Práctico 2

Asistentes de pruebas para lógicos y matemáticos

Cálculo de Predicados y Cálculo de Predicados con Igualdad

Cálculo de Predicados

Objetivos

Hacer pruebas formales en cálculo predicados, para ello se debe aprender a manejar las nociones de proposición, predicado, cuantificador, etc. Utilizar tácticas automáticas de búsqueda de pruebas para el cálculo predicados (se usarán tácticas tales como `auto` y `trivial`). Utilizar estructuradores de tácticas (*tacticals*) en las pruebas.

Principales tácticas a utilizar en estos ejercicios

`apply <term>`: puede usarse para eliminar un cuantificador universal.

`exists <term>`: para probar un existencial indicando que `<term>` es el testigo (witness) del mismo.

`Section <nombre>`: para estructurar las pruebas (permite declarar variables y lemas de forma local).

Predicados

En lógica de primer orden deben considerarse dos categorías de expresiones: aquellas que denotan objetos (los términos) y las que denotan propiedades de los objetos (las fórmulas). En *Coq* hacemos esta diferencia explícita definiendo un conjunto U al cual pertenecerán los objetos. Esto se hace declarando la variable `U:Set` en el contexto.

Un predicado unario P se representa como una función proposicional que depende de un objeto en U (en símbolos: `P:U->Prop`), un predicado binario R se representa como una función proposicional que depende de dos objetos de U (en símbolos: `R:U->U->Prop`). En general, un predicado n-ario P se representa como una función proposicional que depende de n objetos de U.

Cuantificador universal

La fórmula ∀x∈U. B se escribe en Coq como `(forall x:U, B)` (donde `x` puede aparecer libre en `B`). En cálculo de predicados se tiene usualmente un solo dominio de objetos, por lo cual, se suele escribir sólo ∀x B. Sin embargo, en Coq los objetos pueden pertenecer a diferentes dominios (tipos), por lo cual debemos "tipar" las variables.

El alcance del cuantificador universal en `(forall x:U, B)` es `B`. Las nociones de ocurrencia libre y ligada son las usuales.

Convención de parentización: las implicancias `->` y los cuantificadores universales se asocian a la derecha. Por ejemplo `forall x:U, B->C` denota $\forall x \in U.\ (B \rightarrow C)$.

La táctica para eliminar un cuantificador universal es `apply <term>`. Por ejemplo si hay una hipótesis `H: forall (x1:A1)...(xn:An), B` en el contexto y el objetivo a probar es G , la aplicación de la regla de eliminación puede ser de una de las siguientes formas:

- `apply H` (si la cabeza de `B` unifica con G),
- `apply (H a1...an)` (si Coq no puede resolver la unificación de `B` y G y es necesario instanciar manualmente el lema `H` con los valores `a1:A1,... an:An)`.

Ejercicio 2.1. Considere las siguientes declaraciones de variable.

```
Variable U: Set.
Variable A B: U->Prop.
Variable P Q: Prop.
Variable R S: U->U->Prop.
```

Demuestre en Coq los siguientes teoremas:

1. $(\forall x \in U.\ A(x)) \rightarrow \forall y \in U.\ A(y)$
2. $(\forall x,y \in U.\ R(x,y)) \rightarrow \forall x,y \in U.\ R(y,x)$
3. $(\forall x \in U.\ (A(x) \rightarrow B(x))) \rightarrow ((\forall y \in U.\ A(y)) \rightarrow (\forall z \in U.\ B(z)))$

Ejercicio 2.2. Demuestre en Coq los siguientes teoremas.

1. $\forall x \in U.\ (A(x) \rightarrow \sim \forall x \in U.\ \sim A(x))$
2. $\forall x,y \in U.\ R(x,y) \rightarrow \forall x \in U.\ R(x,x)$
3. $(\forall x \in U.\ (P \rightarrow A(x))) \rightarrow (P \rightarrow \forall x \in U.\ A(x))$
4. $(\forall x \in U.\ (A(x) \wedge B(x))) \rightarrow ((\forall x \in U.\ A(x)) \rightarrow (\forall x \in U.\ B(x)))$
5. $((\forall x \in U.\ A(x)) \vee (\forall x \in U.\ B(x))) \rightarrow (\forall x \in U.\ \sim(\sim A(x) \wedge \sim B(x)))$

Ejercicio 2.3. Sea R una relación binaria sobre elementos de un Set A. Pruebe que:

1. Si R satisface: $\forall x\ R(x,x)$ y $\forall x \forall y \forall z(\ R(x,y) \wedge R(x,z) \rightarrow R(y,z)\ )$ entonces R es una relación de equivalencia (reflexiva, simétrica y transitiva).
2. Si R es una relación asimétrica entonces esa relación es irreflexiva. Donde: R es *irreflexiva* si y solamente si ningún objeto está relacionado consigo mismo, y R es *asimétrica* si y solamente si en caso de que *x* esté relacionado con *y* entonces *y* no lo está con *x*, cualesquiera sean los objetos *x* e *y*.

Cuantificador Existencial

La fórmula $\exists x \in U.\ B$ se escribe en Coq `(exists x:U, B)` (donde `x` puede aparecer libre en `B`).

El alcance del cuantificador existencial `(exists x:A, B)` es `B`. Las nociones de ocurrencia libre y ligada son las usuales.

La táctica correspondiente a la regla de *introducción* del cuantificador existencial es `exists <term>`. Por ejemplo, si el objetivo a probar es `(exists x:A, B)` y `a` es un elemento de `A` en el contexto, la táctica `(exists a)` devolverá como nuevo objetivo a probar la fórmula `B[a/x]` (o sea la fórmula `B` en la que las ocurrencias libres de la variable `x` fueron sustituidas por `a`). El término `a` es llamado el testigo (witness) de la prueba del existencial.

La táctica `elim` aplicada a una hipótesis (o lema) cuyo tipo es de la forma `(exists x:A, B)`, transforma al objetivo original G en `(forall x:A, B->G)`. Esto corresponde exactamente con la regla de eliminación del cuantificador existencial.

Ejercicio 2.4.

Usa `exists` (entre otros).

Demuestre en Coq los siguientes teoremas (tenga en cuenta que para probar un existencial, deberá proveer el testigo):

1. $\exists x \in U.\ \exists y \in U.\ R(x,y) \rightarrow \exists y \in U.\ \exists x \in U.\ R(x,y)$
2. $\forall x \in U.\ A(x) \rightarrow \sim \exists x \in U.\ \sim A(x)$
3. $\exists x \in U.\ \sim A(x) \rightarrow \sim \forall x \in U.\ A(x)$
4. $\forall x \in U.\ A(x) \wedge B(x) \rightarrow \forall x \in U.\ A(x) \wedge \forall x \in U.\ B(x)$
5. $\exists x \in U.\ A(x) \vee B(x) \rightarrow \exists x \in U.\ A(x) \vee \exists x \in U.\ B(x)$
6. $\forall x \in U.\ A(x) \vee \forall x \in U.\ B(x) \rightarrow \forall x \in U.\ A(x) \vee B(x)$

Ejercicio 2.5.

Usa `Section` (entre otros).

Considere las siguientes declaraciones que definen al conjunto de los números naturales y a los predicados par (`even`) e impar (`odd`):

```
Section Naturales.
Variable nat: Set.
Variable S: nat->nat.
```

```
Variable a b c: nat.
Variable odd even: nat->Prop.
Variable P Q: nat->Prop.
Variable f: nat -> nat.
```

Demuestre bajo este contexto los siguientes teoremas:

1. $\forall x \in nat.\ \exists y \in nat.\ (P(x) \rightarrow P(y))$
2. $\exists x \in nat.\ P(x) \rightarrow (\forall y \in nat.\ P(y) \rightarrow Q(y)) \rightarrow \exists z \in nat.\ Q(z)$
3. $even(a) \rightarrow \forall x \in nat.\ (even(x) \rightarrow odd\ (S(x))) \rightarrow \exists y \in nat.\ odd(y)$
4. $(\forall x \in nat.\ P(x) \wedge odd(x) \rightarrow even(f(x)))$
   - $\rightarrow$ $(\forall x \in nat.\ even(x) \rightarrow odd(S(x)))$
   - $\rightarrow$ $even(a)$
   - $\rightarrow$ $P(S(a))$
   - $\rightarrow$ $\exists z \in nat.\ even(f(z))$

```
End Naturales.
```

Ejercicio 2.6.

Considere el siguiente conjunto de declaraciones que definen al conjunto de naturales ordenados según el $\leq$ (`le`).

```
Section NaturalesOrdenados.
Variable nat: Set.
Variable S: nat -> nat.
Variable le: nat -> nat -> Prop.
Variable f: nat -> nat.
Variable P: nat -> Prop.

Axiom le_n: forall n:nat, (le n n).
Axiom le_S: forall n m:nat, (le n m) -> (le n (S m)).
Axiom monoticity: forall n m:nat, (le n m) -> (le (f n) (f m)).
```

1. Demuestre los siguientes lemas
   a. Lema **le_x_Sx** : $\forall x \in nat.\ x \leq x+1$.
   b. Lema **le_x_SSx** : $\forall x \in nat.\ x \leq x+2$.
2. Dé tres pruebas diferentes la siguiente fórmula existencial, dando en cada caso un testigo diferente.

   Teorema **T1** : $\forall a \in nat.\ \exists b \in nat.\ f(a) \leq b$.

   a. Demuestre el teorema anterior dando como testigo del existencial el natural

`(S (S (S (S (S (f a))))))`. Reemplace en su prueba las primeras líneas de la forma `apply le_S` por `repeat apply le_S`.

b. Lea la sección Tacticals del manual de referencia y rehaga la prueba utilizando el estructurador de tácticas `do`.

```
End NaturalesOrdenados.
```

Ejercicio 2.7. Utilice los estructuradores de tácticas (de la sección *Tacticals* del manual de referencia) para escribir una prueba de los siguientes teoremas en una sola línea:

1. $(\forall x \in U.\ (A(x) \wedge B(x))) \rightarrow (\forall x \in U.\ A(x)) \wedge (\forall x \in U.\ B(x))$
2. $(\exists x \in U.\ (A(x) \vee B(x))) \rightarrow (\exists x \in U.\ A(x)) \vee (\exists x \in U.\ B(x))$
3. $(\forall x \in U.\ A(x)) \vee (\forall x \in U.\ B(x)) \rightarrow (\forall x \in U.\ (A(x) \vee B(x))$

Ejercicio 2.8. Sea R una relación binaria sobre elementos de un conjunto U y sean T y V relaciones unarias sobre U.

1. Pruebe sin usar tácticas automáticas el siguiente teorema: $\exists y \forall x\ R(x,y) \rightarrow \forall x \exists y\ R(x,y)$.
2. Pruebe el siguiente teorema:

```
Theorem T282:(exists y:U, True)/\(forall x:U, (T x) \/ (V x)) ->
      (exists z:U, (T z)) \/ (exists w:U, (V w)).
```

3. Es necesaria la condición `(exists y:U, True)` en el teorema anterior? Justifique.

Ejercicio 2.9. Demuestre en Coq los siguientes teoremas de la lógica clásica:

1. Lemma `not_ex_not_forall`: $(\sim \exists x \in U.\ \sim A(x)) \rightarrow (\forall x \in U.\ A(x))$.
2. Lemma `not_forall_ex_not`: $(\sim \forall x \in U.\ A(x)) \rightarrow (\exists x \in U.\ \sim A(x))$.

Cálculo de predicados con igualdad

Objetivos

Comprender cómo realizar pruebas utilizando igualdades.

Principales tácticas a utilizar en estos ejercicios

[`reflexivity`] permite probar los objetivos de la forma `a=a`.

[`symmetry`] Si el objetivo corriente es de la forma `a=b`, entonces da como nuevo objetivo

`b=a.`

[`transitivity <term>`] Se aplica a un objetivo de la forma `a=b` y lo reemplaza por los objetivos `a=<term>` y `<term>=b`.

[`rewrite -> <term>`] Si `<term>`:`a=b` entonces reescribe en el objetivo corriente todas las ocurrencias de `a` por `b`. `Rewrite <- <term>` hace lo contrario.

[`replace a with b`] Realiza el mismo trabajo que `Rewrite` pero pide probar la igualdad `a=b`.

[`pattern` …] esta familia de tácticas permiten, al componerlas con las de la familia `rewrite`, reescribir sólo ciertas ocurrencias (no serán necesarias en este práctico).

Ejercicio 2.10.

Considere las siguientes declaraciones en Coq:

```
Section Sec_Peano.
 Variable nat: Set.
 Variable O: nat.
 Variable S: nat->nat.
 Axiom disc: forall n:nat, ~O=(S n).
 Axiom inj : forall n m:nat, (S n)=(S m) -> n=m.
 Axiom allNat : forall n: nat, n = O \/ exists m: nat, S m = n.


 Variable sum prod: nat->nat->nat.
 Axiom sum0 : forall n:nat, (sum n O)=n.
 Axiom sumS : forall n m:nat, (sum n (S m))=(S (sum n m)).
 Axiom prod0 : forall n:nat, (prod n O)=O.
 Axiom prodS : forall n m:nat, (prod n (S m))=(sum n (prod n m)).
```

Bajo este contexto, demuestre en Coq, los siguientes teoremas:

1. `Lemma L10_1: (sum (S O) (S O)) = (S (S O)).`
2. `Lemma L10_2: forall n:nat, ~(O=n /\ (exists m:nat, n = S m)).`
3. `Lemma prod_neutro: forall n:nat, (prod n (S O))= n.`
4. `Lemma diff: forall n:nat, ~(S (S n))=(S O).`
5. `Lemma L10_3: forall n: nat, exists m: nat, prod n (S m) = sum n n.`
6. `Lemma L10_4: forall m n: Nat, n <> O -> sum m n <> O.`
7. `Lemma L10_5: forall m n: Nat, sum m n = O -> m = O /\ n = O.`
8. `Lemma L10_6: forall m n: Nat, prod m n = O -> m = O \/ n = O.`

Ejercicio 2.11.

Considere las siguientes declaraciones:

```
Variable le : nat->nat->Prop.
Axiom leinv: forall n m:nat, (le n m) -> n=O \/
      (exists p:nat, (exists q:nat, n=(S p)/\ m=(S q) /\ (le p q))).
```

Demuestre `Lemma notle_s_o: forall n:nat, ~(le (S n) O).`

```
End Sec_Peano.
```

Ejercicios a entregar:

Ver la fecha límite y los ejercicios requeridos en el sitio EVA del curso.

El archivo a entregar debe cargar correctamente en Coq. Si deja ejercicios sin resolver, debe delimitarlos como comentarios: (... *).*

Al inicio del archivo deben estar los datos de cada integrante; se admiten entregas individuales o de a dos estudiantes.

Usar la plantilla publicada junto con el práctico para el desarrollo de los ejercicios requeridos; no es necesario entregar los ejercicios no solicitados.