Neizrazito, evolucijsko i neuro računarstvo

3. zadatak – Sustav za neizrazito upravljanje

Uvod

Kroz prethodna dva zadatka izradili ste programsko okruženje u kojem možete raditi osnovne operacije nad neizrazitim skupovima. Koristeći taj kod napravite potrebno proširenje koje će Vam omogućiti izradu jednostavnog sustava neizrazitog upravljanja.

U okviru ovog zadatka potrebno je ostvariti sustav za neizrazito upravljanje koji upravlja brodom koji plovi po kanalu. U svrhu lakše vizualizacije i vrednovanja ostvarenog sustava razvijen je simulator ponašanja broda. Cilj ovog djela projekta je napraviti sustav koji može uspješno (bez sudara s obalom) upravljati brodom.

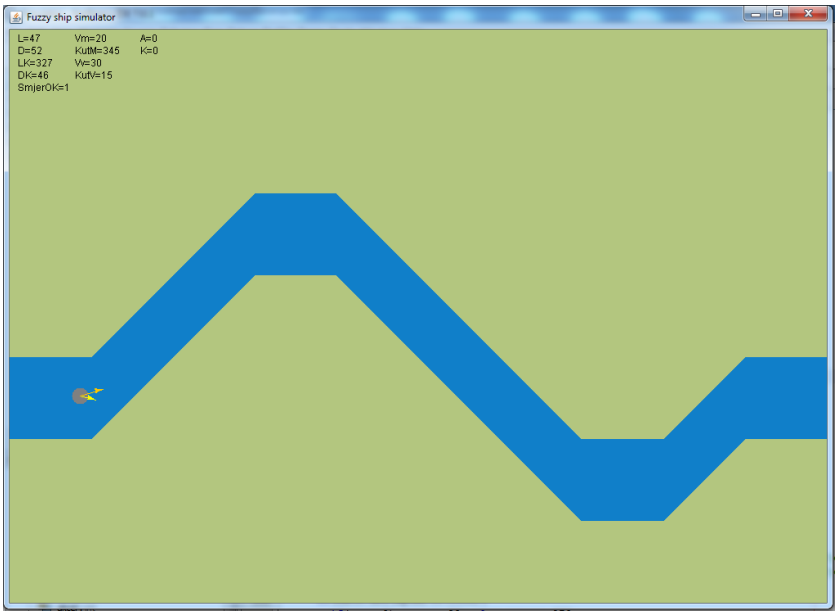
Simulator

Za potrebe simulacije i vizualizacije rada sustava za neizrazito upravljanje koristi se pojednostavljeni model broda koji plovi po kanalu. Brod je predstavljen kružićem polumjera 10 piksela. Dimenzije mape po kojoj brod plovi su 1000 piksela (širina) i 700 piksela (visina).

Na kretanje broda utječu sile koje stvaraju motor broda i vjetar. Vektor brzine broda može se rastaviti na komponentu motora (V_m, na slici 1 označena žuto) i komponentu vjetra (V_v, na slici 1 označena narančasto). One su određene svojim kutem (u odnosu na horizontalu) i iznosom. Iznosi ovih vektora mogu se interpretirati kao brzina u danom smjeru izražena u piksel/s. Ukupan vektor brzine broda dobiva se kao zbroj ove dvije komponente.

Iznos (uvijek između 20 i 50) i smjer vjetra polako se mijenjaju kroz vrijeme na slučajan način (postupno dakle bez naglih promjena). Komponentu V_m brzine kontrolira sustav upravljanja brodom preko dva parametra:

- A – akceleracija
  - ovaj parametar modelira pritisak na papučicu gasa ili papučicu kočnice
  - pozitivna akceleracija iznosa X uzrokovati će povećanje iznosa V_m za X piksela/s svake 1s



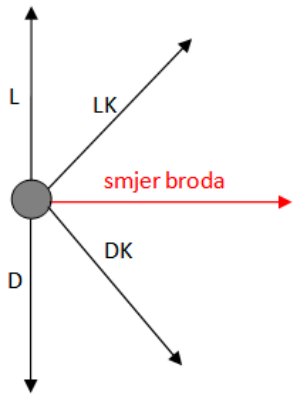
Slika. 1: Simulator broda koji plovi po kanalu.

  - smanjenje iznosa V_m moguće je postići na isti način negativnim iznosom akceleracije
  - V_m nije moguće smanjiti ispod 0

- K – kut kormila
  - ovaj parametar modelira zakretanje broda pomoću kormila
  - kut kormila od $X°$ uzrokuje zakretanje V_m komponente brzine za kut X svakih 1s.
  - pozitivni kutevi uzrokuju skretanje u lijevo (u smjeru suprotnom od kazaljke na satu)
  - negativni kutevi uzrokuju skretanje u desno
  - zbog modeliranja ograničenih mogućnosti zakretanja broda dopušteni raspon vrijednosti je $[-90, 90]$
  - ako se zada vrijednost izvan dopuštenog intervala ona se zaokružuje na najbližu dopuštenu vrijednost

Odluku o vrijednostima upravljačkih parametara sustav za upravljanje donosi na temelju sljedećih informacija:

- L – udaljenost broda od obale prema lijevo
- D – udaljenost broda od obale prema desno



Slika 2: Smjerovi u kojima se mjere udaljenosti L,D,LK i DK uz dani smjer broda.

- LK – udaljenost broda od obale naprijed lijevo pod kutem od 45°
- DK – udaljenost broda od obale naprijed desno pod kutem od 45°
- V – iznos brzine broda (iznos vektora V_m)
- S – podatak o tome kreće li se brod u otprilike pravom smjeru (1 ako da, 0 inače)

Vrijednosti L,D,LK i DK izračunavaju se u odnosu na trenutni smjer u kojem je brod okrenut tj. smjer broda na slici 2 je smjer vektora V_m. Vrijednosti ovih veličina kreću se u intervalu $[0, 1300]$. Ako udaljenost broda od obale padne ispod 10 (polumjer broda) brod se sudario s obalom i simulacija prestaje.

Za brod se smatra da ide u otprilike pravom smjeru ako njegov stvarni smjer ne odudara za više od 90° od predefiniranog ispravnog smjera.

Format rješenja

Simulator će komunicirati s vašim programom pomoću standardnog ulaza i izlaza. Sustav za neizrazito upravljanje može se ostvariti u proizvoljnom programskom jeziku tako da obavlja pseudokod prikazan u nastavku.

```
dok(true)
   lnIn = pročitaj liniju sa standardnog ulaza
   ako je lnIn = "KRAJ", prekini
   lnOut = izracunajIzlaze (lnIn)
   zapiši liniju lnOut na standardni izlaz
   napravi flush standardnog izlaza
```

Pri tome će linija *lnIn* koju ćete dobiti sadržavati 6 cijelih brojeva odvojenih razmacima i to redom L, D, LK, DK, V i S. Linija *lnOut* koju zapisujete mora sadržavati dva cijela broja odvojena razmakom i to redom A i K. Važno je napraviti **flush** standardnog izlaza nakon što ste na njega zapisali podatke jer inače komunikacija **neće raditi**. Na kraju simulacije linija lnIn će sadržavati riječ "KRAJ".

Kada ste implementirali svoj sustav neizrazitog upravljanja, potrebno je inicijalno podesiti simulator. U direktoriju simulatora nalazi se datoteka imena `config.txt`. U njen prvi red potrebno je upisati naredbu kojom se pokreće vaš program (npr. `"myFuzzyControlSys.exe"` ili `"python myFuzzyConSys.py"`). U njenom drugom redu potrebno je navesti broj staze koja će se koristiti (1, 2 ili 3). Postoje tri staze različitih težina, možete jednostavno birati između njih upisujući odgovarajući broj u drugi red datoteke `config.txt`. Primjerice ako se vaš program pokreće naredbom `"C:\fuzzy\fuzzCtrl.exe"` i želite koristiti drugu od ukupno tri definirane staze vaša `config.txt` datoteka će izgledati ovako:

```
C:\fuzzy\fuzzCtrl.exe
2
```

Sada možete pokrenuti simulator i vidjeti kako vaš sustav upravlja brodom (simulator će interno pozivati vaš program koristeći naredbu koju ste upisali u datoteku `config.txt`, ako pokretanje ne uspije na konzoli ćete dobiti ispis s porukom o grešci). To možete napraviti duplim klikom na datoteku simulator.jar ili iz konzole koristeći sljedeću naredbu: `java -jar simulator.jar`.

Ako slučajno nemate instaliran Java Runtime Environment potreban za pokretanje simulatora on se može preuzeti na `http://www.oracle.com/technetwork/java/javase/downloads/index.html`.

Zadatak

Potrebno je napraviti sustav neizrazitog upravljanja koji će na temelju ulaznih vrijednosti L, D, LK, DK, V i S izračunati odgovarajuće vrijednosti za akceleraciju i kut kormila. Modelirajte sustav interno kao dva MISO sustava (jedan koji upravlja akceleracijom, drugi koji upravlja kutem). Prema van, taj sustav možete zatvoriti u jedan sustav koji ima više ulaza i dva izlaza. Razmislite ima li smisla da za skup pravila odaberete skup svih pravila možete dobiti direktno temeljem osnovnih jezičnih izraza svake od jezičnih varijabli, kao što je primjer u knjizi gdje imamo dvije jezične varijable? Koliko bi takvih pravila bilo? Je li Vaš sustav prikladan za primjenu tehnike ubrzanja rada upravljačkog sustava koja je opisana u podpoglavlju 5.5.1 knjige, i zašto?

Primarni cilj upravljačkog sustava treba biti uspješno provođenje broda od početka do kraja kanala bez sudara s obalom. Sekundarni cilj je probati to ostvariti u što kraćem vremenu.

Pravila koja ćete koristiti trebaju biti oblika (ovo je samo primjer):

```
AKO  L je "Kritično blizu ILI Blizu" I LK je "..." I ...
```

```
ONDA K je "Oštro desno"
```

Pravila ispituju kombinaciju ulaznih jezičnih varijabli (ne nužno sve u svakom pravilu). Vrijednost s kojom se uspoređuje pojedina jezična varijabla može biti jednostavan izraz ili pak složen izraz ostvaren uporabom odgovarajućih veznika (u prethodnom primjeru, "Kritično blizu" i "Blizu" su jednostavni izrazi, "Kritično blizu ILI Blizu" je složen izraz. U vašem rješenju za svaki od izlaza koji generirate trebate imati niz pravila.

Rješenje trebate oblikovati na sljedeći način.

1. Izgradite model jednog pravila. Jedno pravilo u konačnici treba biti modelirano kao jedan neizraziti skup (koji s obzirom na trenutne vrijednosti ulaznih varijabli izračunava mjeru pripadnosti svakog od elemenata univerzalnog skupa nad kojim je definiran).

2. Izgradite bazu pravila. Baza pravila je kolekcija pravila modeliranih u prethodnom koraku. U Vašem rješenju imat ćete dvije baze pravila: jednu koja grupira pravila koja određuju akceleraciju te drugu koja upravlja kormilom.

3. Napravite konačni neizraziti zaključak: to je novi neizraziti skup koji predstavlja uniju zaključaka svakog od pravila.

4. Definirajte sučelje koje opisuje postupak dekodiranja neizrazitosti. Napišite implementaciju tog sučelja koje obavlja dekodiranje u skladu s metodom COA. Prima neizraziti skup a vraća cijeli broj.

5. Napišite novo sučelje koje modelira jedan sustav neizrazitog upravljanja. Sustav nudi API preko kojeg je moguće zadati vrijednosti izmjerene senzorima (trenutne udaljenosti od obala i slično) te potom zatražiti vrijednost izlazne varijable. Potom implementirajte dva takva sustava: jedan koji ima bazu pravila za određivanje akceleracije te drugi koji ima bazu pravila za kormilo.

Vanjska petlja vašeg programa trebala bi izgledati ovako (pseudokod):

```
main() {
  // Biramo način dekodiranja neizrazitosti:
  Defuzzifier def = new COADefuzzifier();
  // Stvaranje oba sustava:
  // Grade se baze pravila i sve se inicijalizira
  FuzzySystem fsAkcel = new AkcelFuzzySystemMin(def);
  FuzzySystem fsKormilo = new KormiloFuzzySystemMin(def);
  // Glavna petlja:
  while(true) {
    Pročitaj L, D, LK, DK, V, S sa standardnog ulaza
    // Zadaj ulaze, generiraj neizraziti izlaz, dekodiraj i vrati ga:
    A = fsAkcel.zaključi(L,D,LK,DK,V,S);
```

```
    K = fsKormilo.zaključi(L,D,LK,DK,V,S);
    Zapiši A, K na standardni izlaz
    Flush standardni izlaz
  }
}
```

Hoćete li metodi "zaključi" predavati primljene vrijednosti baš na prikazani način ili nekako drugačije, to je implementacijski detalj. Međutim, morate poštivati opisanu ideju: sustav se gradi jednom, na početku programa. Inicijaliziraju se pravila. Potom se u petlji sustavima samo daju ulazne vrijednosti i sustav radi zaključivanje, uniju zaključaka te dekodiranje neizrazitosti i na kraju vraća rezultat.

Ovisno o Vašoj implementaciji, generiranje neizrazitih skupova koji predstavljaju zaključke pojedinih pravila te njihovu uniju možete računati i "lijevo" (na zahtjev, kada to zatreba proceduri za dekodiranje neizrazitosti).

Prilikom izrade rješenja, nije potrebno pravila pisati u tekstualnom formatu pa pisati parser za njihovu obradu. Pravila i baza mogu biti hardkodirani u vašem programu.

Implementirajte dva mehanizma zaključivanja koja bi korisnik trebao moći odabrati bez velikih promjena po kodu: stroj za zaključivanje koji se temelji na minimumu te stroj za zaključivanje koji se temelji na produktu. Pri tome slobodno radite s restrikcijom koja sustav za kodiranje neizrazitosti implementira isključivo preko jednoelementnih skupova (engl. *singletons*) čime ćete moći raditi efikasnu obradu pojedinačnih pravila. Kombiniranje zaključaka provest ćete kako to propisuje odabrani stroj za zaključivanje.

Uz glavni program pripremite još i sljedeće pomoćne programe (odlučite se za jednu bazu: ili onu za akceleraciju, ili onu za kormilo):

1. program koji korisnika pita da odabere jedno od pravila iz Vaše baze te da zada vrijednosti L, D, LK, DK, V, S; program potom na zaslon ispisuje neizraziti skup koji je zaključak tog pravila; dodatno, ispisuje i dekodiranu vrijednost tog zaključka;

2. program koji korisnika pita da zada vrijednosti L, D, LK, DK, V, S; program računa uniju svih zaključaka Vaše baze i taj neizraziti skup ispisuje; također, ispisuje njegovu dekodiranu vrijednosti.

Ovi programi pomoći će Vam da i sami vidite što pojedino pravilo odnosno čitava baza generira za određene ulaze.

U slučaju bilo kakvih pitanja i/ili nejasnoća, slobodno nas potražite za konzultacije.