Univerzitet u Beogradu
Matematički fakultet
Katedra za računarstvo i informatiku

Šifra predmeta: R265
04.04.2023.

Uvod u interaktivno dokazivanje teorema

Vežbe 7

Zadatak 1 *Isar dokazi u logici prvog reda.*

lemma
  assumes $(\exists\ x.\ P\ x)$
    and $(\forall\ x.\ P\ x \longrightarrow Q\ x)$
   shows $(\exists\ x.\ Q\ x)$
proof −
  from *assms(1)* **obtain** *x* **where** *P x* **by** − (*erule exE*)
  moreover
  from *assms(2)* **have** $P\ x \longrightarrow Q\ x$ **by** (*erule-tac x=x* **in** *allE*)
  ultimately
  have *Q x* **by** − (*erule impE*, *assumption*)
  then show $(\exists\ x.\ Q\ x)$ **by** (*rule-tac x=x* **in** *exI*)
qed

lemma
  assumes $\forall\ c.\ Man\ c \longrightarrow Mortal\ c$
    and $\forall\ g.\ Greek\ g \longrightarrow Man\ g$
   shows $\forall\ a.\ Greek\ a \longrightarrow Mortal\ a$
proof
  fix *Socrates*
  show *Greek Socrates* $\longrightarrow$ *Mortal Socrates*
  proof
    assume *Greek Socrates*
    moreover
    from *assms(2)* **have** *Greek Socrates* $\longrightarrow$ *Man Socrates*
      by (*erule-tac x=Socrates* **in** *allE*)
    ultimately
    have *Man Socrates* **by** − (*erule impE*, *assumption*)
    moreover
    from *assms(1)* **have** *Man Socrates* $\longrightarrow$ *Mortal Socrates*
      by (*erule-tac x=Socrates* **in** *allE*)
    ultimately
    show *Mortal Socrates*
      by − (*erule impE*, *assumption*)
  qed
qed

Dodatni primeri:

Ako svaki konj ima potkovice;
i ako ne postoji čovek koji ima potkovice;
i ako znamo da postoji makar jedan čovek;
dokazati da postoji čovek koji nije konj.

lemma

```
  assumes ∀ x. konj x ⟶ potkovice x
      and ¬ (∃ x. covek x ∧ potkovice x)
      and ∃ x. covek x
    shows ∃ x. covek x ∧ ¬ konj x
proof −
  from assms(3) obtain x where covek x by auto
  have konj x ∨ ¬ konj x by auto
  then show ∃ x. covek x ∧ ¬ konj x
  proof
    assume konj x
    moreover
    from assms(1) have konj x ⟶ potkovice x by auto
    ultimately
    have potkovice x by auto
    with ‹covek x› have covek x ∧ potkovice x by auto
    then have ∃ x. covek x ∧ potkovice x by auto
    with assms(2) have False by auto
    then show ∃ x. covek x ∧ ¬ konj x by auto
  next
    assume ¬ konj x
    with ‹covek x› have covek x ∧ ¬ konj x by auto
    then show ∃ x. covek x ∧ ¬ konj x by auto
  qed
qed
```

Zadatak 2 *Pravilo ccontr i classical.*

Dokazati u Isar jeziku naredna tvrđenja pomoću pravila *ccontr*.

```
lemma ¬ (A ∧ B) ⟶ ¬ A ∨ ¬ B
proof
  assume ¬ (A ∧ B)
  show ¬ A ∨ ¬ B
  proof (rule ccontr)
    assume ¬ (¬ A ∨ ¬ B)
    have A ∧ B
    proof
      show A
      proof (rule ccontr)
        assume ¬ A
        then have ¬ A ∨ ¬ B
          by (rule disjI1)
        with ‹¬ (¬ A ∨ ¬ B)› show False
          by − (erule notE, assumption)
      qed
    next
      show B
      proof (rule ccontr)
        assume ¬ B
        then have ¬ A ∨ ¬ B
          by (rule disjI2)
        with ‹¬ (¬ A ∨ ¬ B)› show False
```

```
      by − (erule notE, assumption)
    qed
  qed
  with ‹¬ (A ∧ B)› show False
    by − (erule notE, assumption)
 qed
qed
```

Dodatni primer:

```
lemma ((P ⟶ Q) ⟶ P) ⟶ P
proof
  assume (P ⟶ Q) ⟶ P
  show P
  proof (rule ccontr)
    assume ¬ P
    have P ⟶ Q
    proof
      assume P
      with ‹¬ P› have False by auto
      then show Q by auto
    qed
    with ‹(P ⟶ Q) ⟶ P› have P by auto
    with ‹¬ P› show False by auto
  qed
qed
```

Dokazati u Isar jeziku naredna tvrđenja pomoću pravila *classical*.

```
lemma P ∨ ¬ P
proof (rule classical)
  assume ¬ (P ∨ ¬ P)
  show P ∨ ¬ P
  proof (rule disjI1)
    show P
    proof (rule classical)
      assume ¬ P
      then have P ∨ ¬ P
        by (rule disjI2)
      with ‹¬ (P ∨ ¬ P)› have False
        by − (erule notE, assumption)
      then show P using FalseE[of P]
        by − (assumption)
    qed
  qed
qed
```

Zadatak 3 *Logčki lavirinti.*

Svaka osoba daje potvrdan odgovor na pitanje: *Da li si ti vitez?*

```
lemma no-one-admits-knave:
  assumes k ⟷ (k ⟷ ans)
    shows ans
```

```
proof (cases k)
  assume k
  with assms have k ⟷ ans  by auto
  with ‹k› show ?thesis by auto
next
  assume ¬ k
  with assms have ¬ (k ⟷ ans) by auto
  then have ¬ k ⟶ ans by auto
  with ‹¬ k› show ?thesis by auto
qed
```

Abercrombie je sreo tri stanovnika, koje ćemo zvati A, B i C. Pitao je A: Jesi li ti vitez ili podanik? On je odgovorio, ali tako nejasno da Abercrombie nije mogao shvati što je rekao. Zatim je upitao B: Šta je rekao? B odgovori: Rekao je da je podanik. U tom trenutku, C se ubacio i rekao: Ne verujte u to; to je laž! Je li C bio vitez ili podanik?

```
lemma Smullyan-1-1:
  assumes kA ⟷ (kA ⟷ ansA)
      and kB ⟷ ¬ ansA
      and kC ⟷ ¬ kB
    shows kC
proof –
  from assms(1) have ansA using no-one-admits-knave[of kA ansA] by simp
  with assms(2) have ¬ kB by simp
  with assms(3) show kC by simp
qed
```

Prema drugoj verziji priče, Abercrombie nije pitao A da li je on vitez ili podanik (jer bi unapred znao koji će odgovor dobiti), već je pitao A koliko od njih trojice su bili vitezovi. Opet je A odgovorio nejasno, pa je Abercrombie upitao B što je A rekao. B je tada rekao da je A rekao da su tačno njih dvojica podanici. Tada je, kao i prije, C tvrdio da B laže. Je li je sada moguće utvrditi da li je C vitez ili podanik?

```
definition exactly-two :: bool ⇒ bool ⇒ bool ⇒ bool where
  exactly-two A B C ⟷ ((A ∧ B) ∨ (A ∧ C) ∨ (B ∧ C)) ∧ ¬ (A ∧ B ∧ C)

lemma Smullyan-1-2:
  assumes kB ⟷ (kA ⟷ exactly-two (¬ kA) (¬ kB) (¬ kC))
      and kC ⟷ ¬ kB
    shows kC
proof(cases kC)
  case True
  then show ?thesis by auto
next
  case False
  with assms(2) have kB by auto
  with assms(1) have ∗:kA ⟷ exactly-two (¬ kA) (¬ kB) (¬ kC) by auto
  have False
  proof (cases kA)
    case True
    with ∗ have exactly-two (¬ kA) (¬ kB) (¬ kC) by auto
    with ‹kA› ‹kB› ‹¬ kC› show ?thesis using exactly-two-def by auto
  next
    case False
```

```
    with ∗ have ¬ exactly-two (¬ kA) (¬ kB) (¬ kC) by auto
    with ‹¬ kA› ‹kB› ‹¬ kC› show ?thesis using exactly-two-def by auto
  qed
  then show ?thesis by auto
qed
```

Dodatni primer:

Abercrombie je sreo samo dva stanovnika A i B. A je izjavio: Obojica smo podanici. Da li možemo da zaključimo šta je A a šta je B?

```
lemma Smullyan-1-3:
  assumes kA ⟷ ¬ kA ∧ ¬ kB
  shows ¬ kA ∧ kB
proof (cases kA)
  case True
  with assms have ¬ kA ∧ ¬ kB by auto
  then have ¬ kA by auto
  with ‹kA› have False by auto
  then show ?thesis by auto
next
  case False
  with assms have ¬ (¬ kA ∧ ¬ kB) by auto
  then have kA ∨ kB by auto
  then show ?thesis
  proof
    assume kA
    with ‹¬ kA› have False by auto
    then show ?thesis by auto
  next
    assume kB
    with ‹¬ kA› show ?thesis by auto
  qed
qed
```