**Dialyzer**

**DIscrepancy AnaLYZer for ERlang**

**Adam Kisielewski**

**Konrad Onieszczuk**

1. **Wprowadzenie**

Celem projektu było zaznajomienie się z narzędziem do obsługi błędów, jakim jest Dialyzer. Jest to narzędzie do analizy statycznej, którego celem jest wykrywanie różnych rozbieżności programowych, błędów typu, czy nieosiągalne fragmenty kodów.

W ramach projektu przygotowano 4 programy, mające na celu pokazanie funkcjonowania tego narzędzia.

1. **Przykłady użycia Dialyzer’a**
   1. **Test1.erl**

Kod programu:

-module(test1).

-export([run/0]).

run() -> add(5, t).

add(A, B) -> A + B.

Gdybyśmy chcieli uruchomić taki program przez konsolę, wynik byłby taki:

Eshell V7.1 (abort with ^G)

1> c(test1).

{ok,test1}

2> test1:run().

\*\* exception error: an error occurred when evaluating an arithmetic expression in function test1:add/2 (test1.erl, line 6)

Niewiele to wyjaśnia, dlaczego program nie chciał zadziałać. Tzn.jest komunikat o tym, że pojawił się błąd w trakcie wykonywania działania matematycznego w funkcji add/2 i jest podane dokładnie, w której linijce ten błąd wystąpił. Jednakże użycie Dialyzer’a powie więcej na temat tego błędu.

Wywołanie Dialyzera:

Proceeding with analysis...

test1.erl:4: Function run/0 has no local return

test1.erl:4: The call test1:add(5,'t') will never return since it differs in the 2nd argument from the success typing arguments: (number(),number())

test1.erl:6: Function add/2 has no local return

test1.erl:6: The call erlang:'+'(A::5,B::'t') will never return since it differ in the 2nd argument from the success typing arguments: (number(),number())

done in 0m2.63s

done (warnings were emitted)

Function run/0 has no local return – jest to jedno z najpopularniejszych ostrzeżeń. Informuje ono użytkownika, że funkcja nie miała możliwości zwrócenia czegokolwiek, gdyż najprawdopodobniej się nie wykonała.

The call test1:add(5,'t') will never return since it differs in the 2nd argument from the success typing arguments: (number(),number()) – użytkownik jest informowany o tym, że funkcja nic nie zwróci, gdyż któryś z argumentów (w tym przypadku drugi) nie pokrywa się ze wzorcem funkcji (litera nie jest liczbą, więc nie można jej do czegoś dodać).

Podobnie ma się rzecz, jeśli chodzi dwa kolejne komunikaty. Funkcja add/2 nic nie zwraca, bo nie ma co zwrócić, gdyż jeden z argumentów.

* 1. **Test1a.erl**

Kod programu:

-module(test1a).

-export([run/0]).

run() -> add(5, 2).

add(A, B) -> A + B.

Wywołanie Dialyzer’a:

Proceeding with analysis... done in 0m6.70s

done (passed successfully)

Świadczy to o tym, że kod jest prawidłowy.

* 1. **Test2.erl**

Kod programu:

-module(test2).

-export([run/0]).

run() ->

Sth = money(5, you),

one\_op(count(Sth), account(Sth)).

money(Num, Name) -> {give, Num, Name}.

count({give, Num, \_}) -> Num.

account({give, \_, X}) -> X.

one\_op(A, B) -> A + B.

Wywołanie Dialyzer’a:

Proceeding with analysis...

test2.erl:4: Function run/0 has no local return

test2.erl:6: The call test2:one\_op(5,'you') will never return since it differs in the 2nd argument from the success typing arguments: (number(),number())

test2.erl:12: Function one\_op/2 has no local return

test2.erl:12: The call erlang:'+'(A::5,B::'you') will never return since it differs in the 2nd argument from the success typing arguments: (number(),number())

done in 0m2.92s

done (warnings were emitted)

Dialyzer bez problemu znajduje błąd typu, pokazując, że argument ‘you’ nie pasuje do wzorca funkcji, gdzie wszystkie argumenty są liczbami. Bez żadnej przeszkody jest w stanie wywnioskować, jakiego typu są elementy w krotce Sth.

Komunikaty pojawiające się w tym programie są takie same, jak w poprzednim przykładzie. Funkcje: run/0 i one\_op/2 nie zwracają żadnej wartości, ponieważ nie mogą się one odpowiednio wykonać.

* 1. **Test2a.erl**

Kod programu:

-module(test2a).

-export([run/0]).

run() ->

Sth = money(5, you),

one\_op(item(count, Sth), item(account, Sth)).

money(Num, Name) -> {give, Num, Name}.

item(count, {give, X, \_}) -> X;

item(account, {give, \_, X}) -> X.

one\_op(A, B) -> A + B.

Wywołanie Dialyzer’a:

Wszystko działa, tak jak ma działać ;)

Proceeding with analysis... done in 0m2.30s

done (passed successfully)

1. **Podsumowanie**

Powyższe przykłady, chociaż niezbyt skomplikowane pokazują w pełni sprawność dialyzera jeśli chodzi o wykrywanie niezgodności.