IME I PREZIME:_____ Ak. god. 2019./2020.

JMBAG:_____

1. domaća zadaća iz Formalnih metoda u oblikovanju sustava

NuSMV

Najprije je potrebno instalirati sustav NuSMV prema uputama u datoteci "NuSMV_upute_2020.pdf"

1. dio

1.1. Prouči primjer mutex_1ex.smv.

1.2. Specificiraj i napiši u CTL notaciji obilježje sigurnosti (engl. *safety property*):
«Dva procesa ne mogu biti istovremeno u kritičnom odsječku.»

Potrebno je napisati dva oblika obilježja:
a) specifikacija da je moguće jedno nepoželjno ponašanje (engl. *refutation*)
b) specifikacija da nema nepoželjnog ponašanja
Nepoželjno ponašanje je u ovom slučaju istovremeno nalaženje u kritičnom odsječku.

```
a) CTLSPEC EF (proc0.state = critical & proc1.state =
critical)

b) CTLSPEC AG !(proc0.state = critical & proc1.state =
critical)
```

1.3. Utvrdi pomoću sustava NuSMV je li ispunjeno navedeno obilježje. Objasni rezultat na temelju koda primjera (ne na temelju ispisa traga).

Oblik pod a) nije ispunjen, dok je oblik pod b) ispunjen, tako da je zadovoljeno obilježje sigurnosti. U liniji koda gdje proces ulazi u kritični odsječak:

```
(state = entering) & (turn = ind) : critical;
```

uz uvjet *state = entering* potrebno je i ispuniti uvjet *turn = ind*, a kako su vrijednosti varijable *ind* različite u instancama modula *user* za procese *proc0* i *proc1*, uvjet *turn = ind* ne može biti ispunjen za oba procesa istovremeno, što znači da oba procesa ne mogu biti istovremeno u kritičnom odsječku.

1.4. Specificiraj i napiši u CTL notaciji obilježje (engl. *liveness property*):
«Ako proces pokuša ući u kritični odsječak, konačno će i ući»
Specifikaciju napiši za oba procesa.

```
CTLSPEC AG (proc0.state = entering -> AF (proc0.state =
critical))
CTLSPEC AG (proc1.state = entering -> AF (proc1.state =
critical))
```

1.5. Utvrdi da li je zadovoljeno navedeno obilježje. Koji su sve problemi s ovom implementacijom?

Nije zadovoljeno navedeno obilježje životnosti. Moguć je problem da neki proces uđe u kritični odsječak i u njemu ostane beskonačno dugo, dok drugi proces nikad u njega ne uđe.

1.6. U mutex_1ex.smv dodaj ograničenje pravednosti (engl. *fairness*): svaka instanca procesa obavlja se beskonačno mnogo puta. Napiši ovdje kako ono glasi.

```
JUSTICE running
```

1.7. Ponovno provjeri prethodno obilježje. Što smo postigli s ovim ograničenjem pravednosti?

Obilježje životnosti i dalje nije zadovoljeno nakon uvođenja ovog ograničenja. Ovim ograničenjem pravednosti samo se postiglo da se svaki proces treba izvršavati beskonačno često.

1.8. U mutex_1ex.smv još dodaj ograničenje pravednosti: svaka instanca procesa ne može beskonačno dugo ostati u **kritičnom** odsječku. Napiši ovdje kako ono glasi. Provjeri sad svojstvo životnosti za mutex_1ex.smv.

```
JUSTICE !(state = critical)
```

Obilježje životnosti i dalje nije zadovoljeno nakon uvođenja ovog ograničenja. Iako je ovim ograničenjem pravednosti postignuto da svaki proces ne ostaje beskonačno dugo u kritičnom odsječku, to ne garantira da će neki proces u ući u kritični odsječak.

1.9. U mutex_1ex.smv dodaj još jedno ograničenje pravednosti: svaka instanca procesa ne može beskonačno dugo ostati u **nekritičnom** odsječku. Napiši ovdje kako ono glasi. Provjeri sad svojstvo životnosti za mutex_1ex.smv.

```
JUSTICE !(state = noncritical)
```

Obilježje životnosti zadovoljeno je nakon uvođenja ovog ograničenja. Ovim ograničenjem pravednosti postiglo se to da svaki proces ne ostaje beskonačno dugo u nekritičnom odsječku. Nakon dodavanja sva tri ograničenja, omogućeno je da svaki procesi ne ostaje beskonačno dugo ni u kritičnom ni u nekritičnom odsječku, te da se treba izvršavati beskonačno često.

1.10. Specificiraj i napiši u CTL notaciji:

«Ako proces proc0 uđe u kritični odsječak, proc0 neće ponovo ući u kritični odsječak sve dok proc1 nije prošao kroz svoj kritični odsječak.»

```
CTLSPEC AG (proc0.state = exiting -> A [!(proc0.state =
critical) U proc1.state = exiting])
```

1.11. Utvrdi je li zadovoljeno navedeno obilježje za mutex_1ex.smv (uz navedena ograničenja pravednosti). Koja obilježja protokola međusobnog isključivanja rješavaju ograničenja pravednosti prethodno navedena, a koji problem je još uvijek prisutan?

Navedeno obilježje je zadovoljeno uz navedena sva ograničenja pravednosti. Zadovoljena su obilježja sigurnosti, životnosti i svojstvo neblokiranja. Nije zadovoljeno svojstvo nedeterminiranog redoslijeda, jer postavljanjem inicijalne vrijednosti varijable *turn* u FALSE daje se prednost procesu *proc0* pri ulasku procesa u kritični odsječak.

2. dio

2.1. Prouči primjer mutex_2ex.smv

2.2. Specificiraj i napiši u CTL notaciji obilježje sigurnosti (engl. *safety property*):
«Dva procesa ne mogu biti istovremeno u kritičnom odsječku.»

Potrebno je napisati dva oblika obilježja:
a) specifikacija da je moguće jedno nepoželjno ponašanje (engl. *refutation*)
b) specifikacija da nema nepoželjnog ponašanja

```
a) CTLSPEC EF (proc0.state = critical & proc1.state =
critical)

b) CTLSPEC AG !(proc0.state = critical & proc1.state =
critical)
```

2.3. Utvrdi da li je ispunjeno zadano obilježje. Objasni rezultat na temelju koda primjera (ne na temelju ispisa traga).

Oblik pod a) nije ispunjen, dok je oblik pod b) ispunjen, tako da je zadovoljeno obilježje sigurnosti. U kodu varijabla *flag* drugog procesa (referencirana kao *oflag*) služi kao zastavica koja određuje da sve dok jedan proces ne izađe iz kritičnog odsječka, drugi proces ne može u njega ući.

```
init(flag) := FALSE;
next(state) :=
     case
          state = reading & !oflag : critical;

...

next(flag) :=
     case
          state = setflag : TRUE;
          state = exiting : FALSE;

...
```

2.4. Specificiraj i napiši u CTL notaciji obilježje (engl. *liveness property*):
«Ako proces pokuša ući u kritični odsječak, konačno će i ući»
Specifikaciju napiši za oba procesa.

```
CTLSPEC AG (proc0.state = reading -> AF (proc0.state =
critical))
CTLSPEC AG (proc1.state = reading -> AF (proc1.state =
critical))
```

2.5. Utvrdi je li zadovoljeno navedeno obilježje. Objasni koji je problem u ovoj implementaciji međusobnog isključivanja.

Nije zadovoljeno navedeno obilježje životnosti. Kako procesi mogu doći u stanje *reading* iz početnog stanja *setflag* bez ikakvih uvjeta u odnosu na drugi proces, moguća je situacija da oba procesa istovremeno završe u stanju *reading* (obje zastavice *flag* su postavljene u TRUE) i time onemoguće jedan drugome ulazak u kritični odsječak.

2.6. Specificiraj i napiši u CTL notaciji taj problem i provjeri ga pomoću NuSMV sustava.

```
CTLSPEC EF (AG (proc0.state = reading & proc1.state =
reading))
```

Sustav NuSMV javlja da je napisana specifikacija istinita.

2.7. Prouči primjer mutex_3ex.smv.

2.8. Je li zadovoljeno obilježje sigurnosti (2.dio, 2. pitanje)?

```
1) CTLSPEC EF (proc0.state = critical & proc1.state =
critical)

2) CTLSPEC AG !(proc0.state = critical & proc1.state =
critical)
```

Oblik pod 1) nije ispunjen, dok je oblik pod 2) ispunjen, tako da je zadovoljeno obilježje sigurnosti.

2.9. Je li zadovoljeno obilježje životnosti (2. dio, 4. pitanje)?

```
1) CTLSPEC AG (proc0.state = testflag -> AF (proc0.state =
critical))

2) CTLSPEC AG (proc1.state = testflag -> AF (proc1.state =
critical))
```

Obilježje sigurnosti životnosti nije zadovoljeno niti za jedan proces.

2.10. Dodajte sad ograničenja pravednosti kao kod zadataka 1.6, 1.8 i 1.9. Je li sad zadovoljeno obilježje životnosti?

Obilježje životnosti nije zadovoljeno ni nakon uvođenja ovog ograničenja.

2.11. Koji je problem u ovoj implementaciji međusobnog isključivanja (bez obzira na uključena ograničenja pravednosti)? Gdje sustav može «zapeti»? Problem specificiraj u CTL notaciji i provjeri pomoću sustava NuSMV.

Moguć je slučaj da oba procesa beskonačno dugo ostanu u stanju *testflag1*, uz uvjet da je zastavica *flag* postavljena u TRUE za oba procesa.

```
CTLSPEC EF (AG (proc0.state = testflag1) -> AG (proc1.state
= testflag1))
```

Sustav NuSMV javlja da je napisana specifikacija istinita.

2.12. Prouči primjer mutex_4ex.smv

Ovo je primjer uspješne implementacije međusobnog isključivanja. Zasniva se na rješenju kojeg je predložio T. Dekker a opisao E. W. Dijkstra.

2.13. Provjeri svojstva sigurnosti i životnosti. Jesu li zadovoljena (uz dodavanje tri ograničenja pristranosti iz 1. dijela)?

```
1) CTLSPEC EF (proc0.state = critical & proc1.state =
critical)

2) CTLSPEC AG !(proc0.state = critical & proc1.state =
critical)
```

Oblik pod 1) nije ispunjen, dok je oblik pod 2) ispunjen, tako da je zadovoljeno obilježje sigurnosti.

```
1) CTLSPEC AG (proc0.state = testflag -> AF (proc0.state = critical))

2) CTLSPEC AG (proc1.state = testflag -> AF (proc1.state = critical))
```

Obilježje životnosti je zadovoljeno za oba procesa.

2.14. Koje se ideje za kontrolu pristupa kritičnom odsječku iz prethodnih (neuspješnih) pokušaja nameću u ovom rješenju?

Ovo rješenje uvodi dvije dodatne zastavice *testturn* i *testturn1*, koje označavaju namjeru ulaska procesa *proc0*, odnosno *proc1* u kritični odsječak. Ako oba procesa pokušaju istovremeno ući u kritični odsječak, samo će jedan proces moći ući, ovisno čiji je red za ulazak, označen varijablom *turn*. Ako je jedan proces već u kritičnom odsječku, drugi proces će zauzeto čekati (engl. *busy wait*) da proces izađe iz kritičnog odsječka, za što se koriste zastavice *testturn* i *testturn1*.

2.15. Prouči primjer mutex_5ex.smv

Ovaj je primjer implementacija Petersonovog algoritma, koji predstavlja pojednostavnjenje prethodnog (Dekkerovog) algoritma.

2.16. Je li zadovoljeno obilježje sigurnosti?

```
1) CTLSPEC EF (proc0.state = critical & proc1.state = critical)

2) CTLSPEC AG !(proc0.state = critical & proc1.state = critical)
```

Oblik pod 1) nije ispunjen, dok je oblik pod 2) ispunjen, tako da je zadovoljeno obilježje sigurnosti.

2.17. Specificiraj i napiši u CTL notaciji obilježje životnosti. Provjeri ga pomoću sustava NuSMV. Je li to obilježje zadovoljeno (uz dodavanje tri ograničenja pravednosti iz 1. dijela)?

```
1) CTLSPEC AG (proc0.state = trying -> AF (proc0.state =
critical))

2) CTLSPEC AG (proc1.state = trying -> AF (proc1.state =
critical))
```

Obilježje životnosti je zadovoljeno za oba procesa.

Sustav NuSMV javlja da je napisana specifikacija istinita.

3. dio

Prouči potpoglavlja 3.1, 3.2, 3.5 i 3.7 iz NuSMV priručnika "NuSMV 2.6 User Manual". Nakon toga riješi sljedeće zadatke:

3.1. Pokreni interaktivno ljusku NuSMV-a. Učitaj model zadan datotekom mutex_1ex_int.smv.

3.2. Inicijaliziraj sustav za verifikaciju. Ukratko obrazloži što se sve događa prilikom pokretanja naredbe "go".

Naredba *go* inicijalizira sustav za verifikaciju. Ekvivalent je nizu sljedećih naredbi:

- *read_model* – učitava NuSMV datoteku u NuSMV
- *flatten_hierarchy* – poravnana hijerarhiju modula
- *encode_variables – gradi BDD varijable nužne za prevođenje modela u BDD*
- *build_flat_model* – prevodi poravnatu hijerarhiju u skalarni FSM
- *build_model* - prevodi poravnatu hijerarhiju u BDD

3.3. Simuliraj kretanje kroz 3 stanja (od proizvoljno odabranog početnoga). Navedi dvije naredbe koje se koriste da bi se to ostvarilo. Koju naredbu treba koristiti da bi se ispisao trag prolaska kroz ta stanja?

Naredbe za simulaciju kretanja kroz 3 stanja:
pick_state -i
simulate -k 3

Naredba za ispis traga prolaska kroz stanja:
show_traces

3.4. Provjeri stroj s konačnim brojem stanja. Kakva je relacija prijelaza tog automata? Može li doći do potpunog zastoja?

Stroj s konačnim brojem stanja provjerava se naredbom *check_fsm*. Sustav NuSMV javlja da je relacija prijelaza tog automata potpuna te da ne može doći do potpunog zastoja, jer ne postoji to stanje.

3.5. Koliko ukupno postoji stanja u modelu, a koliko postoji dosezljivih (engl. *reachable*) stanja? (napomena: *diameter* - promjer FSM-a je minimalan broj koraka potrebnih da bi se došlo do svih dosezljivih stanja)

Naredbom *print_reachable_states* dobivaju se sljedeći podaci:

- broj ukupnih stanja: 32 (2^5)
- broj dosezljivih stanja: 16 (2^4)
- promjer FSM-a: 7

3.6. Provjeri prvu po redu CTL specifikaciju (redni broj 0). Je li ona istinita ili lažna? Koje obilježje protokola međusobnog isključivanja se njome provjerava? Je li to obilježje zadovoljeno?

```
CTLSPEC EF (proc0.state = critical & proc1.state =
critical)
```

Prva CTL specifikacija provjerava se naredbom *check_ctlspec -n 0*. NuSMV javlja da je ta specifikacija lažna. Njome se provjerava obilježje sigurnosti i ono je zadovoljeno.

3.7. Provjeri drugu po redu CTL specifikaciju (redni broj 1). Je li ona istinita ili lažna? Koje obilježje protokola međusobnog isključivanja se njome provjerava? Je li to obilježje zadovoljeno?

```
CTLSPEC AG (proc0.state = entering -> AF proc0.state =
critical)
```

Druga CTL specifikacija provjerava se naredbom *check_ctlspec -n 1*. NuSMV javlja da je ta specifikacija istinita. Njome se provjerava obilježje životnosti i ono je zadovoljeno.

3.8. Sada ukloni obilježja pravednosti iz datoteke mutex_1ex_int.smv, ponovi postupak učitavanja i pripreme za verifikaciju te provjeri drugu po redu CTL specifikaciju. Ima li kakve promjene u odnosu na prethodni zadatak?

Nakon uklanjanja obilježja pravednosti, NuSMV javlja da je druga CTL specifikacija lažna, što sa sobom povlači da obilježje životnosti više nije zadovoljeno.

3.9. Prouči naredbe za provjeru svojstava sustava za rad u stvarnom vremenu koje su zadane s ključnom riječi "COMPUTE" u datoteci mutex_1ex_int.smv. Koje je značenje svake od tih naredbi?

```
COMPUTE MIN[proc0.state = noncritical, proc0.state =
exiting]
```

- naredba vraća duljinu najkraćeg puta od stanja *noncritical* do stanja *exiting* za proces *proc0*

```
COMPUTE MAX[proc0.state = noncritical, proc0.state =
critical]
```

- naredba vraća duljinu najdužeg puta od stanja *noncritical* do stanja *exiting* za proces *proc0*

3.10. Provjeri te naredbe u sustavu NuSMV (prva COMPUTE naredba ima redni broj 2 u modelu, a druga redni broj 3). Navedi rezultat izvođenja tih dviju naredbi. Uzima li naredba COMPUTE u obzir navedena ograničenja pravednosti?

```
COMPUTE MIN[proc0.state = noncritical, proc0.state =
exiting]
```

Naredba za provjeru prve *COMPUTE* naredbe je *check_compute -n 2* i vraća kao rezultat: 3.

```
COMPUTE MAX[proc0.state = noncritical, proc0.state =
critical]
```

Naredba za provjeru druge *COMPUTE* naredbe je *check_compute -n 3* i vraća kao rezultat: beskonačno.

Rezultati navedenih *COMPUTE* naredbi jednaki su sa i bez navedenim ograničenjima pravednosti, tako da one ne uzimaju u obzir ograničenja pravednosti.

4. dio

4.1. Prouči primjer ferryman.smv.

4.2. Specificiraj i napiši u CTL notaciji obilježje:
«Ne postoji siguran put kojim se dolazi do cilja problema.»
Pritom se u specifikaciji trebaju koristiti već definirane makro-instrukcije programa.

```
CTLSPEC !E [safe_state U goal]
```

4.3. Provjeri zadano svojstvo. Je li ono zadovoljeno? Što nam u ovom slučaju daje ispis traga programa? Opiši redoslijed izvođenja kojim se uspješno dolazi do cilja problema.

Zadano svojstvo nije zadovoljeno, jer postoji siguran put kojim se dolazi do cilja problema. Ispis traga programa nam daje protuprimjer u kojem je pokazan redoslijed kojim skeledžija (engl. *ferryman*) uspješno prevozi vuka, ovcu i kupus.
Redoslijed izvođenja je sljedeći (u početku se svi nalaze na desnoj strani obale):
1. Prevezi kozu na lijevu stranu
2. Vrati se na desnu stranu
3. Prevezi kupus na lijevu stranu
4. Prevezi kozu na desnu stranu
5. Prevezi vuka na lijevu stranu
6. Vrati se na desnu stranu
7. Prevezi kozu na lijevu stranu

4.4. Zadani kôd u NuSMV-u sadrži implicitni nedeterminizam uzrokovan varijablom *request*. Izmijeni zadani kôd tako da sadrži **isključivo** eksplicitni nedeterminizam (napomena: ne mijenjati tipove varijabli!).

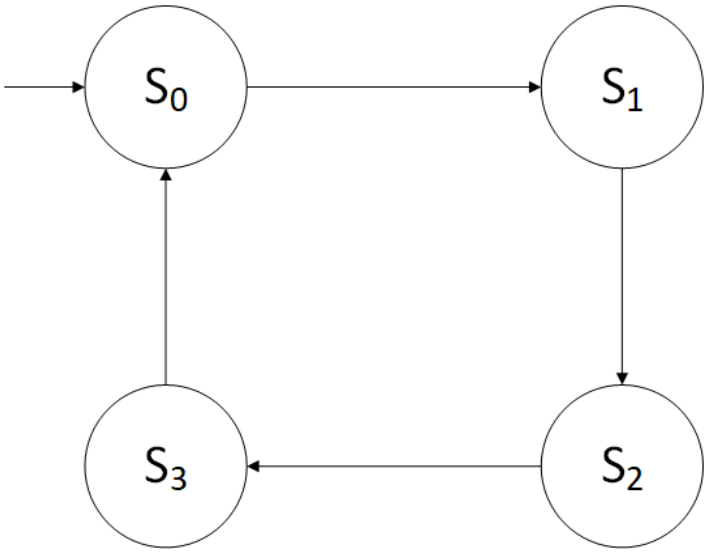
```
MODULE main
VAR
      request: boolean;
      flag: {red, blue};
ASSIGN
      init(flag) := blue;
      next(flag) := case
                        request = FALSE : blue;
                        TRUE : red;
      esac;
```

Rješenje:

```
MODULE main
VAR
     req: {0, 1};
     request: boolean;
     flag: {red, blue};
ASSIGN
     init(req) := {0, 1};
     init(flag) := blue;
     next(flag) := case
                    request = FALSE : blue;
                    TRUE : red;
     esac;
     next(request) := case
                    req = 1 : FALSE;
                    req = 0 : TRUE;
     esac;
     next(req) := {0, 1};
```

4.5. Za zadani kôd u NuSMV-u nacrtaj odgovarajuću Kripke strukturu i odredi:
   a) skup svih mogućih stanja – S_A
   b) skup svih dosezljivih stanja – S_R (uz pretpostavku da su sva početna stanja dosezljiva)

```
MODULE main
VAR
      request : boolean;
      status : {ready, busy};
      negReq : boolean;
ASSIGN
      init(request) := FALSE;
      init(status) := busy;
      init(negReq) := TRUE;
      next(request) := case
                        (status = ready) : FALSE;
                        TRUE: TRUE;
      esac;
      next(status) := case
                        request : ready;
                        TRUE : busy;
      esac;
      next(negReq) := !request;
```



S_0 = {request=FALSE, status=busy, negReq=TRUE}
S_1 = {request=TRUE, status=busy, negReq=TRUE}
S_2 = {request=TRUE, status=ready, negReq=FALSE}
S_3 = {request=FALSE, status=ready, negReq=FALSE}
S_4 = {request=TRUE, status=busy, negReq=FALSE}
S_5 = {request=TRUE, status=ready, negReq=TRUE}
S_6 = {request=FALSE, status=ready, negReq=TRUE}
S_7 = {request=FALSE, status=busy, negReq=FALSE}

a) $S_A = \{S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7\}$

b) $S_R = \{S_0, S_1, S_2, S_3\}$