RAPORT WSTĘPNY

Projekt L3 Rachunek sytuacyjny i wnioskowanie w przód.

Wykonać program realizujący symulację środowiska i misję ratunkową autonomicznego

latającego drona (agenta) w nieznanym terenie, o cyfrowej postaci „N × N × H” „komórek”.

Celem działania agenta jest odnalezienie pilota i zrzucenie mu zaopatrzenia oraz powrót do

miejsca startu. „Zagrożenia” to radary i działa nieprzyjaciela – można je w uproszczeniu

modelować jako prostopadłościany o różnych rozmiarach i wysokości.

a) Wykonać symulator środowiska odpowiedzialny za dostarczanie obserwacji agentowi i za

śledzenie wykonywanych przez niego akcji. Agent „wyczuwa” radary i broń na kilka

kratek wprzód (im wyżej leci tym szybciej może lecieć, ale zużywa wtedy odpowiednio

więcej energii). Radary należy omijać w poziomie a wystrzały można ominąć w poziomie

albo wznieść się powyżej obserwowanego zasięgu działa. W celu zrzucenia zaopatrzenia

dron powinien obniżyć się do minimalnej wysokości H\_min = 0.1 H.

Ewentualny kontakt z radarem „unieruchamia” agenta na pewien czas, a trafienie

pociskiem zmniejsza jego maksymalną prędkość o połowę. Wynik polega na pomiarze

czasu od startu do powrotu przy uwzględnieniu faktu czy zadanie zostało wykonane czy

też nie.

b) Wykonać bazę wiedzy agenta. Wyrazić wiedzę agenta w **rachunku sytuacyjnym** języka

predykatów (można przyjąć własną implementację zapisu formuł). Zrealizować funkcje TELL i ASK dla komunikacji funkcji głównej agenta z bazą wiedzy.

c) Zaimplementować ogólny (uniwersalny) algorytm **wnioskowania w przód**, w celu

wnioskowania o nowych faktach i wnioskowania o wyborze akcji.

d) Wykonać prosty interfejs graficzny. W osobnym okienku pokazywać wykonane

wnioskowania. Parametry N i H mają być zmienne w rozsądnych granicach.

e) Umożliwić wykonanie testowania działania programu dla różnych ustawień

początkowych *świata*. Wykonać testowanie i przeanalizować jego wyniki.

-----------------------------------------------------------------------------------------------------------------

W realizacji projektu stworzony zostanie system ekspercki wykorzystujący bazę wiedzy zbudowaną za pomocą klauzul Horna (posiadających jeden wniosek oraz mogących zawierać wiele założeń). Klauzule te zostały wybrane ze względu na to posiadaną właściwość:

Aby opisać poszczególne kroki posłużę się rysunkiem z wykładu:



Wykład ITO-mgr 1 strona 32

Dla właściwej oceny tego co się dzieje w środowisku agent posiada czujniki. Za ich pomocą możemy poczynić jakieś obserwacje, które z kolei są podstawą do tego, żeby aktualizować stan drona i wiadomości o tym środowisku. Na podstawie tego aktualnego stanu przy wykorzystaniu regułowej bazy wiedzy można wykonać odpowiednie wnioskowanie oraz wykonać jakieś akcje, które zależeć będą od aktualnego stanu środowiska.

Projekt ten rozpatruję w kategorii zmodyfikowanego 3-wymiarowego świata Wumpusa. Agent (dron) startować będzie z pozycji początkowej w terenie NxNxH komórek i będzie poruszał się w różnych kierunkach w celu odnalezienia pilota. Celem agenta będzie odnalezienie pilota i zrzucenie mu zaopatrzenia oraz powrót do miejsca startu. Agent za pomocą swoich czujników będzie wyczuwał zagrożenie (radary i broń) na kilka kratek w przód, aby móc je ominąć i dotrzeć do celu.

Obserwacje i akcje agenta będą następujące:

* Obserwacje: [ zagrożenie, unieruchomienie, zmniejszenie prędkości, obserwacja i ewentualna korekta wysokości ]

- Agent wyczuwać będzie zagrożenie na kilka kratek w przód w każdym kierunku (będzie to uzależnione od podanego parametru)

- W przypadku gdy agentowi nie uda się ominąć radaru i jego tor ruchu będzie przechodzić przez kratkę z radarem będzie on unieruchomiony

- Gdy agent zostanie trafiony strzałą odczuwa zmniejszenie prędkości

- Gdy agent znajdzie się nad pilotem sprawdza swoją wysokość i jeśli nie jest na odpowiedniej wysokości to wykonuje korektę wysokości poziomu 0.1H

* Akcje: { RuchWPrzód, RuchWTył RuchWLewo, RuchWPrawo, RuchWGóre, RuchWDół, Upuścić, Zginąć }

- ,,Upuścić” zaopatrzenie, jeśli dron znajdzie się nad pilotem

- ,,Zginąć” agent zginie, jeśli zostanie trafiony przez dwie strzały z rzędu

Dla pełnego wyjaśnienia schematu postępowania w przypadku algorytmu wnioskowania w przód wprowadzę następujące pojęcia:

**Baza wiedzy** składa się z reguł i faktów.

**Reguła** to wyrażenie postaci np. jeśli A to B. ( A => B), gdzie A nazywamy **założeniem**, natomiast B(**wnioskiem**/konkluzją)

**Fakty** - są to wyrażenia logiczne, którym przypisano wartość logiczną.

Fakty mogą być **dopytywalne** bądź **niedopytywalne**. Fakty niedopytywalne to są takie, które nie występują jako wniosek.

**Predykat** wskazuje na konkretny obiekt( są to np. założenia, wnioski, fakty).

Dla przykładowej bazy wiedzy:

A => B

B => C

C jest faktem **niedopytywalnym** bo zawiera wniosek a z kolei B jest faktem **dopytywalnym** bo nie wynika z żadnej innej reguły.

Przy stworzeniu uniwersalnego algorytmu wnioskowania w przód będą wykorzystywane wprowadzone powyżej pojęcia:

* fakty **niedopytywalne**
* fakty **dopytywalne**

Dla przykładowej bazy wiedzy:

A => D

F, H => G

B => L

D, J => M

C, D => F

A, E => J

Musimy na początku ustalić, które fakty są dopytywalne, a które nie:

|  |  |
| --- | --- |
| **Fakty dopytywalne** | **Fakty niedopytywalne** |
| A | D |
| E | G |
| B | L |
| H | M |
| C | F |
|  | J |

Algorytm będzie po kolei przechodził przez każdą z reguł w bazie wiedzy i pobierał te predykaty, dla których znana jest wartość logiczna i mamy pewność że wartość ta się nigdy nie zmieni.

Warunki, jakie należy spełnić, aby móc zastosować dany algorytm:

- Trzeba znać wartości logiczne faktów dopytywalnych

- Będziemy ,,przeskakiwać” po kolejnych regułach (w przypadku braku pobrania kolejnego predykatu wartość pobrana z poprzedniej reguły przechodzi do kolejnej bez zmiany)

- Warunek stopu (pętla do while w której warunkiem jest to że nie dopiszemy żadnego predykatu po sprawdzeniu wszystkich reguł sprawdzając je kolejno w bazie wiedzy ).

Iteracja przejścia przez algorytm dla przykładowej bazy danych

|  |  |
| --- | --- |
| Krok | Wartość pobrana |
| A -> D | D |
| F, H -> G | (brak) a zatem zostaje D |
| B -> L | D L |
| D, J -> M | D L |
| C, D -> F | D L F |
| A, E -> J | D L F J |
| --------------------------------------------Kolejna iteracja----------------------------------------------- | |
| A -> D | D L F J |
| F, H -> G | D L F J G |
| B -> L | D L F J G |
| D, J -> M | D L F J G M |
| C, D -> F | D L F J G M |
| A, E -> J | D L F J G M |

Wnioskowanie wymaga 2 założeń

1. założenie zamkniętego świata (baza wiedzy, która zostanie stworzona nie będzie dynamiczna (nie będzie można dodać reguł podczas wnioskowania))

2. zasada poprawnego wnioskowania (reguła A =>B (jest spełniona dla par A,B))

|  |  |
| --- | --- |
| A | B |
| 1 | 1 |
| 0 | 1 |
| 0 | 0 |

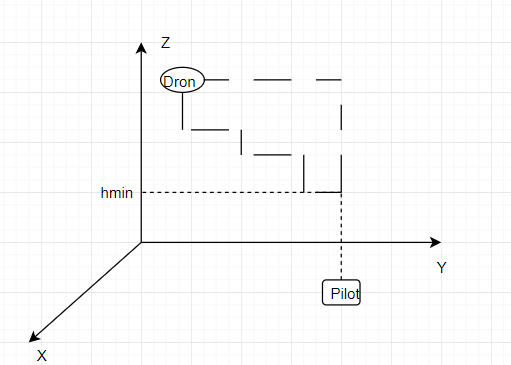
Przy założeniu dwóch wymagań jednocześnie (1 i 2) pozostaną nam pary **A B**

|  |  |
| --- | --- |
| A | B |
| 1 | 1 |
| 0 | 0 |

Wynika to z teorii wnioskowania. Na jedno zapytanie musimy otrzymać jedną odpowiedź.

Początkowo algorytm wnioskowania w przód będzie starał się znaleźć jak najkrótszą ścieżkę.

Poniżej przedstawiono przykładową sytuację:



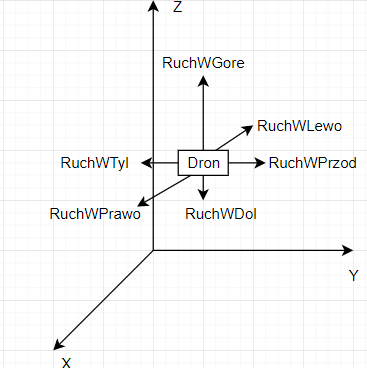
Liniami przerywanymi prowadzącymi od pilota został zaznaczony poziom hmin z którego możliwe jest zrzucenie zaopatrzenia przez agenta. Linie przerywane prowadzące od drona to przykładowe ruchy drona, które spowodują że znajdzie się on na określonym poziomie hmin. Istnieją co najmniej dwa sposoby podejścia do wyznaczenia najkrótszej ścieżki:

1. Agent(dron) dąży do tego aby znaleźć się nad dronem i wtedy zmniejsza swoją wysokość, aby zrzucić mu zaopatrzenie
2. Dron najpierw zniży się do poziomu hmin = 0.1H a następnie będzie próbował dostać się do pilota

Przez ścieżkę będę miał na myśli sumaryczną liczbę ruchów jakie musi wykonać dron, aby przedostać się ze swojego położenia początkowego do pilota. W realizacji projektu wybrano 1 podejście. W przypadku gdy na najkrótszej ścieżce napotka jakieś przeszkody (radar, broń) ścieżka ta będzie modyfikowana z wykorzystaniem rachunku sytuacyjnego tak, aby uniknąć zagrożeń.

Rachunek sytuacyjny stworzony zostanie za pomocą języka predykatów wykorzystując indeksowanie symboli.

**Założenie początkowe**: dron będzie skierowany w kierunku dodatniej osi Y



Ruch w kierunku osi X będzie to ruch w prawo lub w lewo

Ruch w kierunku osi Y będzie to ruch w przód lub w tył

Ruch w kierunki osi Z będzie to ruch w górę lub ruch w dół. Oś Z będzie wyznaczać wysokość.

Wprowadzam założenie symbolu Lx,y,z dla oznaczenia, że dron jest w 3-wymiarowym środowisku i znajduje się w przestrzeni o współrzędnych [x,y,z]. Przykładowy zbiór języka predykatów:

L1 x,y,z  RuchwLewo1  L2 x-c,y,z

L1 x,y,z  RuchwPrawo1  L2 x+c,y,z

L1 x,y,z  RuchwGórę1  L2 x,y,z+c

L1 x,y,z  RuchwDół1  L2 x,y,z-c

L1 x,y,z  RuchwPrzód1  L2 x,y+c,z

L1 x,y,z  RuchwTył1  L2 x,y-c,z

(c – parametr określony przez prędkość)

Bazą wiedzy będzie określony zestaw reguł, które będą zmieniać swoją wartość logiczną (True/False) w zależności od aktualnego położenia drona i możliwego do wykonania przez niego ruchu.

OGÓLNA KONCEPCJA i ZAŁOŻENIA PROGRAMU

Program na początku swojego działania będzie pobierał od użytkownika informację o środowisku - parametry N i H. Środowisko na początku generowane będzie jako przestrzeń wypełniona zerami.

environment = zeros(N, N, H); %inicjalizacja środowiska o wymiarach NxNxH

Rozmieszczenie radarów i broni (prostopadłościan o określonej wysokości, głębokości, szerokości) będzie zainicjalizowane jako :

environment(3:5, 4:6, 5:7) = 'r'; %(od 3 do 5 komórki osi X, od 4 do 6 komórki osi Y, od 5 do 7 komórki osi Z) %wstawienie do pól konkretnych zakresów litery 'r', która będzie oznaczać, że w tych miejscach znajdować się będzie radar

Program przy pomocy informacji z pliku tekstowego dotyczących położenia początkowego drona oraz jego parametrów będzie generował środowisko, a także położenie początkowe drona (agenta) i zagrożeń (radary i broń).

Agent(dron) będzie strukturą, która jako parametry będzie przechowywać swoje położenie początkowe, energię, zasięg czujników, prędkość, położenie pilota, informację czy osiągnięto pozycję pilota oraz informację o tym czy udało się agent powrócić do miejsca startu.

Główną pętlą symulacyjną będzie pętla while:

while( drone.energy > 0 && drone.if\_return\_to\_start != true )%dopóki energia drona jest >0 i nie powrócił do pozycji początkowej

cube\_to\_pass = environment( drone.position.x - drone.sensors\_range:drone.position.x + drone.sensors\_range

drone.position.y - drone.sensors\_range:drone.position.x + drone.sensors\_range

drone.position.z - drone.sensors\_range:drone.position.x + drone.sensors\_range)%zapisanie wycinka środowiska, który będzie nas interesował, przy analizie kolejnego jego ruchu

predicates = detect\_danger(drone.position, cube\_to\_pass)%wykryj zagrożenie aktualizujemy wartosci predykatów za pomocą parametru pozycji drona oraz drogi do przejścia

%początek tworzenia bazy wiedzy (odpowiada to funkcji TELL)

move\_to\_make, drone\_state = inference(predicates, drone);

% funkcja odpowiedzialna za pobranie aktualnych predykatów oraz parametrów drona (przypisuje do zmiennych ruch do zrobienia i aktualny położenie drona)-odpowiada to funkcji ASK pytamy sie czy jesteśmy pod działaniem radaru bądź broni

% inference cały algorytm poruszania się

%musimy mieć informacje gdzie jesteśmy i dokąd zmierzamy

drone = move\_drone(drone, move\_to\_make); %funkcja odpowiedzialna za poruszanie dronem

%przyjmuje informacje o dronie oraz ruchu do przejścia (aktualizacja parametrów drona)

drone = take\_drone\_energy(drone, move\_to\_make); %funkcja odpowiedzialna za pobranie energii drona(ile jej ubyło po wykonaniu ruchu)

moves\_and\_states.moves(simulation\_time) = move\_to\_make;

%zbieranie wyników dla czasu symulacji w wektorze (ruchy) aby później móc je wyświetlić

moves\_and\_states.states(simulation\_time) = drone\_state;

%zbieranie wyników dla czasu symulacji w wektorze (stany) aby później móc je wyświetlić

simulation\_time = simulation\_time + 1; %inkrementacja czasu symulacji po każdym przejściu pętli

end

Analiza rezultatów będzie odbywać się poza główną pętlą:

plot\_simulation(environment, drone, moves\_and\_states); % prezentacja wyników po zebraniu symulacji

analyze\_results(drone, simulation\_time); %analiza rezultatów (dron w czasie)

W każdym kroku uaktualniana będzie informacja o możliwym położeniu agenta po wykonaniu ruchu, aby ograniczyć sprawdzenie położenia radarów i broni do wycinka środowiska). Reguły w bazie wiedzy będą zapisane od początku istnienia programu i zbiór tych reguł nie zmieni się. Każdorazowo uaktualniana będzie o wartościach logicznych predykatów znajdujących się w tej bazie (sprawdzane to będzie co turę (krok symulacji)). Zrealizowane to będzie za pomocą funkcji sprawdzającej czy w następnym kroku symulacji występuje zagrożenie. Wartości predykatów będą aktualizowane za pomocą parametru pozycji drona oraz drogi do przejścia. Będzie to odpowiednik funkcji TELL w bazie wiedzy. Funkcję ASK będzie odzwierciedlać funkcja odpowiedzialna za pobranie wartości logicznych predykatów z bazy wiedzy oraz parametrów drona, dzięki którym możliwe będzie przypisanie do zmiennych ruchu jaki wykona dron oraz jego aktualne położenie. Wyniki poszczególnych obserwacji zbierane będą do wektorów, dzięki czemu po wyjściu z pętli będzie możliwe wyświetlenie poszczególnych kroków symulacji.