Predmet: "Vještačka inteligencija"
Laboratorijska vježba 7: Ekspertni sistemi (1/2)
Odgovorna nastavnica: Vanr. prof. dr Amila Akagić

Sadržaj vježbe:

1 Cilj vježbe

Kroz ovu (i narednu) vježbu studenti će se upoznati sa PyKE-om, interpreterom ekspertnih sistema baziranom na programskom jeziku Python, kroz nekoliko jednostavnih primjera ekspernih sistema.

2 PyKE

Python Knowledge Engine (skaćeno PyKE) se tipično definiše kao interpreter za ekspertne sisteme (eng. *expert system interpreter*) i predstavlja jednostavno okruženje koje olakšava kreiranje i pokretanje jednostavnih ekpertnih sistema. PyKE integriše logičko programiranje sa programskim jezikom Python. Više detalja o PyKE-u može se pročitati na sljedećem linku (studentima se preporučuje da pročitaju dokumentaciju dostupnu na ovom linku). Pored PyKE-a, postoje i neki drugi interpreteri ekspertnih sistema bazirani na Python programskom jeziku (npr. PyCLIPS, koji radi samo u Python 3). Inače, vjerovatno najčešće korišteno okruženje za ekspertne sisteme zove se CLIPS i radi se o interpreteru baziranom na C programskom jeziku (zaintersosvani se upućuju na sljedeći link).

2.1 Instalacija

Za instaliranje PyKE okruženja na svom računaru potrebno je da slijedite niz koraka prikazanih u nastavku:

1. Preuzmite `pyke3-1.1.1.zip` datoteku sa sljedeće lokacije.
2. Nakon preuzimanje, raspakujte preuzetu `pyke3-1.1.1.zip` datoteku.
3. U folderu koji ste dobili nakon što je raspakovana datoteka otvorite terminal (*command line*) i pokrenute dvije komande prikazane ispod:

```
python setup.py build
python setup.py install
```

U folderu koji ste dobili nakon što ste raspakovali preuzetu datoteku nalazi se folder sa primjerima - `examples`. Unutar foldera imate nekoliko različitih primjera jednostavnih ekspetnih sistema i oni će biti korišteni za demonstraciju osnovnih principa tokom ove vježbe. Od posebnog interesa će biti primjer sa relacijama u porodici (`family_relations` folder), s obzirom da je on (kao primjer iz realnog života) najlakši za razumjeti.

2.2 Iskazi u PyKE-u

Iskaz (eng. *statement*) u PyKE-u se definiše kao generalizovani izraz znanja i osnova je za definisanje baze znanje ekspertnog sistema koristeći PyKE. Iskaz se sastoji od tri dijela:

- ime baze znanja,
- ime entiteta znanja (koje tipično definiše vezu između argumenata) i
- argumenti iskaza (koji mogu biti tipovi podataka u Python-u - brojevi, stringovi, True/False ili *tupples* ovih vrijednosti).

Primjer jednog iskaza, iz eskpertnog sistema za porodične relacije, dat je u nastavku:

```
family.son_of(Bruce, Thomas, Norma)
```

U ovom primjeru `family` predstavlja ime baze znanja, ime entiteta (relacija) je `son_of`, a argumenti su Bruce, Thomas i Norma (stringovi). Jasno je da je redolijed argumenata itekako bitan i da ga je potrebno unaprijed utvrditi kako bi bilo jasno ko, u ovom konkretnom primjeru, predstavlja dijete, a ko su roditelji.

2.3 Baza znanja

Baza znanja (eng. *knowledge base*) u PyKE-u okruženju se definiše kroz baze činjenica (eng. *fact bases*), baze pravila (eng. *rule bases*) i baza pitanja (eng. *question bases*). Činjenice i pravila se definišu kroz dvije posebne datoteke i to na načine koji će biti opisani u ovom dijelu vježbe, dok će način definisanja baze pitanja biti opisan na sljedećoj vježbi.

2.3.1 Baza činjenica

Baza činjenica se pohranjuje kao `.kfb` datoteka (pri čemu ekstenzija `.kfb` dolazi od skraćenice za *Knowledge fact base*). Unutar `family_relations` foldera nalazi se datoteka `family.kfb` u kojoj su pohranjene činjenice za ovaj ekspetni sistem i ta datoteka predstavlja bazu činjenica ovog jednostavnog ekspertnog sistema. U tom konkretnom primjeru, sve činjenice su pohranjene u formi `son_of` i `daughter_of`, nakon čega se specificiraju imena djeteta, oca i majke (npr. iz činjenice `daughter_of(holly, brian, shirley)` može se pročitati da je Holly kćerka od Brian-a i Shirley). Iz priloženog primjera jasno se vidi da je baza činjenica zapravo sve ono što je bitno da ekspertni sistem zna, pri čemu se činjenice specificiraju na neki unaprijed dogovoren način.

2.3.2 Baza pravila (pravila lančanja unaprijed)

Baza pravila se pohranjuje kao `.krb` datoteka (pri čemu ekstenzija `.krb` dolazi od skraćenice za `Knowledge rule base`). Kroz pravila se definiše na koji način će eskpertni sistem *konzumirati* činjenice pohranjene u bazi činjenica. Naime, jasno je da ekspertni sistem samo sa činjenicama ne bi imao nikavu upotrebnu vrijednost. Konkretno, na primjeru ekspertnog sistema za porodične relacije iz činjenica je moguće pročitati samo ko je kome sin, odnosno, kćerka, međutim, od eskpertnog sistema se očekuje da odgovori i na neka druga, kompleksnija pitanja, tipa da za određenu osobu izlista i njenu braću, sestre, djeda, baku itd. Upravo ta pravila koja će pomoći ekspertnom sistemu da osnovu činjenica dođe do konkretnih (i korisnih) zaključaka definišu se kroz bazu pravila.

Pravilo se sastoji od dvija dijela: `IF` dio, koji sadrži iskaz koji se naziva premisom i `THEN` dio, koji sadrži iskaz koji se naziva zaključkom. I `IF` i `THEN` dio se mogu sastojati od više činjenica, odnosno ciljeva (eng. *goals*). Pravilo u PyKE-u ima sljedeću sintaksu:

```
IF A, B, and C, THEN D and E.
```

Pravila u PyKE-u mogu biti pravila lančanja unaprijed (eng. *forward chaining rules*) i pravila lančanja unazad (eng. *backward chaining rules*), pri čemu jedna baza znanja može sadržavati pravila i jednog i drugog tipa lančanja.

Pravila lančanja unaprijed koriste `foreach` i `assert` ključne riječi umjesto `IF` i `THEN`, respektivno i funkcionišu na sljedeći način:

- PyKE posmatra prvo pravilo (njegov `IF` dio) i provjerava da li to pravilo zadovoljava neku od poznatih činjenica (pohranjenih u `.kfb` datoteci).
- Ukoliko pravilo zadovoljava poznatu činjenicu (ili više njih), prelazi na `THEN` dio tog pravila i dodaje novu činjenicu (zbog toga se `THEN` dio i specificira korištenjem `assert` ključne riječi). Inače, u literaturi se ovaj proces dodavanja novih činjenica naziva aktiviranjem pravila (eng. *firing the rule*).

- Ove novododane činjenice mogu da dovedu do toga da činjenice nekog drugog pravila budu zadovoljene (da neko drugo pravilo bude aktivirano) i ukoliko se to desi, ponavlja se korak 2.
- Proces se ponavlja sve dok postoje pravila koja se mogu aktivirati.

U primjeru ekspertnog sistema za porodične relacije može se naći baza pravila pohranjena u datoteku `family.kfb`. Primjer jednog pravila (iz te baze) prikazan je u nastavku:

```
son_of
   foreach
      family.son_of($child, $father, $mother)
   asert
      family.child_parent($child, $father, father, son)
      family.child_parent($child, $mother, mother, son))
```

Dakle, za svaku činjenicu koja se pronađe u bazi činjenica, takvu da vrijedi `son_of`, kreiraju se dvije nove činjenice kroz koje se definiše relacija između djeteta i roditelja (`child_parent` odnos), te se kao argumenti ove činjenice navode ime djeteta i roditelja kao prva dva argumenta, informacija o tome da li se radi o majci ili ocu kao treći argument i informacija o tome da li se radi o kćerki ili sinu kao četvrti argument). Također, može se vidjeti da se kao ime baze koristi `family`, odnosno, svaka činjenica počinje ključnom riječju `family`, a to je posljedica toga da su činjenice pohranjene u datoteci koja se zove `family` (uz ekstenziju `rfb`).

Pravila lančanja unazad će detaljnije biti opisana na sljedećoj vježbi.

2.4 Pokretanje ekspertnog sistema

Nakon što su kroz prethodne dijelove vježbe predstavljeni osnovni koncepti koje koristi PyKE okruženje, ispunjeni su uslovi i za pokretanje jednog primjera pisanog u PyKE-u. Kao i u prethodnim dijelovima, i u ovom dijelu će biti korišten primjer ekspertnog sistema za porodične relacije.

Za uspješno pokretanje primjera, potrebno da je da se *import*-uje *driver*. *Driver* za primjer ekspertnog sistema za porodične relacije se nalazi u `family_relations` folderu, pohranjen u datoteci `driver.py` i u ovom dijelu dolazi do povezivanja PyKE-a i njegovih pravila (svojstvenih za deklarativne programske jezika) sa programskim jezikom Python. Prije nego što se *import*-e *driver* za ovaj primjer, upoznati ćemo se sa sadržajom ove datoteke, koji je prikazan u nastavku (dio datoteke):

```
1
2  import contextlib
3  import sys
4  import time
5  from pyke import knowledge_engine, krb_traceback, goal
6  # Compile and load .krb files in same directory that I'm in (recursively).
7  engine = knowledge_engine.engine(__file__)
8
9  fc_goal = goal.compile('family.how_related($person1, $person2, $relationship)')
10
11 def fc_test(person1 = 'bruce'):
12     engine.reset()      # Allows us to run tests multiple times.
13     start_time = time.time()
14     engine.activate('fc_example')  # Runs all applicable forward-chaining rules.
15     fc_end_time = time.time()
16     fc_time = fc_end_time - start_time
17     print("doing proof")
18     with fc_goal.prove(engine, person1=person1) as gen:
19         for vars, plan in gen:
20             print("%s, %s are %s" % \
21                     (person1, vars['person2'], vars['relationship']))
22     prove_time = time.time() - fc_end_time
23     print()
24     print("done")
25     engine.print_stats()
```

```
26      print("fc time %.2f, %.0f asserts/sec" % \
27            (fc_time, engine.get_kb('family').get_stats()[2] / fc_time))
```

Može se vidjeti da se u devetoj liniji koda u priloženom listingu koda (isječak iz `driver.py` datoteke) vrši definisanje cilja, korištenjem `compile` funkcije iz `pyke` biblioteke, odnosno, tu se postavlja pitanje na koje ekspertni sistem treba da odgovori. U ovom konkretnom primjer, ekspertni sistem treba odgovoriti na koji način su povezane osobe koje mu se proslijede kao parametri (parametri `$person1` i `$person2`), s tim da je moguće proslijediti i vrijednost samo jednog parametra, nakon čega se ekspertni sistem odgovoriti kako je ta osoba povezana sa svim ostalim osobama. Inače, bitno je spomenuti da se kao cilj može definisati samo neka od činjenica koja se nalazi u bazi pravila (dakle neka od činjenica koja je dodana - *assert*-ana kao posljedica ispunjenja uslova u *if* dijelu pravila).

Nakon što je definisan cilj, kreirana je funkcija `fc_test` (kao primjer za prosljeđivanje unaprijed), te se u tijelu te funkcije vrši instanciranje baze pravila, u 14. linija koda, korištenjem `activate` funkcije, kojoj se kao parametar prosljeđuje datoteka u kojoj je pohranjena baza pravila (`fc_example` u ovom primjeru). U 18. linija koda se vrši prikupljanje rezultata, korištenjem `prove` funkcije, koja se poziva nad definisanim ciljem, a kojoj se prosljeđuje definisani *engine* kao prvi parametar (a sam *engine* je definisan bazom pravila, pri čemu se više detalja o načinu definisanja *engine*-a može naći na sljedećem linku), dok se kao *ostali* parametri prosljeđuju vrijednosti parametara iz cilja (u ovom slučaju se prosljeđuje vrijednost samo prvog parametera - `person1`, pri čemu je *default*-na vrijednost ovog parametra `bruce`). Pored navedenog, u ovom isječku koda se računa i vrijeme koje je bilo potrebno da eskpertni sistem odgovori na postavljeno pitanje.

Da bi se ovaj isječak koda pokrenuo, potrebno je, kako je već i rečeno, *import*-ovati *driver*, na način koji je prikazan ispod:

```
1 import sys
2 sys.path.append('/Users/user/Downloads/pyke-1.1.1/examples/family_relations')
3 import driver
```

Sada se nad ovim *driver*-om može pozvati funkcija `fc_test`:

```
1 driver.fc_test()
```

Nakon pokretanja, eskpertni sistem će odgovoriti na koji način je Bruce (s obzirom da je to *default*-ni parametar ove funkcije, pri čemu je moguće proslijediti i neko drugo ime kao parametar ove funkcije) povezan sa svim ostalima članovima porodice.

3 Zadaci za rad u laboratoriji

Zadaci u nastavku su predviđeni za samostalni rad studenata.

Zadatak 1 - Ekspertni sistem za porodične relacije

1. Pozvati funkciju `fc_test` iz prethodnog dijela vježbe na način da joj proslijedite neko ime kao parametar (*default*-na vrijednost je Bruce).
2. Izmijeniti `driver.py` datoteku (čiji sadržaj je prikazan u prethodnom dijelu vježbe i koja se nalazi u `family_relations` folderu na način da funkciji `fc_test` mogu proslijediti dva imena za koje ekspertni sistem treba da odgovori kako su povezani.
3. Izmijeniti *driver* ekspertnog sistema za porodične relacije na način da mu se može proslijediti jedno ime i da on vraća sve osobe kojima je proslijeđena osoba brat.
4. Izmijeniti *driver* ekspertnog sistema za porodične relacije na način da mu se može proslijediti jedno ime i da on vraća sve osobe kojima je proslijeđena osoba sestra.
5. Izmijeniti *driver* ekspertnog sistema za porodične relacije na način da mu se može proslijediti jedno ime i da on vraća sve osobe kojima je proslijeđena osoba brat ili sestra (eng. *sibling*).
6. Izmijeniti prethodni zadatak na način da je moguće proslijediti dvije osobe te ukoliko se te dvije osobe brat/sestra sistem treba da to da i ispiše. U suprotnom sistem ne bi trebao ništa ispisati.
7. Izmijeniti *driver* ekspertnog sistema za porodične relacije na način da mu je moguće proslijediti jedno ime i da on vraća sve osobe koje su djed ili baka (eng. *grandparent*) proslijeđenoj osobi.

Zadatak 2 - Ekspertni sistem za vrijeme

Potrebno je napraviti jednostavni ekspertni sistem koristeći PyKE koji će govoriti korisnicima šta treba da ponesu sa sobom, u zavisnosti od vremenskih prilika. Pretpostaviti da je ekspert za vrijeme iznio sljedeća pravila:

1. Ukoliko pada kiša i puše vjetar ponesite kabanicu.
2. Ukoliko pada kiša ali ne puše vjetar ponesite kišobran.
3. Ukoliko ne pada ni kiša niti puše vjetar nemojte ništa nositi sa sobom.

Neka bazu činjenica čine dvije činjenice koje govore da li pada kiša i da li puše vjetar. Naprimjer, baza činjenica (`.kfb` datoteka) bi mogla imati sljedeći sadržaj:

```
pada_kisa(True)
puse_vjetar(True),
```

čime se govori da trenutno i pada kiša i da puše vjetar. Na osnovu pravila koje je dao ekspert potrebno da je osmislite bazu pravila (`.krb` datoteku), koristeći isključivo pravila lančanja unaprijed (`foreach` i `assert` ključne riječi). Potrebno je napisati i jednostavnu Python funkciju (*driver*) koja će testirati rad ekspertnog sistema. Sistem bi trebao da odgovara na pitanje *Šta ponijeti sa sobom?*, odnosno, nakon što se pokrene ekspertni sistem, trebala bi biti ispisana poruka *Trebate ponijeti {Kabanicu, Kišobran, Ništa}*. Testirati ekspertni sistem sa različitim kombinacijama činjenica (kiša pada i vjetar ne puše, kiša ne pada i vjetar puše, kiša ne pada i vjetar ne puše).