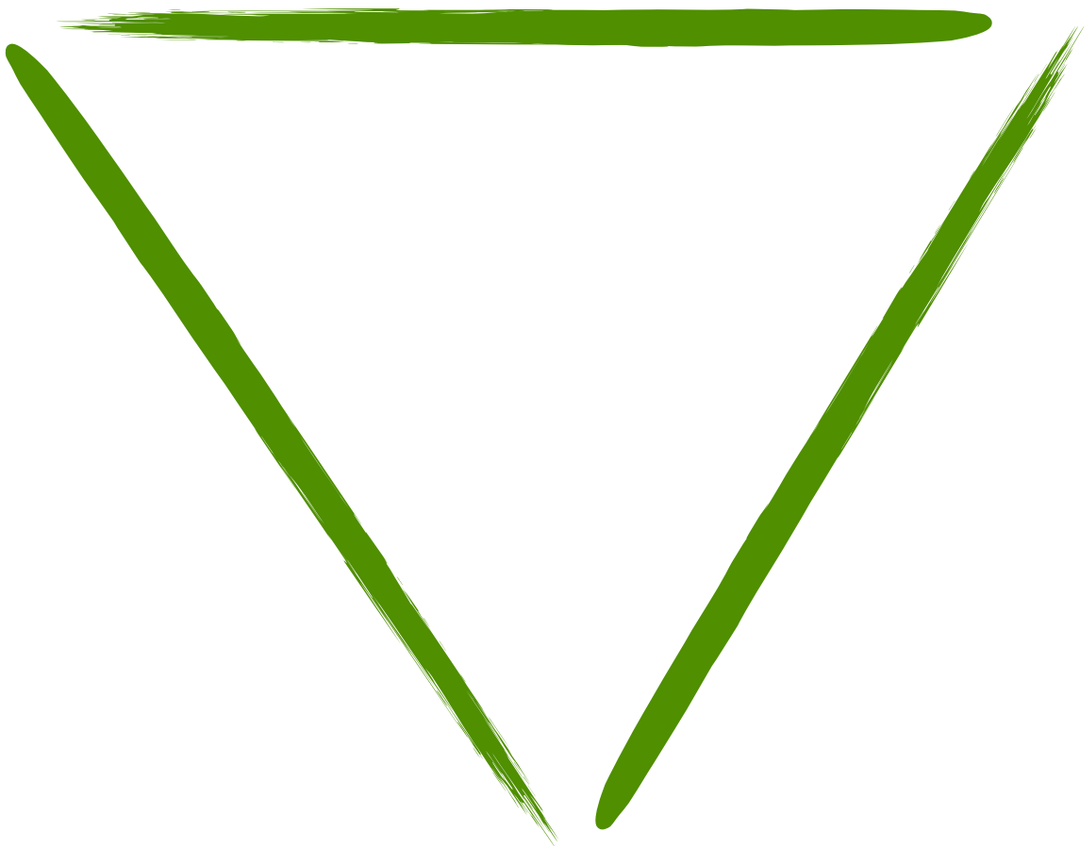
Alter Hut #3

Vollständiger Test ist (außer bei trivialen Testobjekten) **nicht** möglich.

cc BY SA

Und jetzt?

mehr Tests bessere Tests



Wartbarkeit

cc BY SA

Noch mehr Tests?

CC BY SA

```scala
final case class Elephant(id: ID, name: String, memory: Seq[Squee]) {
  override def equals(other: Any): Boolean = other match {
    case Elephant(i, _, _) => id == i
    case _ => false
  }

  val abbreviatedName =
    if (name.isEmpty) "<unnamed>"
    else if (name.length <= 10) name
    else name.substring(0, 9) + "…"

  def popularity = memory map age filter recent map weight reduce (_ + _)

  def age(squee: Squee) = weeksBetween(squee moment, now)
  def recent(span: Weeks) = span isLessThan weeks(50)
  def weight(span: Weeks) = (cos(span.getWeeks * Pi / 50d) + 1d) / 2d
}

final class ID private (val code: Int)
object ID {
  def apply(code: Int): ID = new ID(math abs code)
}

case class Squee(moment: DateTime)
```

VS

```scala
class FooSpec extends Specification with DataTables {
  "An elephant" should {
    val id = ID(42)

    "be identifiable" in {
      Elephant(id, "Mona", Nil).id must be equalTo id
    }

    "be identified only by its ID" in {
      Elephant(id, "Mona", Nil) must be equalTo Elephant(id, "Functo", Nil)
    }

    "not equal something else" in {
      Elephant(id, "Mona", Nil) == "fluffy cloud" must beFalse
    }

    "have a name" in {
      Elephant(id, "Mona", Nil).name must be equalTo "Mona"
    }

    "have an abbreviated name" in {
      "name"             || "abbreviated" |
      //-----------------||--------------|
      ""                 !! "<unnamed>"   |
      "Mona"             !! "Mona"        |
      "0123456789"       !! "0123456789"  |
      "0123456789A"      !! "012345678…"  |
      "Eyjafjallajökull" !! "Eyjafjall…"  |> {

      (name, expectedAbbreviation) =>
        Elephant(id, name, Nil).abbreviatedName must be equalTo expectedAbbreviation
      }
    }

    val memory = List(now, now - 25.weeks, now - 1.year) map Squee

    "remember the squees" in {
      Elephant(id, "Mona", memory).memory must be equalTo memory
    }

    "have a popularity rating" in {
      Elephant(id, "Mona", memory).popularity must be equalTo 1.5d
    }
  }

  "An ID" should {
    "always be positive" in {
      ID(-42).code must be equalTo 42
    }
  }
}
```


CC BY SA

$$\text{loc(tests)} \overset{?}{>} 2\,\text{loc(impl)}$$

$\longrightarrow$ code smell

CC BY SA

→ mehr (sinnvolle) Tests

→ nicht mehr Code

CC BY SA

Bessere Testfälle?

Äquivalenzklassenzerlegung,

Grenzwertanalyse

usw.

CC BY SA

Alter Hut #4

Blindheit gegenüber
den **eigenen** Fehlern.

Pair Programming?

CC BY SA

automatisiert

Durchführung

Testfallauswahl

manuell

CC BY SA

(insert drum roll here)

CC BY SA

Zufällige Testfälle.

nein, wirklich!

... also zumindest fast

CC BY SA

Property-Based Testing

- nicht wirklich neu
- Properties
- “geführte” Auswahl zufälliger Testdaten

cc BY SA

ScalaCheck

- http://github.com/rickynils/scalacheck
- “A powerful tool for automatic unit testing.”
- klein & einfach

CC BY SA

CC BY SA

ScalaCheck

- test case minimisation
- stateful testing

CC BY SA

ScalaCheck läuft...

- mit Specs2 !
- mit ScalaTest
- mit SBT
- oder alleine

cc BY SA

3.– 6. September 2012
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vielen Dank

Andreas Flierl

imbus AG

imbus AG

Spezialisierter Lösungsanbieter für Software-Qualitätssicherung und Software-Test

Innovativ seit 1992

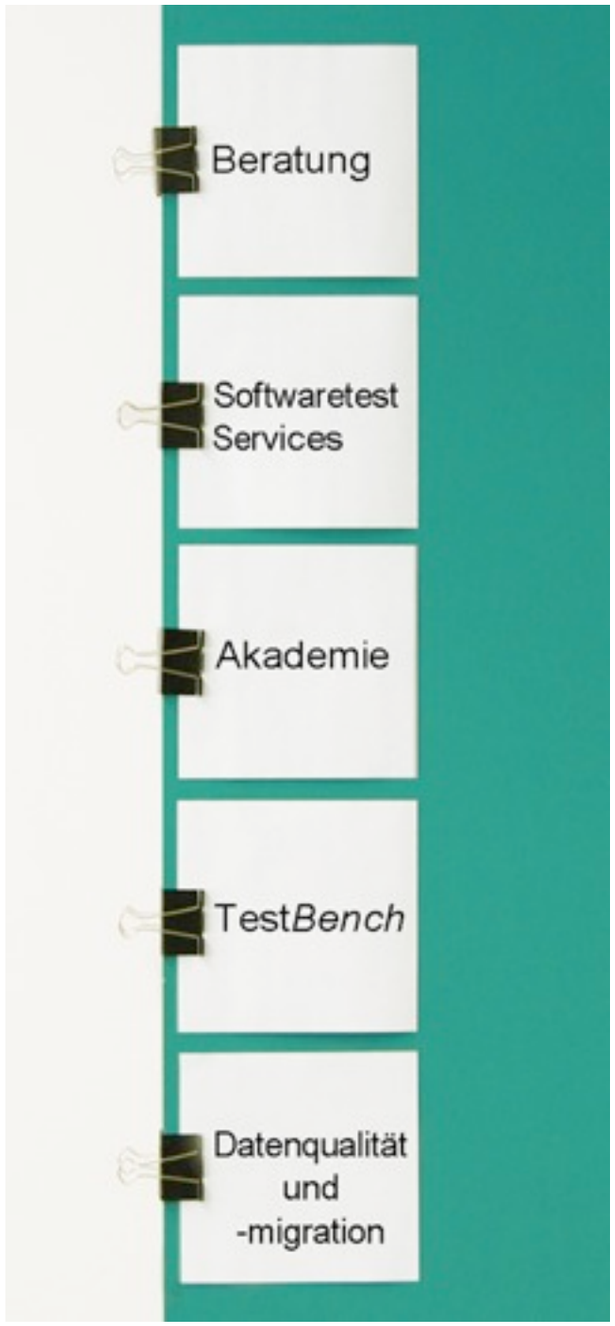
Erfahrung und Know-how aus über 4.000 erfolgreichen Projekten

210 Mitarbeiter an fünf Standorten in Deutschland

Beratung, Test-Services, Training, Tools, Datenqualität

Für den gesamten Software-Lebenszyklus

www.imbus.de



cc BY SA

Bild-Quellen

Elefant auf Folie 3:
Happy elephant vector by VectorPortal.com
Lizenz: Creative Commons 3.0 Attribution Unported

Bilder auf den “Alter Hut”-Folien:
#1 Bundesarchiv, Bild 183-28124-0003 / CC-BY-SA
#2 Deutsche Fotothek
#3 Wolfgang Sauber
#4 Deutsche Fotothek
Aus den “Wikimedia Commons”.
Lizenz: http://creativecommons.org/licenses/by-sa/3.0/

Velociraptor auf Folie 55:
Digital + graphite drawing of Velociraptor mongoliensis
Autor: Matt Martyniuk
Aus den “Wikimedia Commons”.
Lizenz: http://creativecommons.org/licenses/by-sa/3.0/

CC BY SA