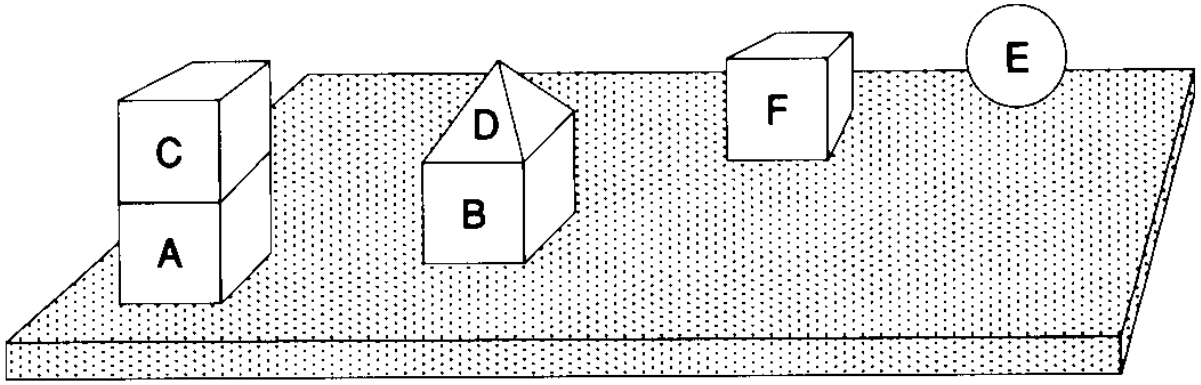
SISTEMI ZASNOVANI NA PRAVILIMA

Person: PICK UP A BIG RED BLOCK.
Computer: OK. (does it)
Person: GRASP THE PYRAMID.
Computer: I DON'T UNDERSTAND WHICH PYRAMID YOU MEAN.
Person: FIND A BLOCK WHICH IS TALLER THAN THE ONE YOU ARE HOLDING AND PUT IT INTO THE BOX.
Computer: BY "IT", I ASSUME YOU MEAN THE BLOCK WHICH IS TALLER THAN THE ONE I AM HOLDING.
Computer: OK. (does it)
Person: WHAT DOES THE BOX CONTAIN?
Computer: THE BLUE PYRAMID AND THE BLUE BLOCK.
Person: WHAT IS THE PYRAMID SUPPORTED BY?
Computer: THE BOX.
Person: HOW MANY BLOCKS ARE NOT IN THE BOX?
Computer: FOUR OF THEM.
Person: IS AT LEAST ONE OF THEM NARROWER THAN THE ONE WHICH I TOLD YOU TO PICK UP?
Computer: YES, THE RED CUBE.
Person: IS IT SUPPORTED?
Computer: YES, BY THE TABLE.
Person: CAN THE TABLE PICK UP BLOCKS?
Computer: NO.
Person: CAN A PYRAMID BE SUPPORTED BY A BLOCK?
Computer: YES.
Person: CAN A PYRAMID SUPPORT A PYRAMID?
Computer: I DON'T KNOW.
Person: STACK UP TWO PYRAMIDS.
Computer: I CAN'T.
Person: WILL YOU PLEASE STACK UP BOTH OF THE RED BLOCKS AND EITHER A GREEN CUBE OR A PYRAMID?
Computer: OK. (does it)



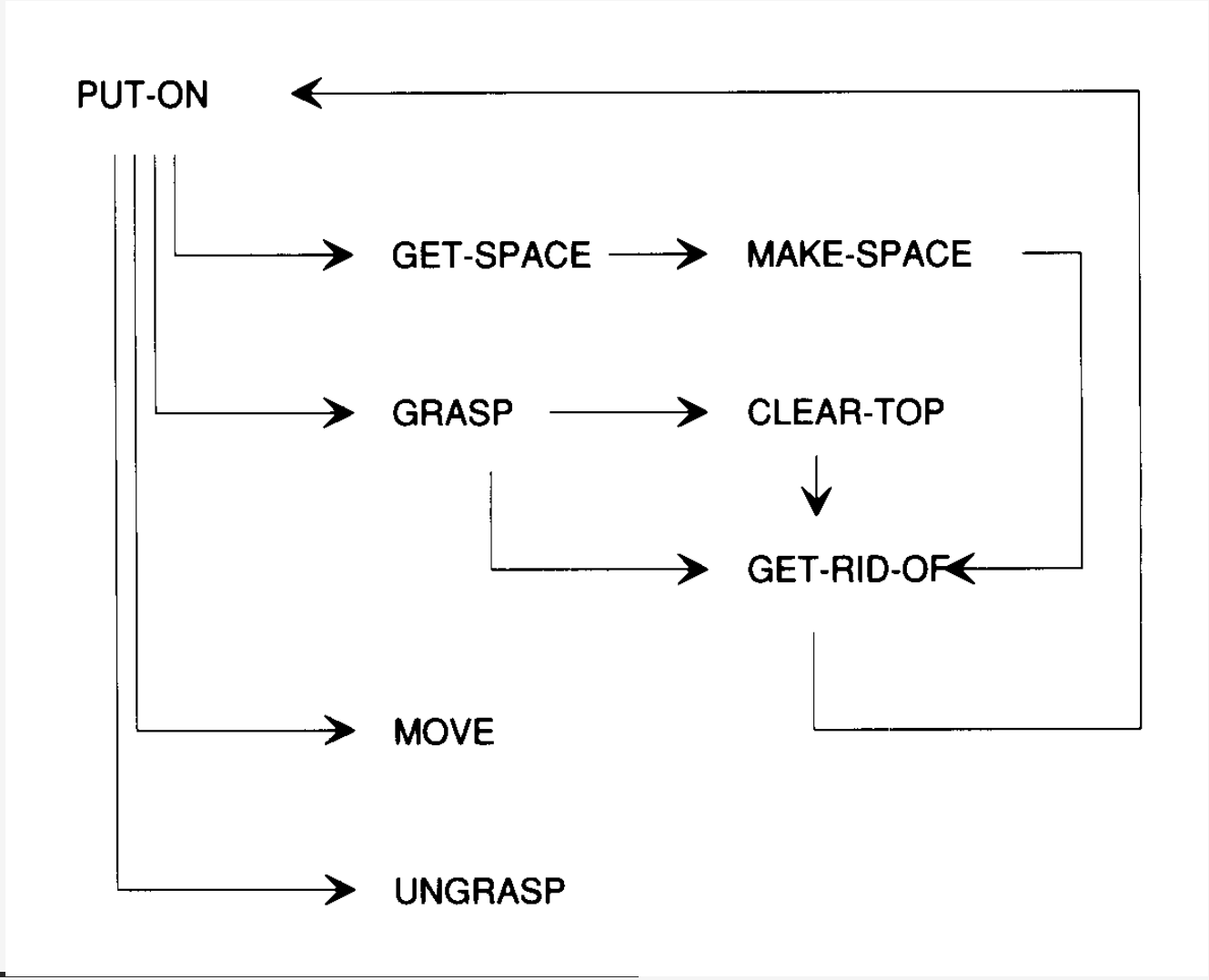
Problem: stavi kocku A na kocku B

SVET BLOKOVA

SHRDLU was an early natural language understanding computer program, developed by Terry Winograd at MIT in 1968–1970. In it, the user carries on a conversation with the computer, moving objects, naming collections and querying the state of a simplified "blocks world", essentially a virtual box filled with different blocks.

Terry Winograd

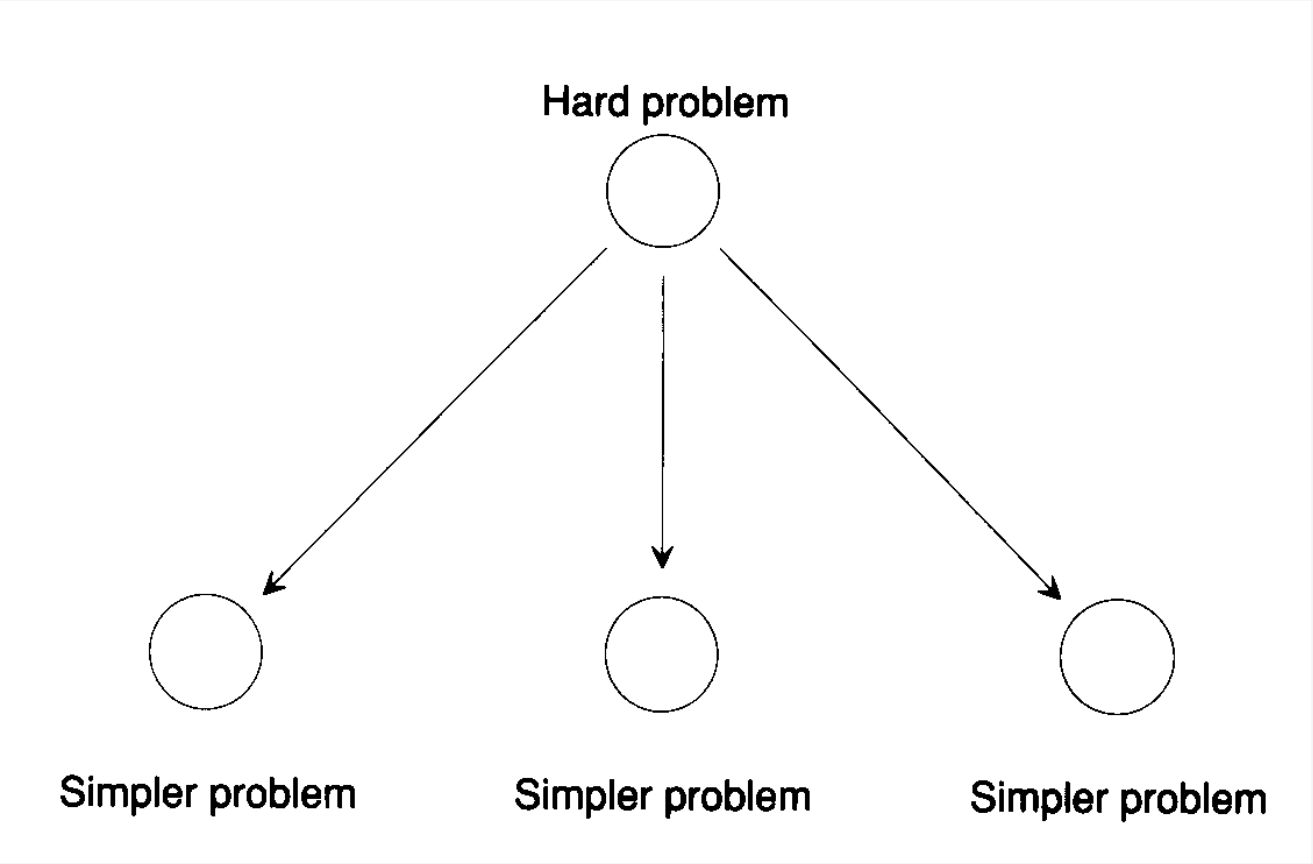
SVET BLOKOVA



Procedure koje se koriste u svetu blokova

Svet blokova

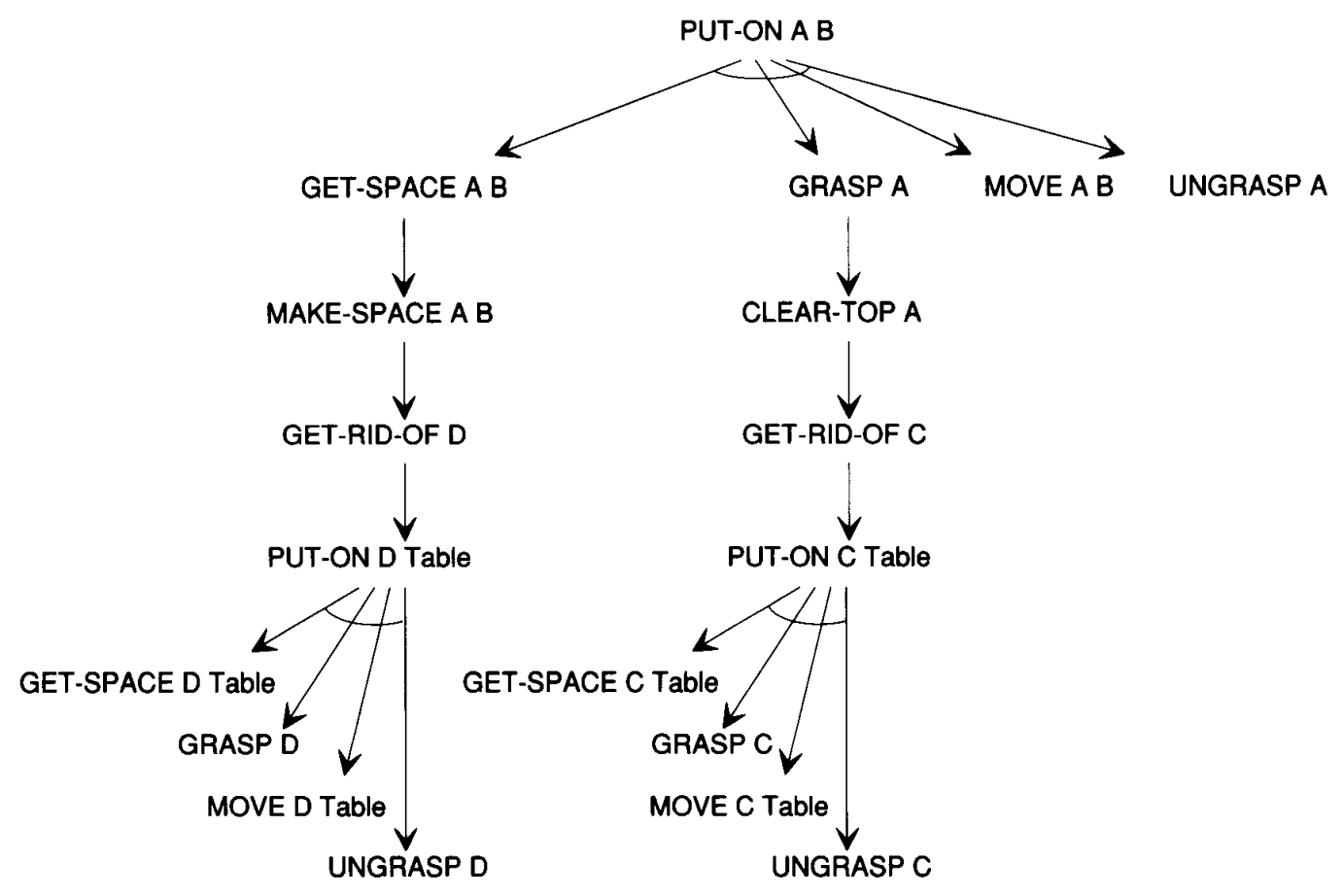
- Dok rešava problem premeštanja blokova, agent kreće od osnovnog (krajnjeg cilja), i razlaže ga na podciljeve.
- Na taj način gradi STABLO CILJEVA - GOAL TREE



Goal tree

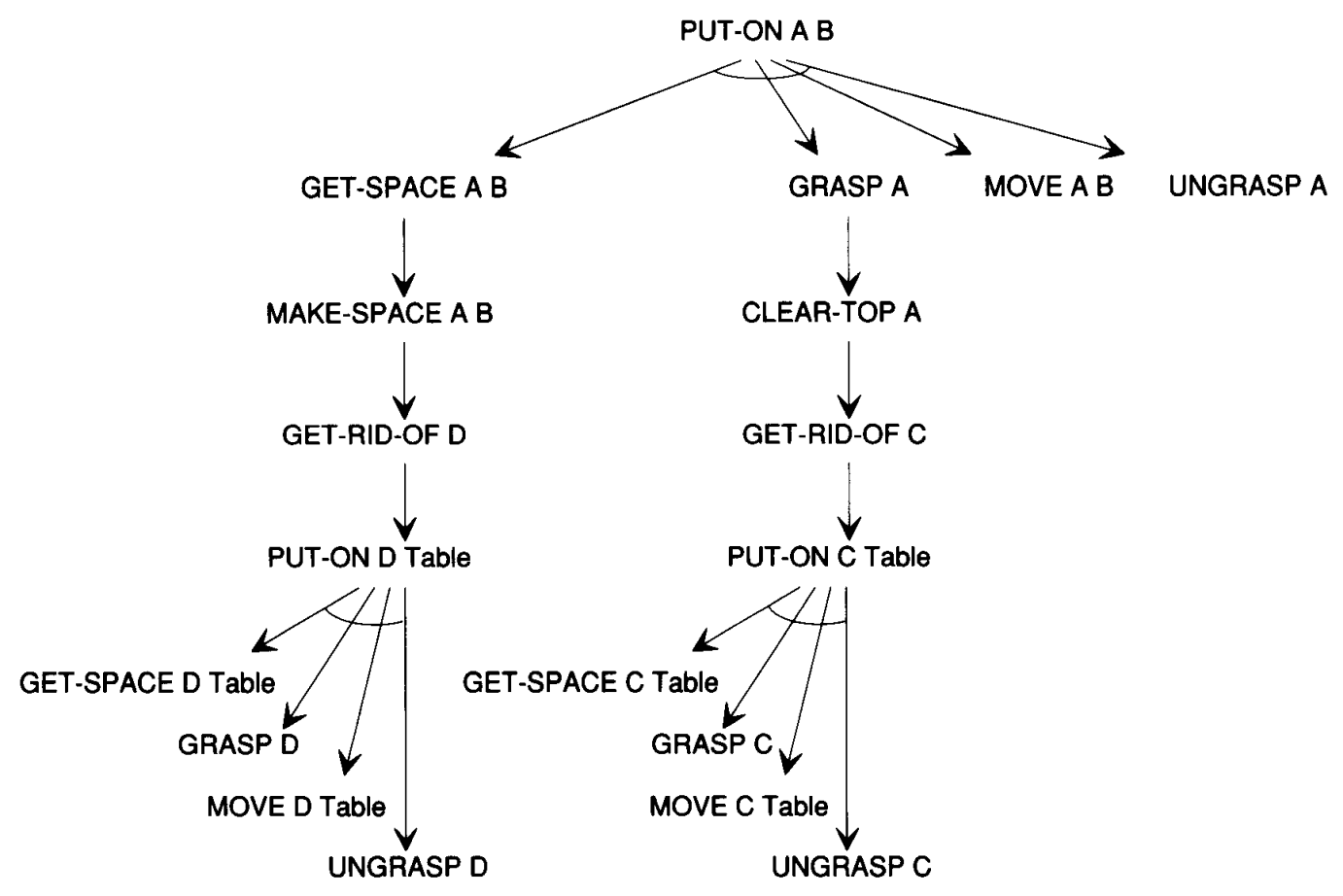
- Glavni cilj se nalazi u korenu stabla.
- Grane vode ka akcijama – podciljevima koje je potrebno ostvariti, da bi bio postignut nadcilj.
- Ciljevi koji su zadovoljeni bez potrebe da se realizuju podciljevi nalaze se u krajnjim čvorovima stabla – listovima.
- Stabla ciljeva sadrže ciljeve koji se ostvaruju:
  - ako se ostvare svi njihovi podciljevi (AND)
  - ako se ostvari bar jedan njihov podcilj (OR)
- AND-OR stabla

AND-OR stablo za svet blokova



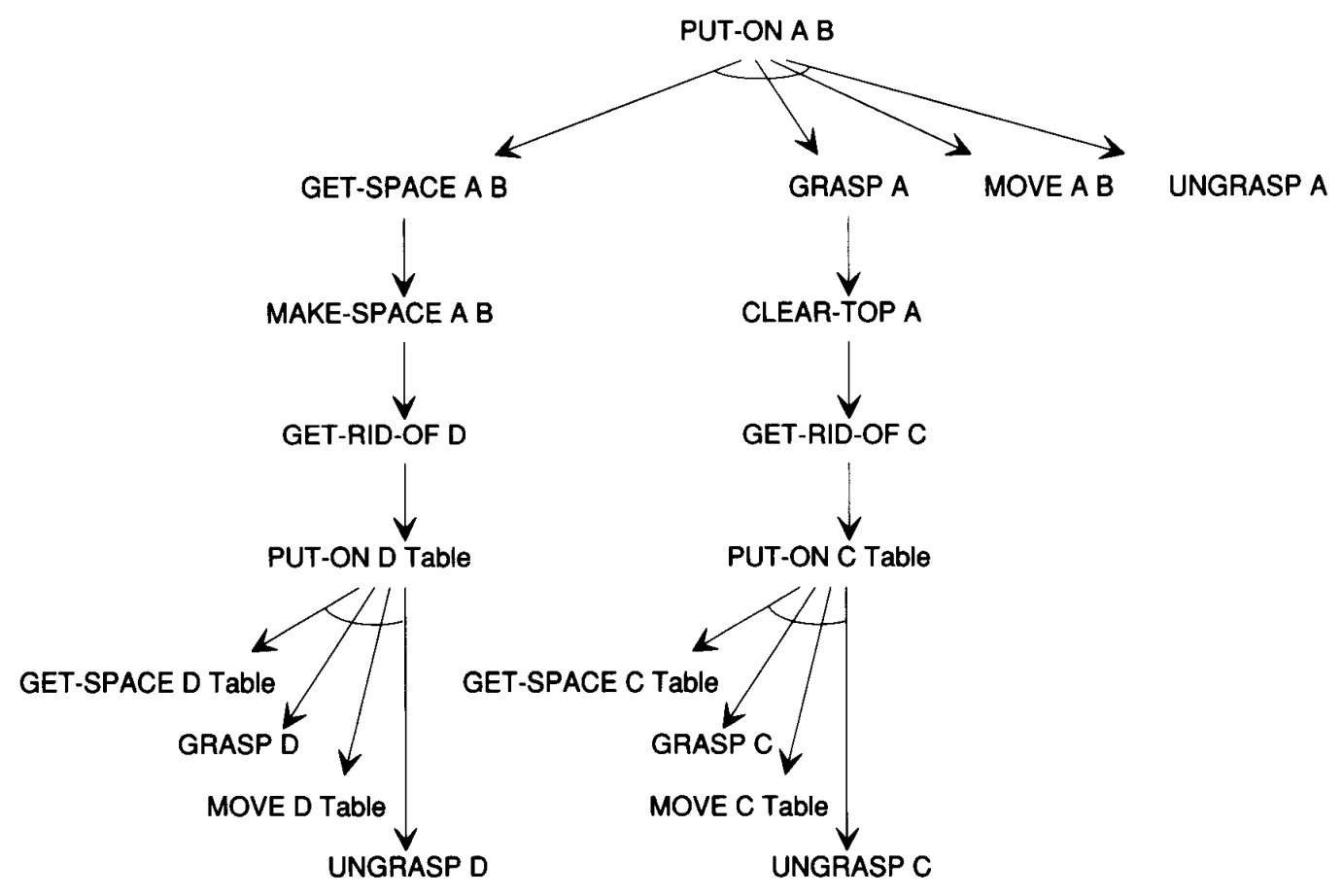
AND-OR stablo za svet blokova

Da bismo odgovorili na „ZAŠTO“



AND-OR stablo za svet blokova

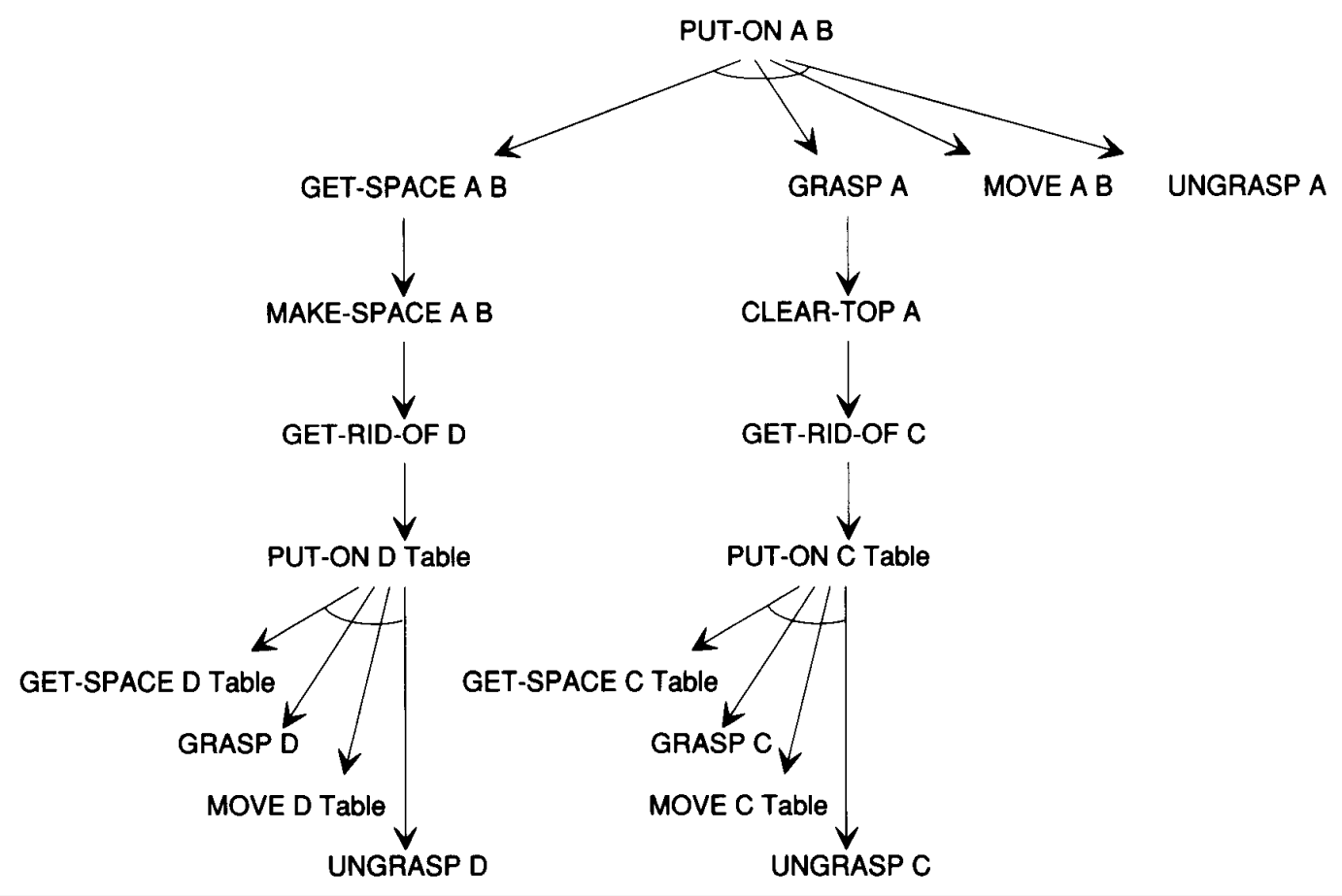
Da bismo odgovorili na „ZAŠTO“ pitanja krećemo se uz stablo.



AND-OR stablo za svet blokova

Da bismo odgovorili na „ZAŠTO“ pitanja krećemo se uz stablo.

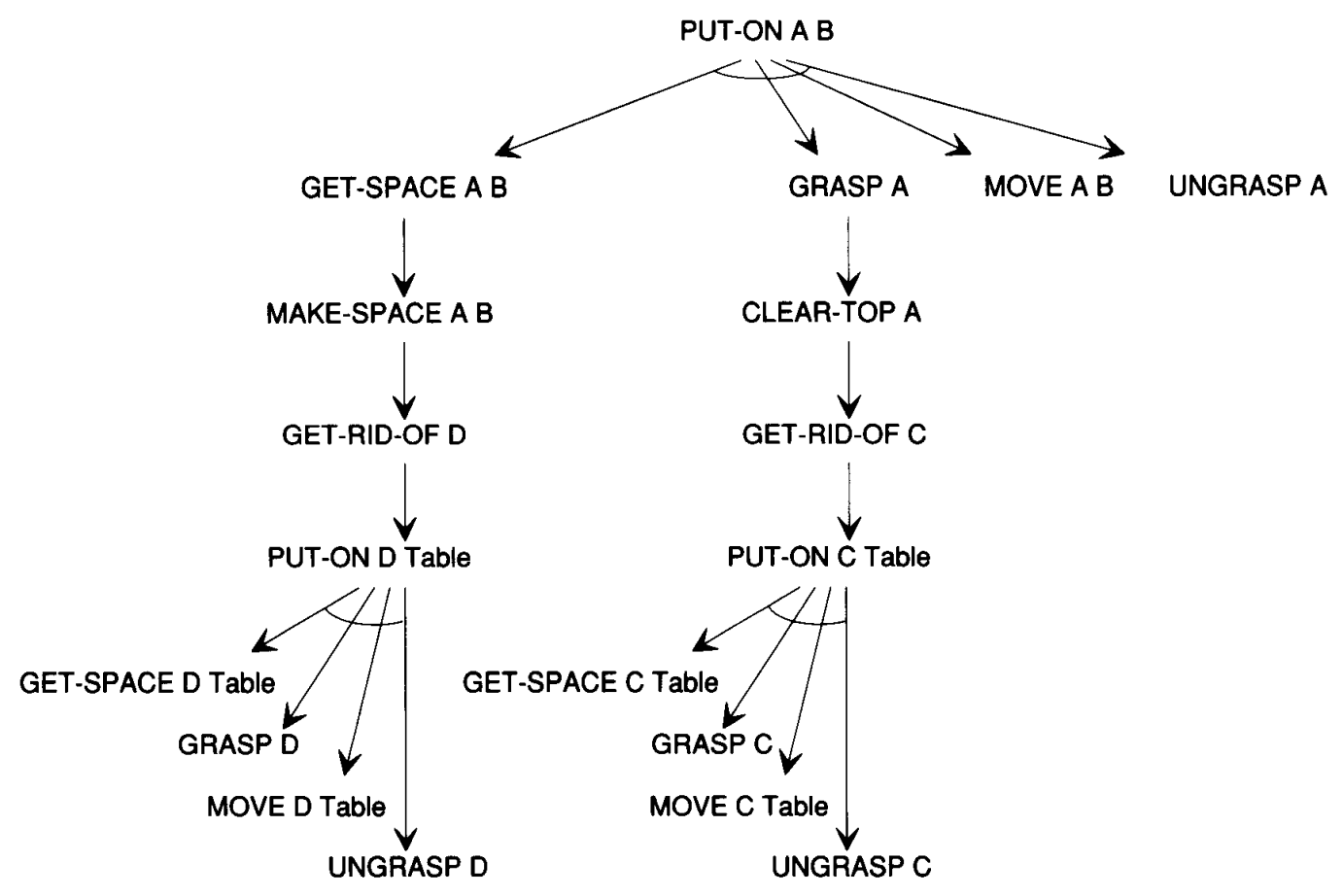
Da bismo odgovorili na „KAKO“



AND-OR stablo za svet blokova

Da bismo odgovorili na „ZAŠTO“ pitanja krećemo se uz stablo.

Da bismo odgovorili na „KAKO“ pitanja krećemo se niz stablo.



Šta je ES?

„Ekspertni sistem je kompjuterski program koji reprezentuje znanje i zaključuje na osnovu znanja iz nekog uskog domena u cilju rešavanja problema ili davanja saveta."

„Inteligentni kompjuterski program koji koristi znanje i procedure zaključivanja za rešavanje problema koji su dovoljno teški da zahtevaju značajnu ljudsku ekspertizu za svoje rešavanje."

„Ekspertni sistem je kompjuterski program koji simulira proces ljudskog rezonovanja i primenom ekspertskog znanja rešava probleme."

Šta je ekspert?

- Ekspert – osoba koja poseduje znanje ili veštine koje su većini ljudi nepoznate ili nedostupne.

- Prvi ES su koristili isključivo ekspertsko znanje, dok se danas koristi i znanje iz knjiga, časopisa i slično, pa se termini *ekspertni sistemi* i *sistemi bazirani na znanju* koriste kao sinonimi.

Karakteristike ES

- Simulira ljudsko razmišljanje o problemu
- Rezonuje na osnovu reprezentacije ljudskog znanja
- Probleme rešava korišćenjem heuristika ili aproksimacija, koje za razliku od algoritamskih metoda ne garantuju uspeh
- Obično je ograničen na neki specifičan domen

Ekspertni sistem — Sistem zasnovan na znanju — Sistem produkcionih pravila

- N. Wirth u knjizi o Pascal-u:

Algorithms + Data Structures = Programs

- J.C. Giarratano & G.D. Riley o ES:

Knowledge + Inference = Expert Systems

Predstavljanje znanja

Najčešće vrši korišćenjem

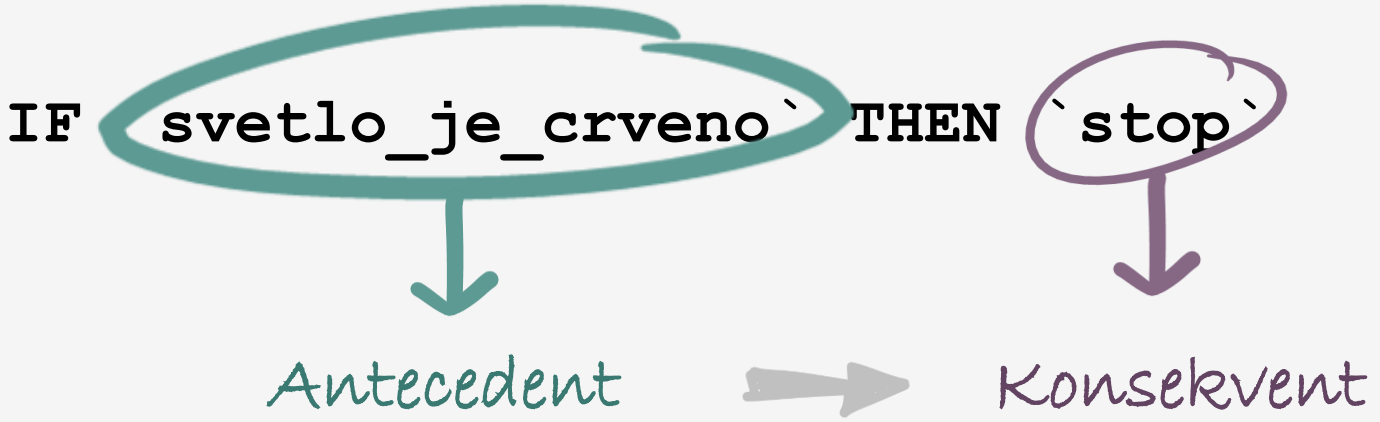
- Pravila (*rules*)

```
IF `svetlo_je_crveno` THEN `stop`
```

- Tvrdnji (*assertions*)

```
`svetlo_je_crveno`
```

If if_1 if_2 ⋮ then $then_1$ $then_2$ ⋮



Sistem produkcionih pravila

```
(defrule p1
(kisa)
=>
(assert (staza mokra)))

(defrule p2
(prskalica)
=>
(assert (staza mokra)))

(defrule p3
(staza mokra)
=>
(assert(staza klizava)))
```

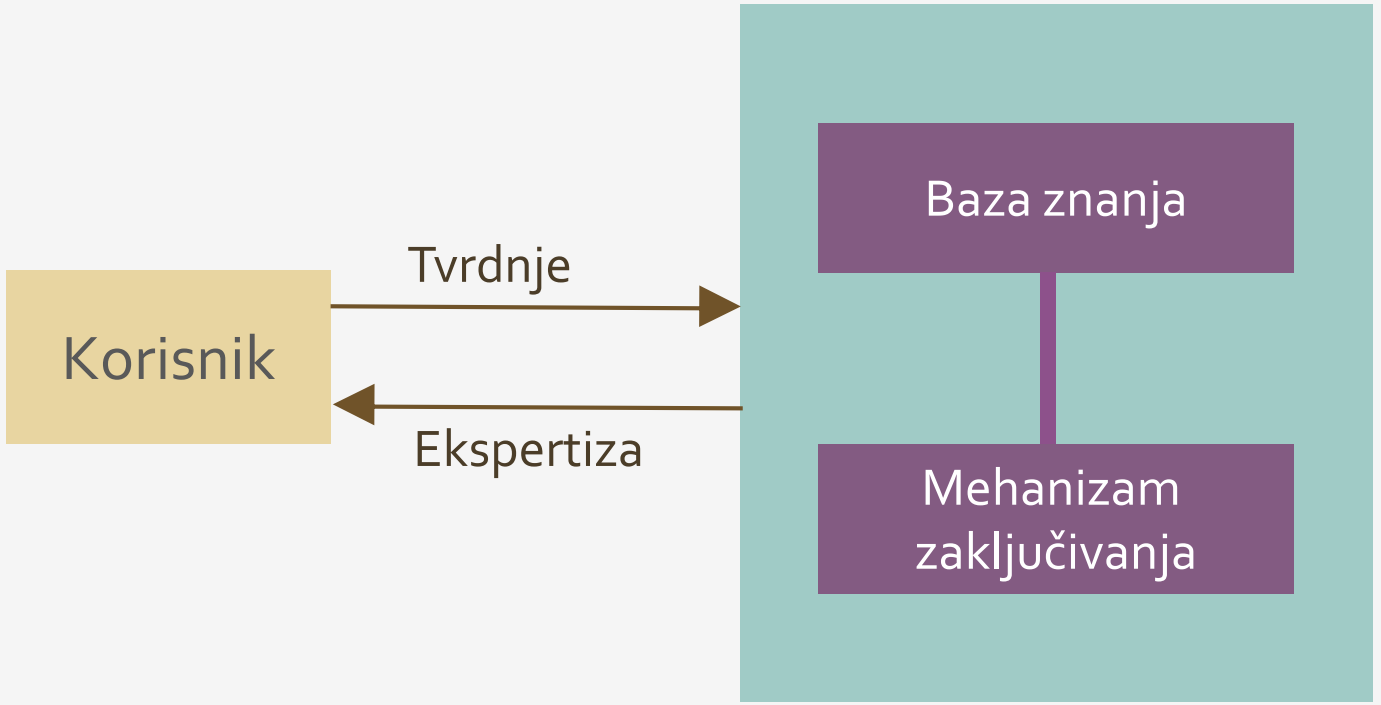
- Sistem radi sa skupom pravila i skupom tvrdnji.
- Tvrdnja je izjava (iskaz) da je nešto tačno.
- Pravila traže određene uzorke (*patterns*) u tvrdnjama - *antecedenti* pravila
- I obično stvaraju nove tvrdnje *konsekvente*.
- Pravila mogu i da brišu postojeće tvrdnje.

- Tvrdnje ili činjenice?

Facts and assertions are subtly different: A *fact* is something known to be true; an assertion is a statement that something is a fact. Thus, assertions can be false, but facts cannot be false.

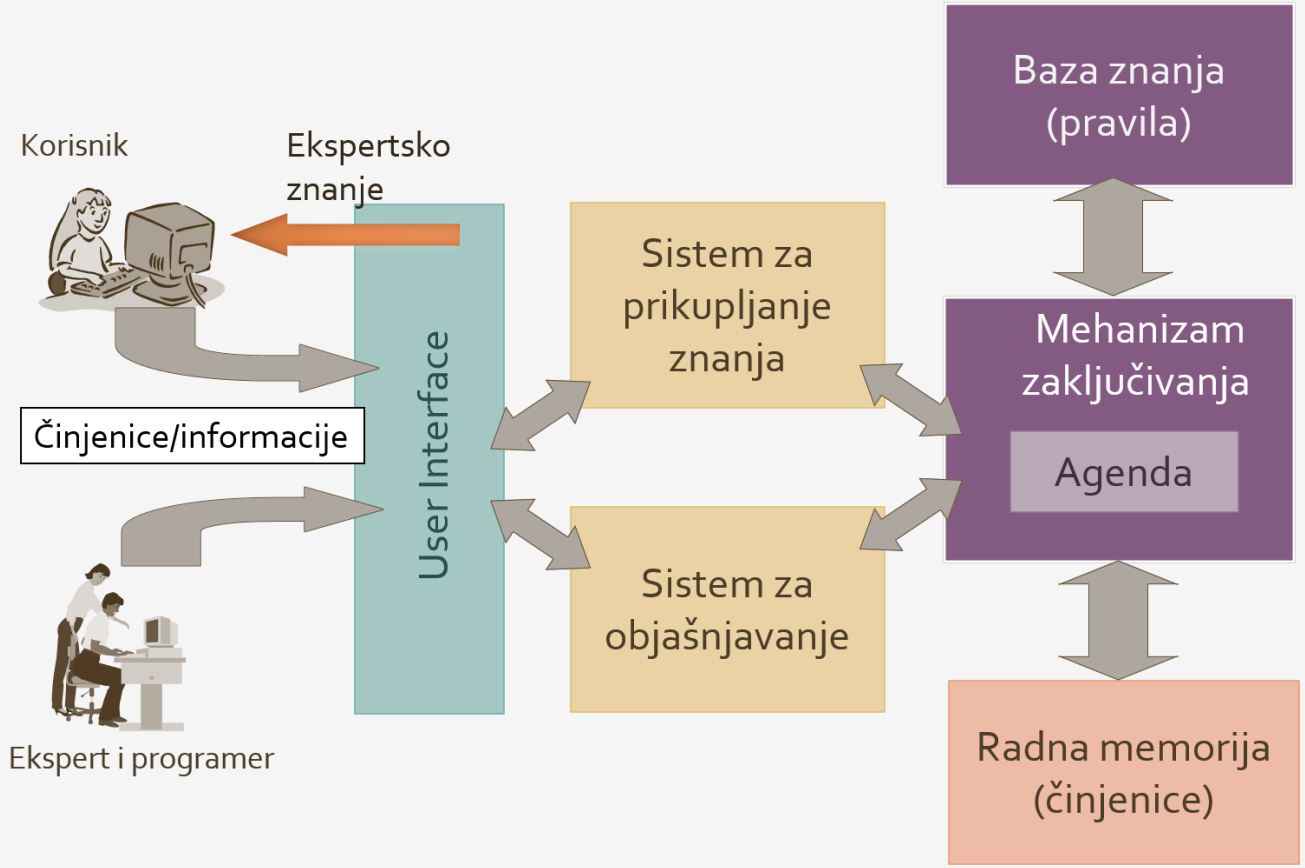
Osnovna struktura ekspertnog sistema

- Korisnik opskrbljuje ES tvrdnjama, a za uzvrat dobija ekspertski savet.
- Interno, ES se sastoji od dve glavne komponente: baze znanja i mehanizma zaključivanja (*inference engine*).
- U bazi je znanje na osnovu koga mehanizam zaključuje. Ovi zaključci su odgovor ES-a na upit korisnika.
- ES rezonuje na osnovu znanja iz određenog domena – on nije *general problem solver*



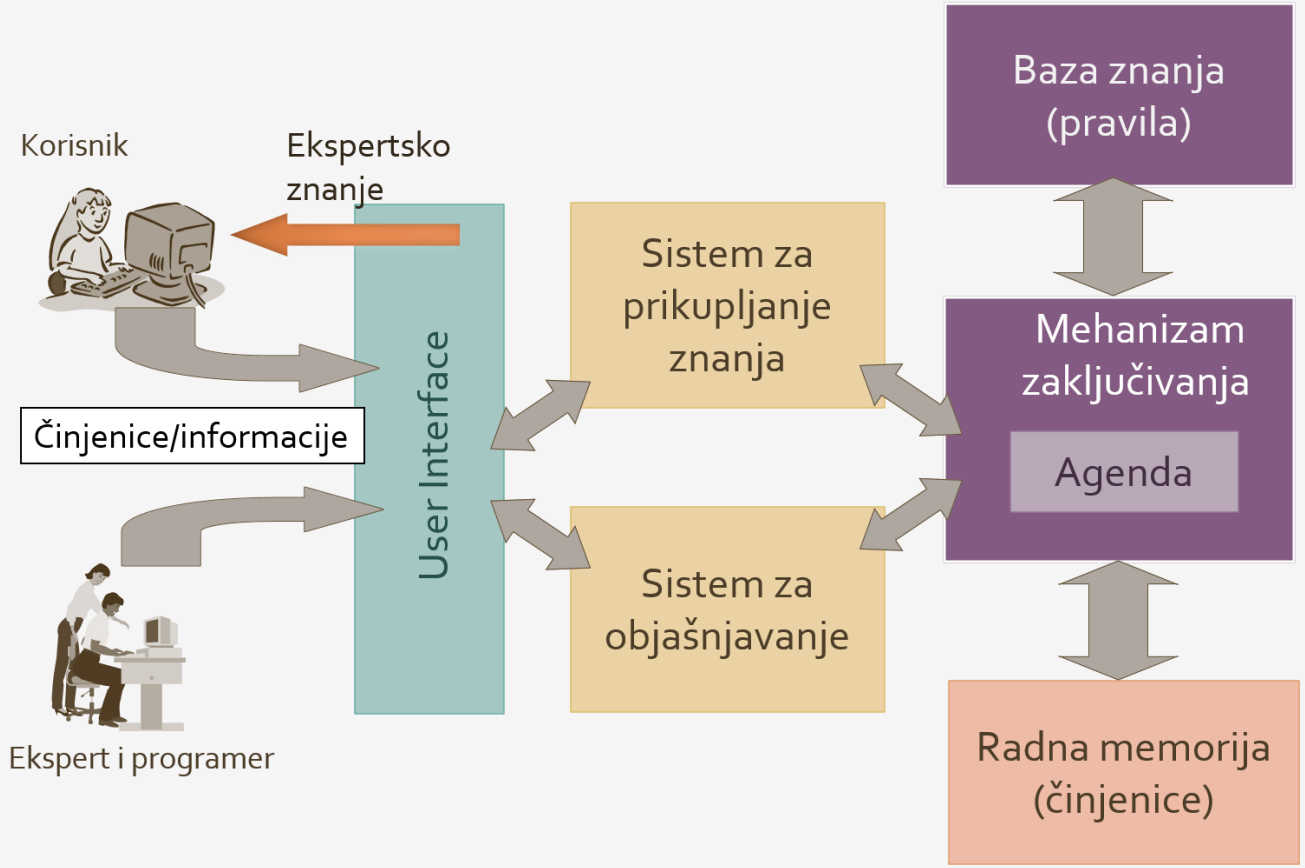
Glavne komponente ES-a:

1. **user interface**
   - interakcija sa korisnicima
   - razvoj i održavanje baze znanja
2. **sistem za prikupljanje znanja**
   - omogućava korisniku da automatski unosi znanje u sistem
3. **baza znanja (*knowledge base*)**
   - sadrži znanje kodirano pravilima



Glavne komponente ES-a:

4. **radna memorija**
   - sadrži trenutno aktuelne činjenice
5. **mehanizam zaključivanja (*inference engine*)**
   - vrši zaključivanje izvršavajući pravilo sa najvišim prioritetom u agendi
   - agenda je lista pravila zadovoljenih činjenicama koje su u radnoj memoriji
6. **sistem za objašnjavanje**
   - objašnjava korisniku način rezonovanja ESa



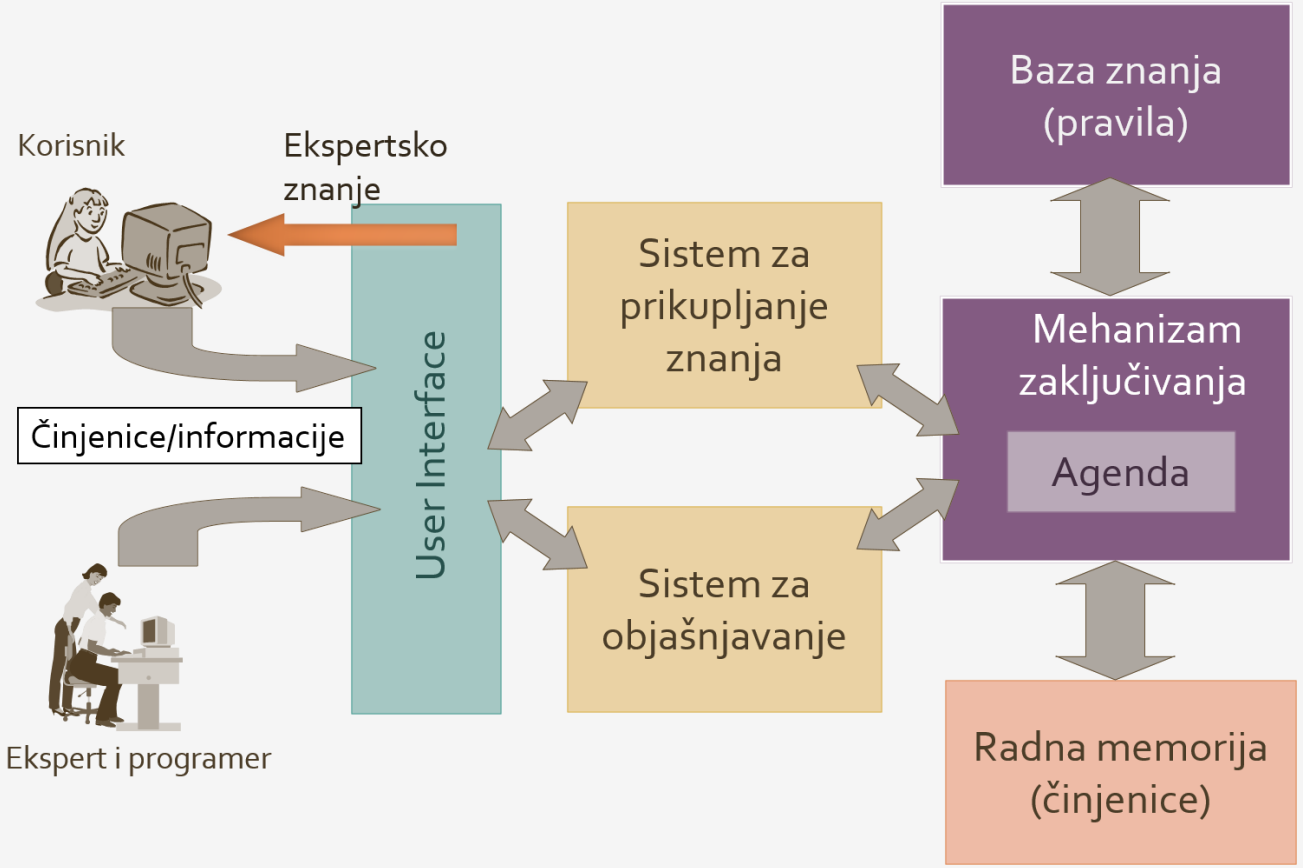
Osnovni koncepti ES

- Prikupljanje znanja

– Transfer i transformacija ekspertskog znanja, potrebnog za rešavanje nekog problema, od izvora znanja do programa.

– Inženjer znanja (Knowledge engineer)

- Predstavljanje znanje
- Zaključivanje
- Objašnjavanje



Ciklus mehanizma za zaključivanje u CLIPS-u

- inference engine pronalazi pravila čiji su antecedenti zadovoljeni
  - ✓ leva strana mora odgovarati činjenici (*match a fact*) u radnoj memoriji
- pravila koja su zadovoljena smeštaju se u agendu i nazivaju se aktivirana pravila
  - ✓ pravila su u agendi poređana po prioritetu
- Razrešavanje konflikta
  - ✓ bira pravilo iz agende sa najvišim prioritetom

Ciklus mehanizma za zaključivanje u CLIPS-u

- Izvršenje pravila (firing)
  - ✓ sprovodi akcije određene konsekventom odabranog pravila
  - ✓ uklanja pravilo iz agende
- Update-uje agendu pravila
  - ✓ pravila čiji su antecedenti zadovoljeni dodaje u agendu
  - ✓ iz agende uklanja pravila koja nisu zadovoljena

Ciklus se završava kada u agendi nema više pravila ili kada se naiđe na eksplicitnu komandu za zaustavljanje programa

Promenljive u pravilima

- Pravila mogu sadržati promenljive.
- Promenljiva dobija vrednost kroz proces uparivanja tvrdnji sa uslovima pravila
- Konsekvent može sadržati promenljive koje su se kroz uslov pravila vezale za konkretnu vrednost
- Promenljive omogućavaju uparivanje uslova pravila sa više činjenica

```
IF( AND( 'parent (?x) (?y)',
         'parent (?x) (?z)' ),
THEN( 'sibling (?y) (?z)' ))
```

Tvrdnje:

```
'parent marge bart'
'parent marge lisa'
```

Zaključak

```
'sibling bart lisa'
```

Ali i:

```
'sibling bart bart'
```

- Antecedent može sadržati AND, OR, i NOT uslovne elemente.
- AND zahteva da svi iskazi postoje u listi
- OR zahteva da bar jedan iskaz postoji u listi
- NOT zahteva se iskaz ne poklapa ni sa jednom činjenicom iz liste.
- AND, OR, i NOT mogu biti međusobno ugnježdeni.

Uslovni element NOT

- Ako postoji NOT u antecendentu, mehanizam za zaključivanje će pretražiti tvrdnje da se „uveri“ da nema stavki koje odgovaraju uslovu pravila.
- Nove promenljive se ne mogu prvi put u pravilu uvoditi pod dejstvom NOT klauzule – mehanizam za zaključivanje neće znati šta da radi sa njima.
- NOT klauzula se najčešće javlja unutar AND klauzule, pri čemu se nove promenljive uvode u prethodnim uslovima koji su pod dejstvom AND uslovnog elementa. Na primer, ova klauzula će se podudarati sa činjenicama koje tvrde da su ptice, ali se ne tvrdi da su pingvini:
-

Uslovni element NOT

- Na primer, sledeća NOT klauzula će se upariti uslov pravila sa svim objektima za koje se tvrdi da su ptice, ali se ne tvrdi da su pingvini:

```
AND( '(?x) je ptica'
       NOT ('(?x) je pingvin')
     )
```

- Međutim, sledeće nije dozvoljeno:

```
AND( NOT( '(?x) je pingvin' )
      '(?x) je ptica'
     )
```

Primer:

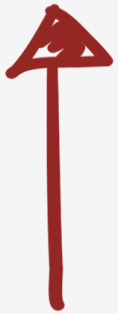
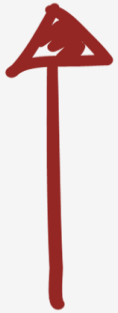
```
Pravilo za kradju:

IF 'ti imas (?x)'
THEN 'ja imam (?x)'
DELETE 'ti imas (?x)'

A0: 'ti imas svesku'
A1: 'ti imas olovku'
A2: 'ti imas gumicu'
```

Metode zaključivanja

- Ulančavanje unapred (*Forward chaining*)
  - ✓ Zaključivanje od tvrdnji ka zaključcima koji iz njih slede
  - ✓ Tokom procesa ulančavanja unapred nove tvrdnje mogu biti dodate u listu

During forward chaining, whenever an *if* pattern is observed to match an assertion, the antecedent is **satisfied**. Whenever all the *if* patterns of a rule are satisfied, the rule is **triggered**. Whenever a triggered rule establishes a new assertion or performs an action, it is **fired**.

- Ulančavanje unazad (*Backward chaining*)
  - ✓ Zaključivanje od hipoteza (potencijalnih zaključaka) ka tvrdnjama koje podržavaju hipoteze.

Forward chaining	*Backward chaining*
Zaključci	Početna hipoteza (Cilj)
	
Izvedene činjenice	Međuhipoteze (Podciljevi)
	
Činjenice	Dokazi (Činjenice)

Primer u pseudokodu i CLIPS-u

```
P0:       IF (?x je sisar)
          THEN (?x nije krokodil)

P1:       IF (?x nije krokodil)
              AND(?x zivi pod vodom)
          THEN (?x je morska krava)

P2:       IF (?x je sisar)
              AND(?x zivi pod vodom)
          THEN (?x je nilski konj)

P3:       IF (?x je krokodil)
              OR (?x je nilski konj))
          THEN (?x je opasan)

P4:       IF (?x nije krokodil)
          THEN (?x je bezopasna)

A0: (Spaki je sisar)
A1: (Fibi je sisar)
A2: (Fibi zivi pod vodom)
A3: (Roki je krokodil))
```



```
(defrule p0
          (zivotinja ?x je sisar)
          =>
     (assert (zivotinja ?x nije krokodil)))

(defrule p1
          (zivotinja ?x nije krokodil)
          (zivotinja ?x zivi pod vodom)
          =>
     (assert (zivotinja ?x je morska krava)))

(defrule p2
          (zivotinja ?x je sisar)
          (zivotinja ?x zivi pod vodom)
          =>
     (assert (zivotinja ?x je nilski konj)))


(defrule p3
          (or (zivotinja ?x je krokodil)
              (zivotinja ?x je nilski konj))
=>
(assert (zivotinja ?x je opasna)))

(defrule p4
          (zivotinja ?x is nije krokodil)
          =>
     (assert (zivotinja ?x je bezopasna)))

(assert  (zivotinja Spaki je sisar)
(zivotinja Fibi je sisar)
(zivotinja Fibi zivi pod vodom)
(zivotinja Roki je krokodil))
```

Ulančavanje unazad

- Kada pokušava da dokaže hipotezu, proces ulančavanja unazad prvo pokušava da nađe odgovarajuću tvrdnju (matching assertion) u listi.
- Ako odgovarajuća tvrdnja nije pronađena, traži se pravilo koje ima zaključak koji se poklapa sa hipotezom.
- Uslovi nađenog pravila se nadalje posmatraju kao hipoteze i pokušava se ulančavanje unazad od njih, ka tvrdnjama u listi, kroz ponavljanje koraka 1, 2 i 3.
- Tokom procesa ulančavanja unazad nema dodavanja novih činjenica u listu
- Ako postoji konflikt, uvek se bira pravilo koje je prvo na redu
- Ako postoji uparivanje uslova (niza uslova) jednog pravila sa više različitih tvrdnji (grupa tvrdnji) u listi onda se uzima tvrdnja koja je prva po redu.
- Ulančavanje unazad se može predstaviti AND-OR stablima

Ulančavanje unazad

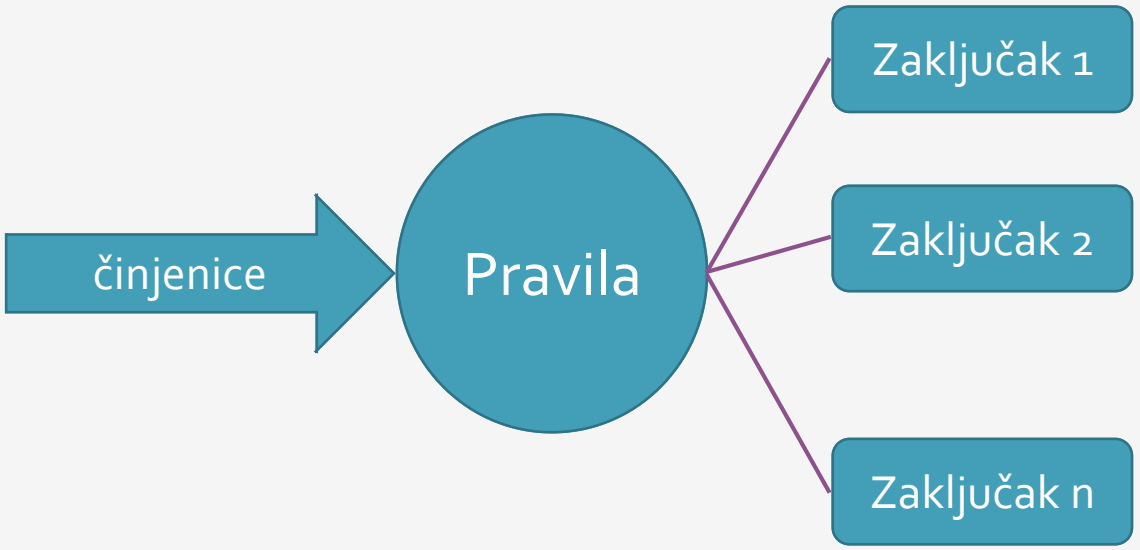
- Ulančavanje unazad se može predstaviti AND-OR stablima
- U koren stabla stavljamo hipotezu koju želimo da dokažemo.
- Ako je hipoteza među tvrdnjama, onda smo je dokazali.
- Ako nije, tražimo pravilo čiji se konsekvent poklapa sa našom hipotezom i stablo razvijamo, tako što dodajemo nove čvorove koji odgovaraju uslovima pravila.

Ulančavanje unazad

- Na primer, ako pokušavam da dokažem da je **`Roki je krokodil`**, odmah sam gotova jer imam tvrdnju koja ide u prilog mojoj hipotezi.
- To je lako gotovo. Ali najčešće situacija nije tako trivijalna. Neka treba na primer da dokažemo hipotezu **`Roki je opasan`**
- Toga nema među tvrdnjama.
- Tražimo da li postoji pravilo sa takvim zaključkom. Nalazimo pravilo P3.
- Gledamo da li imamo tvrdnje koje se poklapaju sa uslovima ovog pravila.
- Pošto je dovoljno da važi jedan od dva uslova vezanih OR logičkim veznikom, a mi imamo tvrdnju da je **`Roki je krokodil`**, onda možemo da kažemo da je hipoteza dokazana.

Kada se koristi ulančavanje unazad?

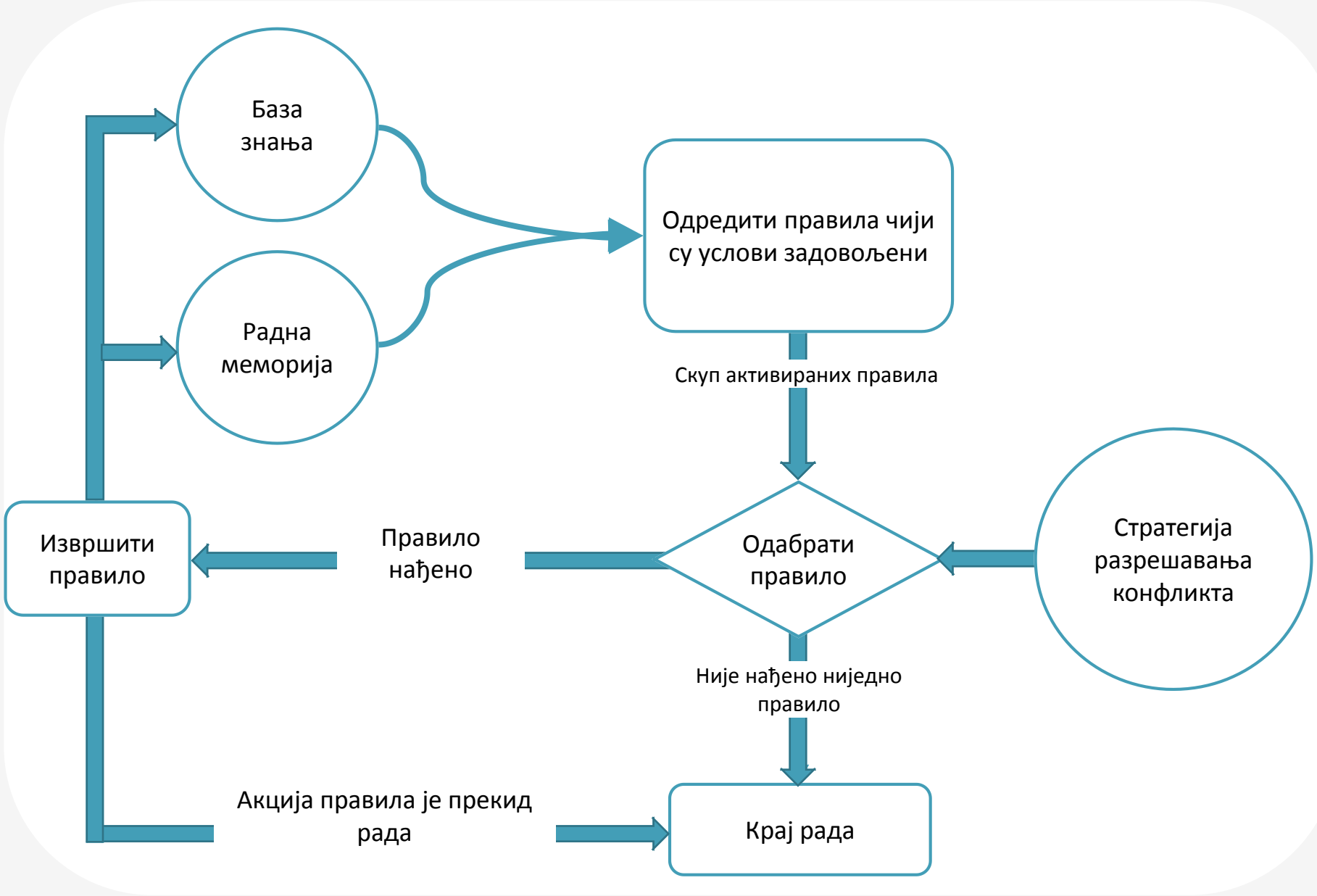
- Kada su pravila takva da se na osnovu određenog skupa činjenica može doneti više zaključaka
- FAN-OUT

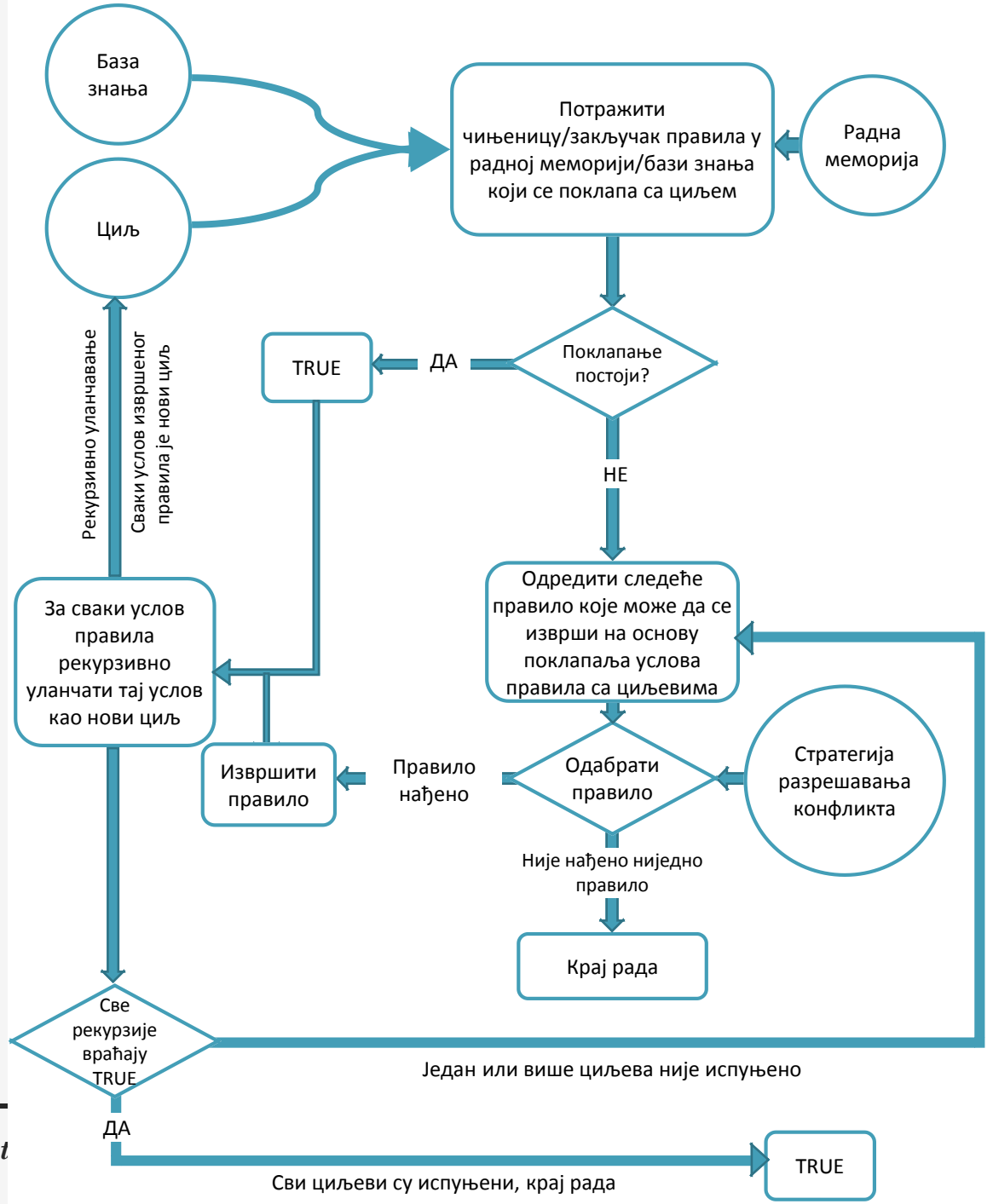


- Ako nemamo činjenice, a zanima nas da li važi jedan konkretan zaključak, koristićemo ulančavanje unazad.

Kada se koristi ulančavanje unapred?

- Kada dokazivanje određene hipoteza vodi ka razmatranju velikog broja pravila, bolje je koristiti ulančavanje unapred.
- FAN-IN
- Ako na raspolaganju imamo veliki broj činjenica, i želimo da razmotrimo šta se sve na osnovu njih može zaključiti, onda koristimo ulančavanje unapred.


База знања
Радна меморија
Одредити правила чији су услови задовољени
Скуп активираних правила
Одабрати правило
Стратегија разрешавања конфликта
Правило нађено
Извршити правило
Није нађено ниједно правило
Акција правила је прекид рада
Крај рада


База знања
Циљ
Потражити чињеницу/закључак правила у радној меморији/бази знања који се поклапа са циљем
Радна меморија
Поклапање постоји?
ДА
TRUE
НЕ
Рекурзивно уланчавање
Сваки услов извршеног правила је нови циљ
За сваки услов правила рекурзивно уланчати тај услов као нови циљ
Одредити следеће правило које може да се изврши на основу поклапања услова правила са циљевима
Одабрати правило
Стратегија разрешавања конфликта
Правило нађено
Извршити правило
Није нађено ниједно правило
Крај рада
Све рекурзије враћају TRUE
Један или више циљева није испуњено
ДА
Сви циљеви су испуњени, крај рада
TRUE

Prednosti ES

- Rešava probleme podjednako dobro ili bolje od eksperta iz date oblasti.
- Cena ekspertize po korisniku je niža.
- Ne može dati otkaz, ili otići u penziju.
- Često daje odgovor brže nego čovek ekspert.
- Ne uključuje emocije u svoj rad.
- Može sadržati znanja više eksperata.
- Može se koristiti u okruženjima opasnim za ljude.
- Objašnjava i opravdava svoja rešenja.

Klase ekspertnih sistema

- Konfiguracija
- Dijagnoza
- Instruisanje
- Interpretacija
- Monitoring
- Planiranje
- Prognoziranje
- Kontrola

Neke oblasti primene ES

- Medicina (MYCIN dijagnoza bakterijskih infekcija)
- Hemija (SPEX planiranje eksperimenata u molekularnoj biologiji)
- Elektronika (CADHELP instruisanje dizajniranja uz pomoć kompjutera)
- Geologija (PROSPECTOR interpretacija geoloških podataka o mineralima) ...