POLITECHNIKA
GDAŃSKA

Property based testing

Jak testować zachowanie, a nie przypadki

Paulina Brzęcka Marek Borzyszkowski

14 grudnia 2024

Wstęp

Istnieje wiele koncepcji testowania oprogramowania, jedną z nich jest testowanie na podstawie właściwości [1].

Wstęp

Istnieje wiele koncepcji testowania oprogramowania, jedną z nich jest testowanie na podstawie właściwości [1]. Dziś dowiecie się:

1. na czym polega testowanie na podstawie właściwości.
2. Czym różni się ono od klasycznego podejścia do testowania.
3. Jakie są strategie testowania na podstawie właściwości.
4. Jak przy wykorzystaniu konceptu QuickCheck znaleźć wartości początkowe nieprzechodzące testy.

Przykład

TODO

Strategie

Testowanie na podstawie właściwości nie jest proste w zastosowaniu, przynajmniej przy pierwszej próbie. Istnieją jednak pewne schematy wskazujące drogę, jak stworzyć takie testy.

1. Różne ścieżki, ten sam wynik

Strategie

Testowanie na podstawie właściwości nie jest proste w zastosowaniu, przynajmniej przy pierwszej próbie. Istnieją jednak pewne schematy wskazujące drogę, jak stworzyć takie testy.

1. Różne ścieżki, ten sam wynik
2. Tam i z powrotem

Strategie

Testowanie na podstawie właściwości nie jest proste w zastosowaniu, przynajmniej przy pierwszej próbie. Istnieją jednak pewne schematy wskazujące drogę, jak stworzyć takie testy.

1. Różne ścieżki, ten sam wynik
2. Tam i z powrotem
3. Są rzeczy niezmienne

Strategie

Testowanie na podstawie właściwości nie jest proste w zastosowaniu, przynajmniej przy pierwszej próbie. Istnieją jednak pewne schematy wskazujące drogę, jak stworzyć takie testy.

1. Różne ścieżki, ten sam wynik
2. Tam i z powrotem
3. Są rzeczy niezmienne
4. Z czasem rzeczy przestają się zmieniać

Strategie

Testowanie na podstawie właściwości nie jest proste w zastosowaniu, przynajmniej przy pierwszej próbie. Istnieją jednak pewne schematy wskazujące drogę, jak stworzyć takie testy.

1. Różne ścieżki, ten sam wynik
2. Tam i z powrotem
3. Są rzeczy niezmienne
4. Z czasem rzeczy przestają się zmieniać
5. Dziel i rządź

Strategie

Testowanie na podstawie właściwości nie jest proste w zastosowaniu, przynajmniej przy pierwszej próbie. Istnieją jednak pewne schematy wskazujące drogę, jak stworzyć takie testy.

1. Różne ścieżki, ten sam wynik
2. Tam i z powrotem
3. Są rzeczy niezmienne
4. Z czasem rzeczy przestają się zmieniać
5. Dziel i rządź
6. Łatwiej zweryfikować niż zaimplementować

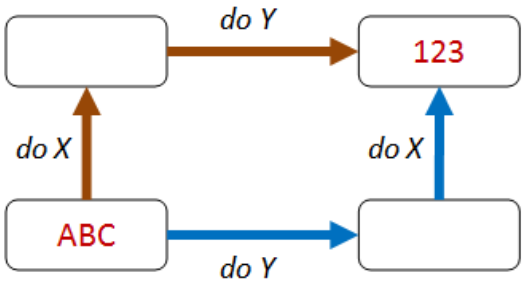
Strategie

Testowanie na podstawie właściwości nie jest proste w zastosowaniu, przynajmniej przy pierwszej próbie. Istnieją jednak pewne schematy wskazujące drogę, jak stworzyć takie testy.

1. Różne ścieżki, ten sam wynik
2. Tam i z powrotem
3. Są rzeczy niezmienne
4. Z czasem rzeczy przestają się zmieniać
5. Dziel i rządź
6. Łatwiej zweryfikować niż zaimplementować
7. Testowanie z wyrocznią

Strategie - Różne ścieżki, ten sam wynik

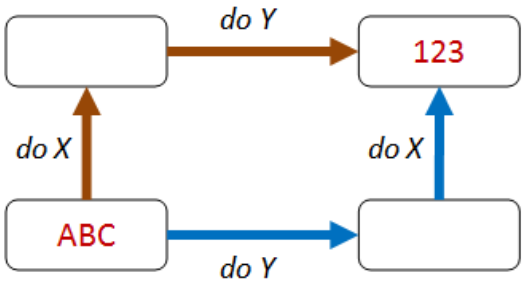
Jedną z podstawowych strategii skorzystanie z komutatywności niektórych operacji. Można to zrobić poprzez wykonanie operacji w różnej kolejności.



Rysunek: Strategia - komutatywność

Strategie - Różne ścieżki, ten sam wynik

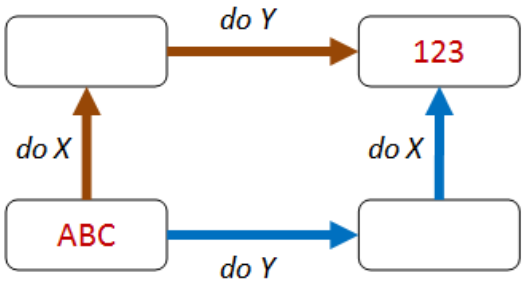
Jedną z podstawowych strategii skorzystanie z komutatywności niektórych operacji. Można to zrobić poprzez wykonanie operacji w różnej kolejności.
Przykładem takiej strategii może być komutatywność dodawania z , gdzie wykorzystano `add x y`, jak i odwrotność tej operacji `add y x`.



Rysunek: Strategia - komutatywność

Strategie - Różne ścieżki, ten sam wynik

Jedną z podstawowych strategii skorzystanie z komutatywności niektórych operacji. Można to zrobić poprzez wykonanie operacji w różnej kolejności.
Przykładem takiej strategii może być komutatywność dodawania z , gdzie wykorzystano `add x y`, jak i odwrotność tej operacji `add y x`.
Innym przykładem jest test metody `sort` danej listy. Wykonanie sortowania, a następnie dodanie do każdego elementu listy `1` powinno dać taki sam efekt jak dodanie `1` do każdego z elementów listy, a następnie jej posortowanie
Rys. **??**

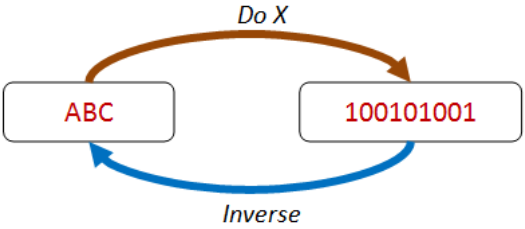


Rysunek: Strategia - komutatywność

Strategie - Tam i z powrotem

Przykładem takiego testu mogą być przeciwne operacje matematyczne jak:

- `dodawanie/odejmowanie`

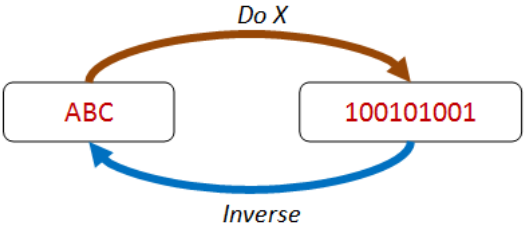


Rysunek: Strategia - inwersja

Strategie - Tam i z powrotem

Przykładem takiego testu mogą być przeciwne operacje matematyczne jak:

- `dodawanie/odejmowanie`
- `mnożenie/dzielenie`

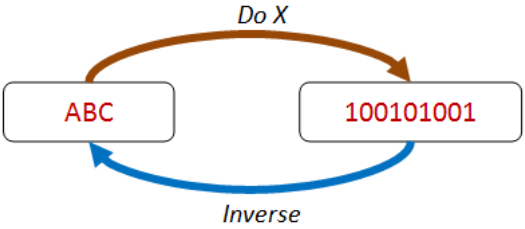


Rysunek: Strategia - inwersja

Strategie - Tam i z powrotem

Przykładem takiego testu mogą być przeciwne operacje matematyczne jak:

- `dodawanie/odejmowanie`
- `mnożenie/dzielenie`
- `potęga/logarytm`.



Rysunek: Strategia - inwersja

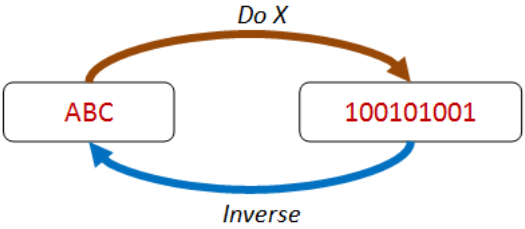
Strategie - Tam i z powrotem

Przykładem takiego testu mogą być przeciwne operacje matematyczne jak:

- `dodawanie/odejmowanie`
- `mnożenie/dzielenie`
- `potęga/logarytm`.

Innymi przykładami są operacje niekoniecznie matematyczne:

- `serializacja/deserializacja`



Rysunek: Strategia - inwersja

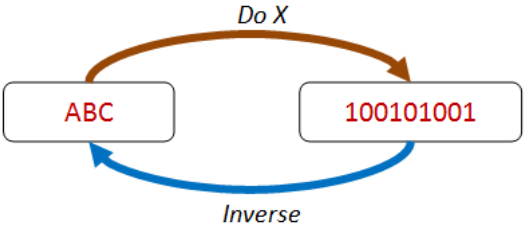
Strategie - Tam i z powrotem

Przykładem takiego testu mogą być przeciwne operacje matematyczne jak:

- `dodawanie/odejmowanie`
- `mnożenie/dzielenie`
- `potęga/logarytm`.

Innymi przykładami są operacje niekoniecznie matematyczne:

- `serializacja/deserializacja`
- `zapis/odczyt z pliku`



Rysunek: Strategia - inwersja

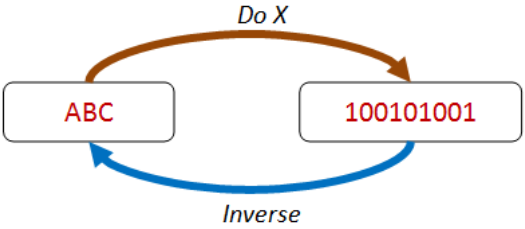
Strategie - Tam i z powrotem

Przykładem takiego testu mogą być przeciwne operacje matematyczne jak:

- `dodawanie/odejmowanie`
- `mnożenie/dzielenie`
- `potęga/logarytm`.

Innymi przykładami są operacje niekoniecznie matematyczne:

- `serializacja/deserializacja`
- `zapis/odczyt z pliku`
- `wstaw/sprawdź czy zawiera`.



Rysunek: Strategia - inwersja

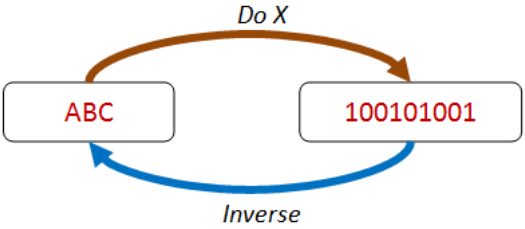
Strategie - Tam i z powrotem

Przykładem takiego testu mogą być przeciwne operacje matematyczne jak:

- `dodawanie/odejmowanie`
- `mnożenie/dzielenie`
- `potęga/logarytm`.

Innymi przykładami są operacje niekoniecznie matematyczne:

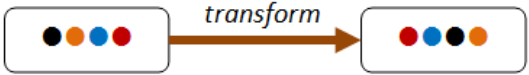
- `serializacja/deserializacja`
- `zapis/odczyt z pliku`
- `wstaw/sprawdź czy zawiera`.
- `odwrócenie listy/odwrócenie listy`



Rysunek: Strategia - inwersja

Strategie - Są rzeczy niezmienne

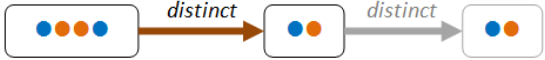
Czasami testowana funkcja przetwarzając dane zachowuje część ich właściwości Rys. 3. Chociażby funkcje `sort` lub `map` wykonane na liście `n` elementów, zwracaja odpowiednio zmodyfikowaną listę `n` elementową.



Rysunek: Strategia - niezmienność

Strategie - Z czasem rzeczy przestają się zmieni

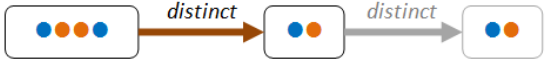
Inną właściwością funkcji może być niezmienność wyniku funkcji po ponownym jej zaaplikowaniu Rys. 4. Innymi słowy, wykonanie funkcji 2 razy daje taki sam efekt, jak jednokrotne jej zaaplikowanie.



Rysunek: Strategia - idempotentność

Strategie - Z czasem rzeczy przestają się zmieni

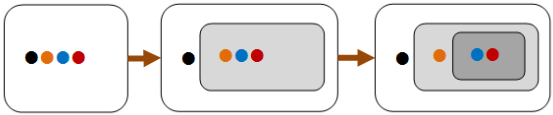
Inną właściwością funkcji może być niezmienność wyniku funkcji po ponownym jej zaaplikowaniu Rys. 4. Innymi słowy, wykonanie funkcji 2 razy daje taki sam efekt, jak jednokrotne jej zaaplikowanie. Przykładami takich operacji, dla których taki typ testu miałby zastosowanie to metoda `distinct` wykonana na danej liście, lub wykonanie `update` na danej bazie danych.



Rysunek: Strategia - idempotentność

Strategie - Dziel i rządź

Istnieją sposoby na testowanie na podstawie właściwości jest wykorzystanie rekursywności struktur przekazywanych do funkcji, takich jak `listy`, `drzewa` Rys. 5.

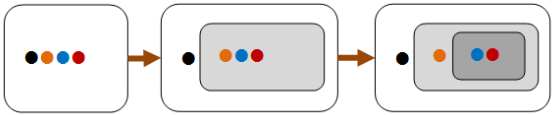


Rysunek: Strategia - rekursywność

Strategie - Dziel i rządź

Istnieją sposoby na testowanie na podstawie właściwości jest wykorzystanie rekursywności struktur przekazywanych do funkcji, takich jak `listy`, `drzewa` Rys. 5. Przykładem może być sprawdzenie za pomocą tej metody funkcji sort List. **??**.

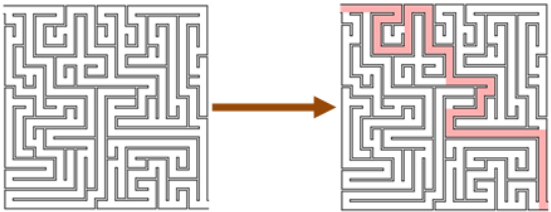
`TU WSTAWIĆ LISTING`



Rysunek: Strategia - rekursywność

Strategie - Łatwiej zweryfikować niż zaimplemen

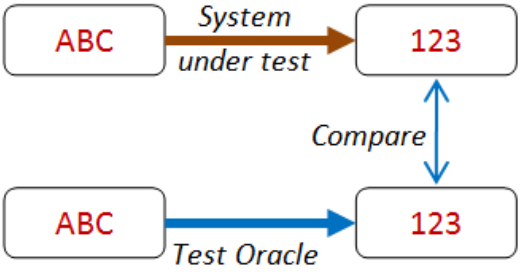
Niekiedy testowana funkcja jest skomplikowana, ale jej rezultat da się łatwo sprawdzić. Przykładem może być funkcja wyszukująca wyjście z labiryntu Rys. 6, gdzie sam algorytm wyszukiwania odpowiedniej ścieżki jest skomplikowany, natomiast samo sprawdzenie, czy ścieżka dobrze prowadzi do wyjścia można w łatwy sposób zweryfikować.



Rysunek: Strategia - łatwe sprawdzenie

Strategie - Testowanie z wyrocznią

Zdaża się, że funkcjonalność została już napisana i trzeba ją zrefactorować, przepisać, napisać od nowa Rys. 7. Warto wtedy wykorzystać wartości zwracane przez oryginalnie zaimplementowany algorytm jako pewną wartość wyniku, pewnego rodzaju wyrocznię, uznając go jako prawdę. W taki sposób można sprawdzić, czy nowa funkcja w pewnym stopniu pokrywa się ze starą funkcją. Czasami też istnieje wiele algorytmów doprowadzających do tego samego wyniku, mające różne złożoności, czy też działające równolegle. Można wykożystać wtedy najprostrzy algorytm jako wyrocznię, ze względu na najmniejsze prawdopodobieństwo napisania takiego algorytmu z błędem.



Rysunek: Strategia - wyrocznia

Pytania?

Bibliografia I

[1] S. Wlaschin, “The ”Property Based Testing” series”,, s. 6, grud. 2014. adr.: https://fsharpforfunandprofit.com/posts/property-based-testing/.

Podziękowania

Chcielibyśmy podziękować Panu dr. inż. Janowi Cychnerskiemu za stworzenie i udostępnienie stylu *pg-beamer*, co zostało wykorzystane do stworzenia tej prezentacji.
`https://github.com/jachoo/pg-beamer`

Dziękujemy za uwagę!

POLITECHNIKA
GDAŃSKA