Politechnika Warszawska
Wydział Matematyki i Nauk Informacyjnych

Reprezentacja wiedzy

Raport z testów projektu grupy nr 4

Programy działań z akcjami współbieżnymi

Autorzy:

Dragan Łukasz
Flis Mateusz
Izert Piotr
Pielat Mateusz
Rząd Przemysław
Siry Roman
Waszkiewicz Piotr
Zawadzka Anna

4 czerwca 2016

1 Opis projektu

Tematem testowanego przez nas projektu są programy działań z akcjami współbieznymi. Rozpatrywana klasa systemów dynamicznych spełnia następujące warunki:

- Prawo inercji
- Niedeterminizm
- W języku kwerend występują akcje złożone (zbiory co najwyzej k akcji atomowych), w języku akcji jedynie akcje atomowe
- Pełna informacja o wszystkich akcjach atomowych i wszystkich ich skutkach bezpośrednich
- Z każdą akcją atomową zwiazany jest jej warunek wstepny (ew. TRUE) i końcowy (efekt akcji)
- Wykonywane są jedynie akcje bezkonfliktowe (żadne dwie akcje składowe nie mogą mieć wspólnych zmiennych, na które w jakimkolwiek stanie mają wpływ)
- Wynikiem akcji złożonej jest suma skutków wszystkich składowych akcji bezkonfliktowych
- Akcje mogą być niewykonalne w pewnych stanach; jeśli akcja jest niewykonalna, to każda akcja ją zawierająca jest niewykonalna
- Dopuszczalny jest opis częściowy zarówno stanu poczatkowego, jak i pewnych stanów wynikajacych z wykonań sekwencji akcji

Opracowywany jezyk kwerend ma za zadanie umożliwiać tworzenie zapytań, pozwalających na uzyskanie odpowiedzi na następujace pytania:

- Czy podany program P działań jest możliwy do realizacji zawsze/kiedykolwiek ze stanu początkowego?
- Czy wykonanie programu P działań w stanie początkowym prowadzi zawsze/kiedykolwiek do osiągniecia celu γ?
- Czy cel γ jest osiagalny ze stanu początkowego?

2 Przeprowadzone testy

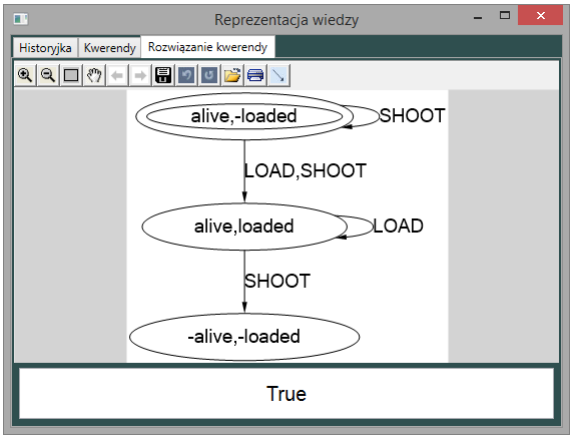
2.1 Test 1

Zdefiniowana dziedzina:

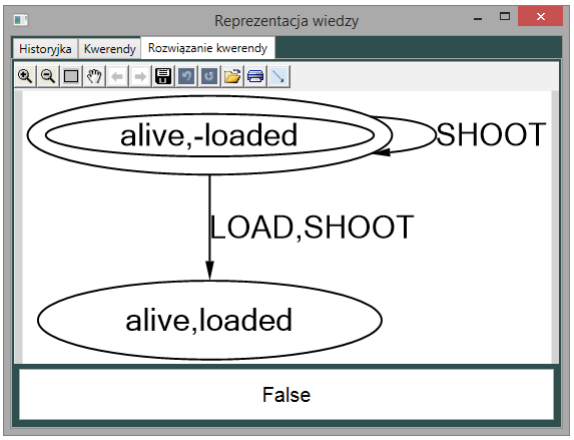
```
initially alive
initially ¬loaded
LOAD causes loaded
SHOOT causes ¬loaded if loaded
SHOOT causes ¬alive if loaded
```

Zadane kwerendy:

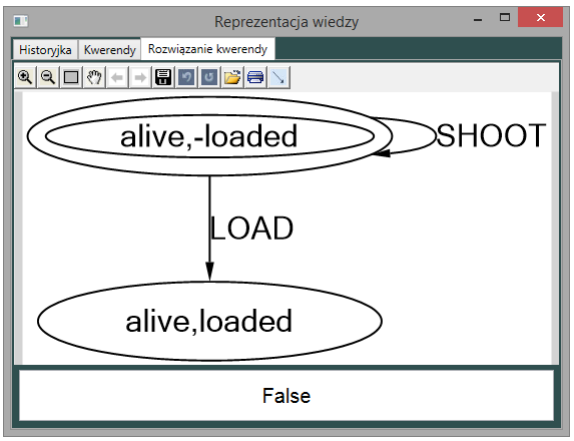
- `ever accessible ¬alive`
  Odpowiedź: `TRUE`
  Jest to odpowiedź poprawna, ponieważ istnieje ciąg akcji, który prowadzi do stanu spełniającego podany cel, co ilustruje poniższy graf:



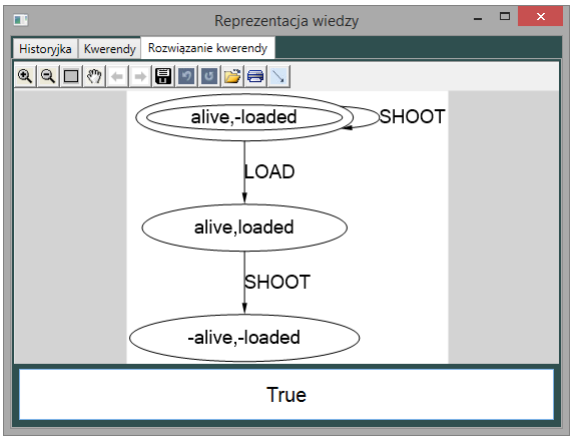
- `always accessible ¬alive`
  Odpowiedź: `FALSE`
  Jest to prawda, ponieważ nie wszystkie ciągi akcji prowadzą do stanu, który spełnia podany warunek.



- `ever ¬alive after SHOOT, LOAD`
  Odpowiedź: `FALSE`
  Jest to odpowiedź poprawna, ponieważ ze stanu początkowego, w którym zmienna *loaded* nie jest prawdziwa, akcje *SHOOT* i *LOAD* nie zmienią stanu zmiennej *alive*.



- `always executable SHOOT, LOAD, SHOOT`
  Odpowiedź: `TRUE`
  Odpowiedź jest zgodna z prawdą - w dziedzinie nie istnieją zadania, które miałyby uniemożliwiać wykoanie ciągu podanych akcji.


Reprezentacja wiedzy
Historyjka
Kwerendy
Rozwiązanie kwerendy
alive,-loaded
SHOOT
LOAD
alive,loaded
SHOOT
-alive,-loaded
True

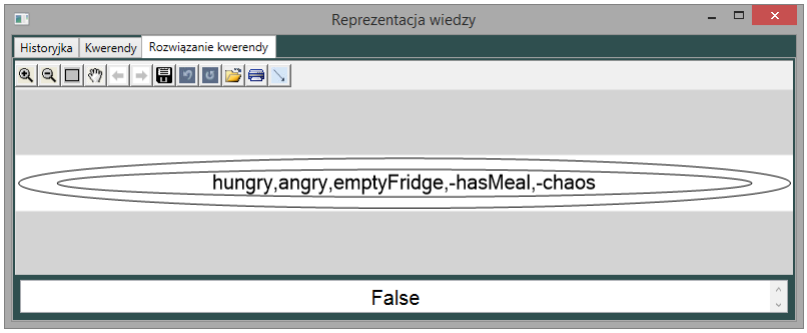
2.2 Test 2

Zdefiniowana dziedzina:

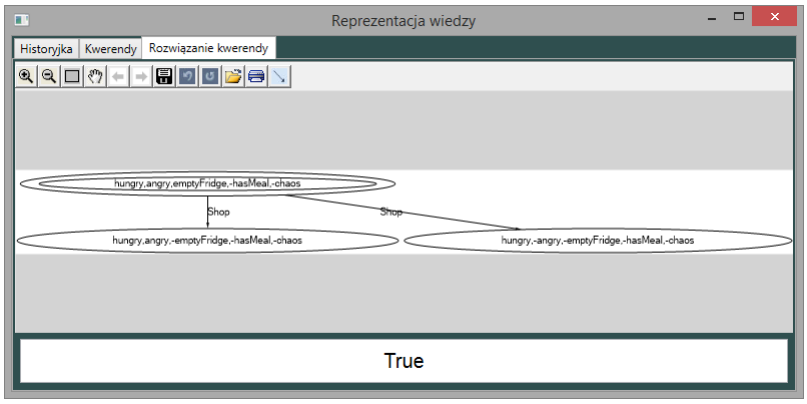
```
initially hungry ∧ angry
initially emptyFridge ∧ ¬hasMeal ∧ ¬chaos
impossible Eat if ¬hasMeal
impossible Cook if emptyFridge
Shop causes ¬emptyFridge
Shop releases angry if angry
Cook causes chaos if angry
Cook causes hasMeal
Cook releases emptyFridge if emptyFridge
Eat causes ¬hasMeal ∧ ¬hungry ∧ ¬angry
```

Zadane kwerendy:

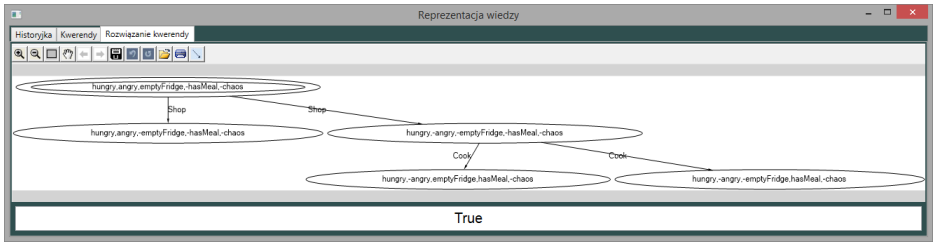
- `always executable COOK, SHOP`
  Odpowiedź: `FALSE`
  Odpowiedź jest poprawna, ponieważ zdanie impossible uniemożliwia wykonanie akcji *COOK* w przypadku gdy zmienna *emptyFridge* jest prawdziwa.



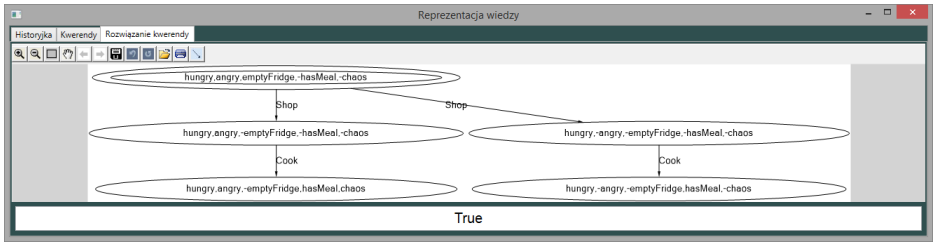
- `always executable {COOK, SHOP}`
  Odpowiedź: `TRUE`
  Odpowiedź jest poprawna, ponieważ w tym wypadku zdefiniowana została akcja złożona z akcji *COOK* i *SHOP*. Akcja *COOK* nie jest wykonywalna ze stanu początkowego, więc pod uwagę brana jest tylko akcja *SHOP*, która jestw wykonywalna ze stanu początkowego.



- `ever accessible emptyFridge` $\wedge$ $\neg$`angry` $\wedge$ `hungry` $\wedge$ $\neg$`chaos`
  Odpowiedź: `TRUE`
  Jest to odpowiedź poprawna, jednak nie jest zrozumiałe, dlaczego w grafie zwróconym przez program widnieją stany, które nie spełniają podanych założeń i nie należą do ścieżki, która prowadzi do takiego stanu.



- `ever` $\neg$`emptyFridge after SHOP, COOK`
  Odpowiedź: `TRUE`
  Odpowiedź jest zgodna z prawdą, ponieważ akcja *COOK* może, ale nie musi zmienić stan zmiennej *emptyFridge*.



2.3 Test 3 - kapelusznik w krainie czarów

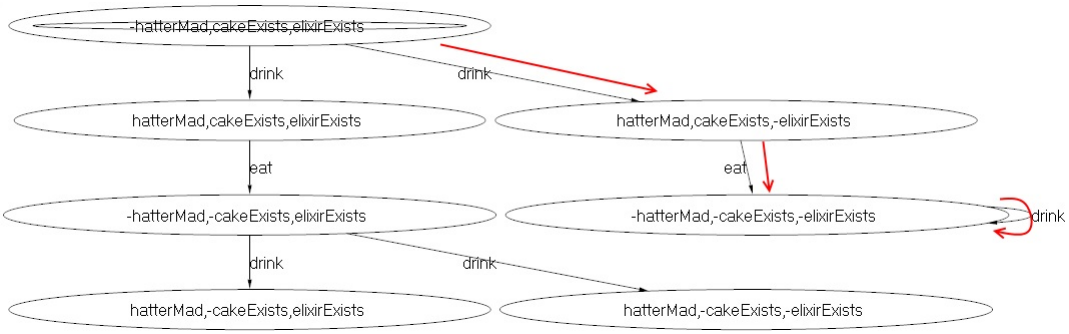
Zdefiniowana dziedzina:

```
initially ¬hatterMad ∧ cakeExists ∧ elixirExists
drink causes hatterMad if elixirExists
eat causes ¬hatterMad
impossible eat if ¬cakeExists
drink releases elixirExists if elixirExists
eat causes ¬cakeExists
```

Zadane kwerendy:

- `ever cakeExists after eat`
  Oczekiwana odpowiedź: `FALSE`
  Odpowiedź programu: `FALSE`
  Odpowiedź jest poprawna, ponieważ jedzenie zawsze powoduje brak ciastka, a w dziedzinie nie ma możliwości przywrócenia ciastka.
- `always executable drink, drink`
  Oczekiwana odpowiedź: `TRUE`
  Odpowiedź programu: `TRUE`
  Odpowiedź jest poprawna, ponieważ nawet jeśli eliksir się skończy, akcja picia jest wykonywalna.
- `always accessible hatterMad`
  Oczekiwana odpowiedź: `TRUE`
  Odpowiedź programu: `TRUE`
  Odpowiedź jest poprawna.
- `ever  hatterMad after drink, eat, drink, eat`
  Oczekiwana odpowiedź: `FALSE`
  Odpowiedź programu: `FALSE`
  Odpowiedź jest poprawna - druga próba zjedzenia ciastka jest niewykonalna.

- `ever ¬hatterMad after drink, eat, { drink, eat}`
  Oczekiwana odpowiedź: `TRUE`
  Odpowiedź programu: `TRUE`
  Odpowiedź jest poprawna, ścieżkę można zobaczyć na poniższym obrazku.



2.4 Test 4

2.5 Test 5

2.6 Test 6

2.7 Test 7

2.8 Test 8

2.9 Test 9

3 Wnioski