Umjetna inteligencija – Laboratorijska vježba 2

UNIZG FER, ak. god. 2016/17.

Zadano: 28.3.2017. Rok predaje: 9.4.2017. do 23.59 sati.

Uvod

U ovoj laboratorijskoj vježbi pomoći ćete agentu Pacardu pronaći put do cilja labirinta zlog GhostWumpusa. Rješavati ćete dva problema, jedan od kojih je rezolucija opovrgavanjem, a drugi pretraživanje mape potpomognuto rezolucijom. Implementirati ćete algoritam rezolucije te logiku pretraživanja labirinta GhostWumpusa.

Kod koji opisuje ponašanje Pacardovog svijeta već je napisan i funkcionalan, te je na vama samo zadatak implementirati algoritme koji će kontrolirati Pacardovo kretanje te logičko razmišljanje. Kod koji ćete koristiti možete skinuti kao zip arhivu na stranicama fakulteta ovdje, ili na github repozitoriju predmeta ovdje.

Kod za projekt se sastoji od Python datoteka, neke od kojih ćete trebati pročitati i razumjeti za implementaciju laboratorijske vježbe, neke koje ćete trebati samostano uređivati te nekih koje ćete moći ignorirati. Nakon raspakiranja zip arhive, opis datoteka i direktorija koje ćete vidjeti je u nastavku:

Datoteke koje ćete uređivati:	
pacard.py	Datoteka u kojoj ćete implementirati kretanje Pacarda
logic.py	Datoteka u kojoj ćete implementirati rezoluciju opovrgavanjem
Datoteke koje trebate proučiti:	
logicAgents.py	Datoteka koja sadrži agente bazirane na algoritmima pretraživanja
pacman.py	Glavna datoteka koja pokreće igru
game.py	Logika iza Pacmanovog svijeta
util.py	Korisne strukture podataka za impementaciju rješenja
commands.txt	Naredbe za konzolu u txt formatu
Pomoćne datoteke koje možete ignorirati:	
graphicsDisplay.py	Pacmanovo grafičko sučelje
graphicsUtils.py	Pomoćne funkcije za grafičko sučelje
textDisplay.py	ASCII grafike za Pacmana
ghostAgents.py	Agenti koji kontroliraju duhove
keyboardAgents.py	Sučelje za kontroliranje Pacmana pomoću tipkovnice

U okviru ove laboratorijske vježbe neće biti autogradera, te vas potičemo da smislite vlastite testove za vaše implementacije. Na dnu *logic.py* postoji osnovni test rezolucije opovrgavanjem, dok je mini-mapa za testiranje pretraživanja definirana u *layouts/miniWumpus.lay*. Veće mape možete samostalno napisati tako da napravite novu datoteku s ekstenzijom *.lay* te mu predate ime kao argument pacmanu - *-l mojWumpusLayout*.

Verziju mape koju kontrolirate ručno pokrećete sa:

```
python pacman.py -l miniWumpus -g WumpusGhost
```

Dok verziju mape u kojoj je implementirana funkcija pretraživanja miniWumpusSearch pokrećete pomoću:

```
python pacman.py -l miniWumpus -p PacardAgent -a fn=miniWumpusSearch -g WumpusGhost
```

Testirajte vašu implementaciju logike kretanja Pacarda pomoću naredbe:

```
python pacman.py -l miniWumpus -p PacardAgent -a fn=logicBasedSearch -g WumpusGhost
```

Laboratorijska vježba ukupno nosi 6.25 bodova, od kojih je podjednaki udio (3.125) na prvom podzadatku implementacije rezolucije te na drugom podzadatku implementacije pretraživanja.

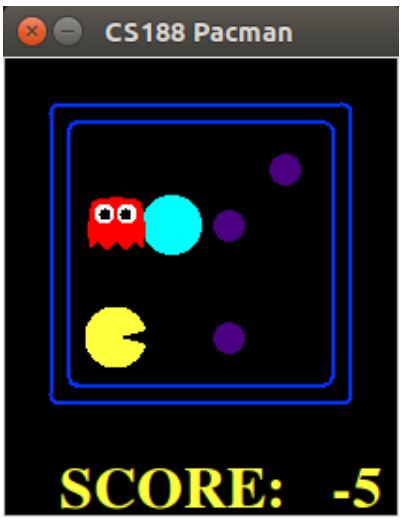
Pacman GhostWumpus problem propozicijske logike

Kao što ste upoznati, davnih godina Umjetne Inteligencije kapetan Jean-Luc Picard je muku mučio na planetu Wumpusworld, gdje je čudovište Wumpus the Hutt teleportiralo Picarda u svoju pećinu i potom ga probalo pojesti. Zahvaljujući studentima prethodnih godina, Picard je uspio pobjeći zlom Wumpusu, te je prošao kroz mnoge avanture do duboko u svoju starost. Kroz jednu od tih avantura prošao je planetom Pacearth, na kojemu su ga tamošnji nezasitni žitelji Pacpeople primili kao jednog od svojih, te mu dali počasno ime 'Pacard'.

Ono što Picard nije znao je da su Pacpeople iznimno znanstveno razvijeni usprkos njihovoj primitivnoj vanjštini, te su poskrivećki uzeli Picardov DNA uzorak te kombinacijom s Paco Hungreus kromosomima stvorili polu Pacmana te polu Picarda, kojeg su nazvali Lean-Juc Pacard. No, kroz ove eksperimente uzrokovali su anomaliju u prostorvremenu koja je odvojila dio duše zloglasnog Wumpusa te ga povezala sa zakletim neprijateljima Pacpeoplea, duhovima. Kao što naslućujete, GhostWumpus je nastao - te ne naučivši ništa iz prethodnog pokušaja, pokušao zarobiti Pacarda u pećini na planetu GhostWorld.

Lean-Juc Pacard našao se u nevolji. Koristeći duhovsku magiju, GhostWumpus ga je teleportirao u svoju pećinu u kojoj je prethodno posložio otrovne kapsule znajući da je najveća slabost Pacarda njegova proždrljivost. No, nije računao na istančan osjet mirisa mladog Pacarda niti na to da će ga teleportacijska magija izmoriti do te mjere da se više ne može kretati. Na sreću, uspio je zagasiti svijetlo u svojoj pećini na vrijeme - te jedina stvar na koju se Pacard može osloniti su svoji osjeti mirisa te dodira. Na nesreću, Picard je za vrijeme svog boravka poklonio Pacpeople teleportersku tehnologiju, te su oni otkrili lokaciju GhostWumpusove pećine i otvorili izlaz teleportera na nasumičnom mjestu u njoj. Pomognite Pacardu da kroz izbjegavanje otrovnih kapsula te GhostWumpusa dođe sigurno natrag do Paceartha.

Pećina u kojoj se Pacard nalazi može se prikazati 2D rešetkom veličine N×M pri čemu je svako polje rešetke označeno točno jednom od četiri moguće oznake: *"W"* (polje na kojem se nalazi čudovište GhostWumpus), *"P"* (*poison*, polje na kojem se nalazi otrovna kapsula), *"T"* (polje na kojem se nalazi teleporter) te *"O"* (obično polje na kojem nisu ni GhostWumpus ni otrovne kapsule ni teleporter). Nadalje, na svim susjednim poljima od polja na kojem se nalazi GhostWumpus (susjednim poljima smatraju se ona koja dijele stranicu rešetke, a ne vrh) osjeti se značajan smrad od Wumpusovske strane GhostWumpusa, dok se na svim poljima koja su susjedna poljima s otrovnim kapsulama osjeti miris



Slika 1: GhostWumpusova pećina

kemikalija. Polje s teleporterom emitira blagu svjetlost na svoja susjedna polja, no samo polje na kojem se nalazi teleporter ne svijetli. Primjer izgleda Wumpusove pećine prikazan je na slici 1.

Vaš zadatak je, primjenom razrješavanja opovrgavanjem u propozicijskoj logici, osigurati da Pacard dođe do teleportera, a da pritom zaobiđe GhostWumpusa i otrovne kapsule (što ovisno o konfiguraciji pećine može, ali i ne mora biti moguće). U svakom koraku Pacard na temelju spoznaja o prethodno posjećenim poljima te na temelju informacija o smradu wumpusa i mirisu kemikalija na danome polju pokušava zaključiti oznake za sva susjedna polja, u čemu može i ne mora uspjeti jer u nekom trenutku na temelju poznatih činjenica ne mora biti moguće utvrditi oznaku nekog polja. Prijelaz s jednoga polja na drugo Pacard obavlja na temelju sljedećih pravila (pravila su poredana silazno po prioritetu):

1. Ukoliko postoji susjedno polje za koje je izvedena (ili prethodno poznata) oznaka *"T"*, Pacard prelazi na polje s teleporterom i vraća se na Pacearth;

2. Ukoliko postoji susjedno polje za koje je izvedena (ili prethodno poznata) oznaka *"O"* (ekvivalentno je da za to polje nije izvedeno ni *"T"* ni *"P"* ni *"W"*), Pacard prelazi na to sigurno polje. Ukoliko postoji više takvih sigurnih susjednih polja Pacard odabire ono s najmanjom pozicijom (ako je pozicija polja predstavljena uređenim parom (x, y) onda prelazi na susjedno polje za koje je vrijednost $20x + y$ najmanja);

3. Ukoliko postoji susjedno polje za koje na temelju trenutnog znanja nije moguće zaključiti oznaku, Pacard prelazi na takvo stanje. Ako je takvih polja više, prelazi na ono s najmanjom pozicijom;

4. Pacard shvaća da se ne smije nigdje pomaknuti (ako nijedno od prethodnih pravila nije zadovoljeno to znači da je okružen isključivo poljima koja znače sigurnu smrt – otrovne kapsule i GhostWumpusovo polje) te ostaje na trenutnome polju čekati spas s Pacearth.

Modul za razrješavanje opovrgavanjem u propozicijskoj logici potrebno je implementirati neovisno o primjeni na problem GhostWorlda, kako je opisano u nastavku teksta.

Razrješavanje opovrgavanjem

Možete pretpostaviti da će formule s kojima ćete raditi rezoluciju već biti svedene na konjunktivnu normalnu formu odnosno da ne trebate implementirati svođenje formula na takav oblik.

Implementirajte dokazivač teorema u propozicijskoj logici temeljen na postupku rezolucije opovrgavanjem. Kao rezolucijsku strategiju treba koristiti strategiju skupa potpore. Funkcija kao ulaz uzima skup premisa $F_1, \ldots, F_n$ i ciljnu formulu G, a kao izlaz vraća *True* ako je ciljna formula dedukcija premisa, a *False* inače. Obratite pažnju na faktorizaciju koju uvijek treba provoditi nad svakom rezolventom. U svakom slučaju treba obratiti pozornost na to da se jedan te isti par klauzula ne razrješava više puta, kao i na to da se ne ponavlja generiranje već postojećih klauzula.

Potrebno je također implementirati jednostavnu strategiju pojednostavljenja kojom će se nakon svake primjene rezolucijskog pravila iz skupa klauzula uklanjati redundantne i nevažne klauzule. Uklanjanje redundatnih klauzula temelji se na ekvivalenciji apsorpcije $F \wedge (F \vee G) \equiv F$. Ako su klauzule prikazane kao skupovi literala, čim se u skupu klauzula nađe par klauzula C_1 i C_2, takvih da $C_1 \subseteq C_2$, klauzula C_2 može se ukloniti. Uklanjanje nevažnih klauzula svodi se na uklanjanje svih klauzula koje su valjane formule. Klauzula je valjana ako i samo ako sadrži komplementaran par literala.

Povezivanje Wumpusworlda i razrješavanja opovrgavanjem

Napišite funkciju-omotač kojoj je moguće zadati skup premisa i ciljnu formulu (u CNF obliku), a koja vraća informaciju o tome je li cilj moguće dokazati iz premisa. Tu funkciju omotača ćete pozivati iz programa za navigaciju Pacarda po pećini kad god trebate utvrditi oznaku za neko polje.

Svako polje pećine (x, y) moguće je opisati sa sedam literala propozicijske logike:

- $S_{(x,y)}$ – na polju (x, y) osjeti se smrad (*stench*) čudovišta GhostWumpus;
- $C_{(x,y)}$ – na polju (x, y) osjeti se miris otrova (*chemicals*);
- $G_{(x,y)}$ – na polju (x, y) uočava se svjetlost teleportera (*glow*);
- $W_{(x,y)}$ – na polju (x, y) nalazi se GhostWumpus;
- $P_{(x,y)}$ – na polju (x, y) nalazi se otrovna kapsula;
- $T_{(x,y)}$ – na polju (x, y) nalazi se teleporter;
- $O_{(x,y)}$ – polje (x, y) je sigurno.

Na temelju prethodno opisanih pravila funkcioniranja Wumpusova svijeta jednostavno izvodimo sljedeće formule propozicijske logike za svako polje pećine:

- Ako se na nekom polju osjeti smrad, onda se na nekome od susjednih polja nalazi GhostWumpus, tj. $S_{(x,y)} \rightarrow (W_{(x-1,y)} \vee W_{(x+1,y)} \vee W_{(x,y-1)} \vee W_{(x,y+1)})$;

- Ako se na nekom polju ne osjeti smrad, onda se ni na jednome od susjednih polja ne nalazi GhostWumpus, tj. $\neg S_{(x,y)} \rightarrow (\neg W_{(x-1,y)} \wedge \neg W_{(x+1,y)} \wedge \neg W_{(x,y-1)} \wedge \neg W_{(x,y+1)})$
- Ako se na nekom polju osjeti miris kemikalija, onda se na nekome od susjednih polja (moguće i na više od jednoga) nalazi otrovna kapsula, tj. $C_{(x,y)} \rightarrow (P_{(x-1,y)} \vee P_{(x+1,y)} \vee P_{(x,y-1)} \vee P_{(x,y+1)})$;
- Ako se na nekom polju ne osjeti miris kemikalija, onda se ni na jednome od susjednih polja ne nalazi otrovna kapsula, tj. $\neg C_{(x,y)} \rightarrow (\neg P_{(x-1,y)} \wedge \neg P_{(x+1,y)} \wedge \neg P_{(x,y-1)} \wedge \neg P_{(x,y+1)})$
- Ako se na nekom polju vidi svjetlost, onda se na nekome od susjednih polja nalazi teleporter, tj. $G_{(x,y)} \rightarrow (T_{(x-1,y)} \vee T_{(x+1,y)} \vee T_{(x,y-1)} \vee T_{(x,y+1)})$
- Ako se na nekom polju ne vidi svjetlost, onda se ni na jednome od susjednih polja ne nalazi teleporter, tj. $\neg G_{(x,y)} \rightarrow (\neg T_{(x-1,y)} \wedge \neg T_{(x+1,y)} \wedge \neg T_{(x,y-1)} \wedge \neg T_{(x,y+1)})$
- Ako se na nekome polju nalazi Wumpus, onda se Wumpus ne nalazi ni na jednome drugome polju, tj. $W_{(x,y)} \rightarrow (\neg W_{(x',y')})$ (za svaki (x', y') različit od (x, y)).
- Ako se na nekome polju ne osjeti smrad Wumpusa niti miris kemikalija, onda su sva polja oko njega sigurna (*"O"*) , tj. $(\neg C_{(x,y)} \wedge \neg S_{(x,y)}) \rightarrow (O_{(x-1,y)} \wedge O_{(x+1,y)} \wedge O_{(x,y-1)} \wedge O_{(x,y+1)})$
- Ako na nekom polju nije Wumpus niti otrovna kapsula, onda je to polje sigurno ((*"O"*)), tj. $(\neg W_{(x,y)} \wedge \neg P_{(x,y)}) \rightarrow (O_{(x,y)}$

Navedena pravila možete ručno pretvoriti u CNF. Ovo je samo osnovni skup pravila koje možete koristiti za zaključivanje. Slobodni ste (ali ne i obavezni) napisati još formula koje mogu pospješiti zaključivanje, a koja su u skladu s pravilima Wumpusove pećine. Razmislite, na primjer, što se može zaključiti iz činjenice da se na dvama poljima koja dijele vrh osjeti GhostWumpusov smrad (a znate da se GhostWumpus nalazi na točno jednom polju na mapi).

Dodatne upute, pojašnjenja i primjeri izvođenja

U drugom zadatku, potrebno je pomoći Jean-Luc Pacardu da dođe do teleportera pomoću pravila propozicijske logike. Unutar zadatka, zadan nam je skup formula propozicijske logike koja vrijede za svako polje pećine u kojoj se nalazimo – nešto što možemo smatrati *ponašanjem svijeta* u kojem se nalazimo. Da bi primjenili za pravila, doduše, potrebni su nam još neki podaci, a te podatke dobivamo kroz navigaciju po pećini. Jean-Luc, naime, na svakom koraku može koristiti svoja osjetila kako bi došao do idućih informacija – ukoliko se nalazi na polju (x, y), može zaključiti $S_{(x,y)}$ ili $\neg S_{(x,y)}$, $C_{(x,y)}$ ili $\neg C_{(x,y)}$, te $G_{(x,y)}$ ili $\neg G_{(x,y)}$.

Pomoću tih informacija, koje kad su jednom otkrivene vrijede zauvijek (te se mogu spremiti u neku *bazu znanja*), možemo zaključiti nešto o svim sujednim poljima pomoću pravila svijeta (hint: radi jednostavnosti, koristite metodu *"getSuccessors()"* pri generiranju susjednih polja, a prema tome, i klauzula). Osam zaključaka koje želimo dokazati za **svako susjedno stanje** su:

1. Nalazi li se na njemu GhostWumpus ili ne ($W_{(x,y)}$, $\neg W_{(x,y)}$)?
2. Nalazi li se na njemu otrovna kapsula ili ne ($P_{(x,y)}$, $\neg P_{(x,y)}$)?

3. Nalazi li se na njemu teleporter ili ne ($T_{(x,y)}$, $\neg T_{(x,y)}$)?
4. Je li to polje sigurno ili ne ($O_{(x,y)}$, $\neg O_{(x,y)}$)?

Primjetite također da je redoslijed provjere bitan, budući da znanje koje bismo izveli (i spremili u *bazu znanja*) u prva tri para klauzula direktno utječe na ishod četvrtog para (stanje je sigurno samo ako na njemu nema niti Wumpusa niti otrovne kapsule).

U nastavku ćemo proći kroz primjer izvođenja programa za mapu sa Slike 1 uz detaljniji ispis.

```
Visiting:  (1, 1)
Sensed:  ~s(1, 1)
Sensed:  ~b(1, 1)
Sensed:  ~g(1, 1)
Concluded:  ~w(1, 2)
Concluded:  ~p(1, 2)
Concluded:  ~t(1, 2)
Concluded:  o(1, 2)
Concluded:  ~w(2, 1)
Concluded:  ~p(2, 1)
Concluded:  ~t(2, 1)
Concluded:  o(2, 1)
```

U stanju $(1, 1)$ ne osjetimo ništa s osjetilima, te zaključujemo da su sve susjedne ćelije prazne. Dodajemo stanja $(1, 2)$ i $(2, 1)$ u naš popis sigurnih stanja, te kao iduće stanje posjećujemo stanje $(1, 2)$ radi prioriteta.

```
Visiting:  (1, 2)
Sensed:  s(1, 2)
Sensed:  ~b(1, 2)
Sensed:  ~g(1, 2)
Concluded:  ~p(1, 3)
Concluded:  ~t(1, 3)
Concluded:  ~p(2, 2)
Concluded:  ~t(2, 2)
```

U stanju $(1, 2)$ osjetimo smrad GhostWumpusa, te iako ne možemo zaključiti na kojem stanju se nalazi GhostWumpus, znamo da se na nijednom susjednom stanju ne nalazi niti teleporter niti otrovna kapsula. Susjedna stanja dodajemo u listu nesigurnih stanja, budući da za njih nismo uspjeli dokazati niti da su sigurna niti da su nesigurna. Još uvijek imamo jedno sigurno stanje, te prema tome kao iduće stanje posjećujemo $(2, 1)$. Primjetite da iako stanja $(1, 2)$ i $(2, 1)$ nisu susjedi, možemo slobodno prelaziti između njih – čim smo u jednom trenutku mogli doći do stanja $(2, 1)$, znamo da postoji način da se vratimo tamo. Metoda "reconstructPath()" će se pobrinuti za put između stanja.

```
Visiting:  (2, 1)
Sensed:  ~s(2, 1)
Sensed:  b(2, 1)
Sensed:  ~g(2, 1)
Concluded:  ~w(2, 2)
```

```
Concluded:  o(2, 2)
Concluded:  ~w(3, 1)
Concluded:  p(3, 1)
Concluded:  ~t(3, 1)
```

U stanju $(2, 1)$ osjetimo miris kemikalija, te znamo da je na jednom od susjednih polja otrovna kapsula. No, ne osjetimo smrad GhostWumpusa, te uspješno zaključujemo da na niti jednom od susjednih polja nema GhostWumpusa! Ovo nam je riješilo problem koji smo imali s poljem $(2, 2)$, za koje smo u prethodnom koraku bili sigurni da se na njemu ne nalazi otrovna kapsula, no nismo znali nalazi li se na njemu GhostWumpus. Dodajemo stanje $(2, 2)$ u popis sigurnih stanja, te ćemo ga kao jedino sigurno stanje iduće posjetiti.

Primjetite da smo u ovom koraku mogli (uz dodatne klauzule) naknadno zaključiti da se na stanju $(1, 3)$ nalazi GhostWumpus budući da su sva ostala okolna stanja stanja $(1, 2)$ sigurna, a na njemu smo osjetili smrad GhostWumpusa. Ekvivalentno možemo zaključiti da se na stanju $(3, 1)$ nalazi otrovna kapsula, no ovo nije nužno za izvedbu laboratorijske vježbe.

```
Visiting:  (2, 2)
Sensed:  ~s(2, 2)
Sensed:  ~b(2, 2)
Sensed:  g(2, 2)
Concluded:  ~w(2, 3)
Concluded:  ~p(2, 3)
Concluded:  o(2, 3)
Concluded:  ~w(3, 2)
Concluded:  ~p(3, 2)
Concluded:  o(3, 2)
```

Posjećujemo stanje $(2, 2)$, te na njemu samo vidimo sjaj teleportera. Kako nam teleporter nije opasan, već želimo doći do njega (ispravak greške iz prethodnih klauzula), zaključujemo da su sva okolna stanja sigurna, te kao iduće stanje prelazimo na $(2, 3)$ radi prioriteta.

```
Visiting:  (2, 3)
Game over: Teleported home!
```