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:

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, obniżenie 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 obniża swoją wysokość

* Akcje: { RuchWPrzód, RuchWTył RuchWLewo, RuchWPrawo, RuchWBok, 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 brak 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 poruszając się z góry na dół po wszystkich regułach)

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

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

- reguła A =>B (jest spełniona dla par A,B):

|  |  |  |
| --- | --- | --- |
| A | B | A=>B |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 0 | 0 |

Początkowo algorytm wnioskowania w przód będzie starał się znaleźć jak najkrótszą ścieżkę do celu (pilota). W przypadku gdy na najkrótszej ścieżce napotka jakieś przeszkody (radar, broń) ścieżka ta będzie modyfikowana za pomocą rachunku sytuacyjnego tak, aby uniknąć zagrożeń.

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

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  ZwroconyZgodniezX1 RuchwLewo1  L2 x,y+c,z

L1 x,y,z  ZwroconyZgodniezX1 RuchwPrawo1  L2 x,y-c,z

L1 x,y,z  ZwroconyZgodniezX1 RuchwGórę1  L2 x+c,y,z

L1 x,y,z  ZwroconyZgodniezX1 RuchwDół1  L2 x-c,y,z

(c – parametr określony przez prędkość)

Wyjątkiem od takich standardowych reguł będą sytuacje w których dron znajduje się na granicy którejś z osi, czy też znajduje się w przestrzeni radaru i zostanie unieruchomiony.

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 jest jako przestrzeń wypełniona zerami.

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

Rozmieszczenie radarów i broni będzie zainicjalizowane jako :

environment(3:5, 4:6, 5:7) = 'r'; %(od 3 do 5, od 4 do 6, od 5 do 7) %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 będzie generował środowisko, a także położenie początkowe drona (agenta) i niebezpieczeństw (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 predykatow za pomocą parametru pozycji drona oraz drogi do przejścia%początek tworzenia bazy wiedzy (odpowiada to funkcji TELL - baza%wiedzy będzie się zmieniać w każdym kroku symulacji

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 jestemy pod działaniem radaru,%czy gun

% inference caly algorytm poruszania sie%musimy miec informacje gdzie jestesmy i dokad 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

Za każdym razem po podaniu parametru zasięg czujników drona 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 tworzone w oparciu o rozpoznanie w którym miejscu aktualnie znajduje się agent (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 aktualnych predykatów z bazy wiedzy oraz parametrów drona, które to przypisywać będą do zmiennych jaki ruch wykona dron oraz jego aktualne położenie.