Facultad de Ciencias - UNAM
Lógica Computacional 2023-2
Práctica 6: Introducción a `Coq`

Javier Enríquez Mendoza
Ramón Arenas Ayala
Óscar Fernando Millán Pimentel
Kevin Axel Prestegui Ramos

16 de Mayo de 2023
Fecha de entrega: 28 de Mayo de 2023

1. Introducción

El flujo de ideas entre la lógica y las Ciencias de la Computación, no ha sido unidireccional a lo largo de la historia. Las CC, también han realizado grandes aportes a la lógica. Uno de estos, es el desarrollo de software que ayude a realizar demostraciones de proposiciones lógicas. Este software puede clasificarse en dos categorías:

- Automatic theorem provers.- un ejemplo básico de estos es el algoritmo Hao Wang que implementamos en la práctica pasada.
- Proof Assistants (*Asistentes de Prueba*).- Los cuales son herramientas híbridas, que automatizan la mayoría de procesos sencillos al realizar demostraciones, mientras dependen de la guía humana para aspectos más complejos.

`Coq`, es un asistente de prueba que ha estado en desarrollo desde 1983, y que en años recientes ha obtenido popularidad entres usuarios tanto en la parte de investigación como en la industria. Esté se puede ver como una combinación de un pequeño lenguaje funcional (`Gallina`) y un conjunto de herramientas para la demostración de afirmaciones lógicas.

2. Objetivo

El objetivo de esta práctica, es poner en práctica los conceptos revisados durante las sesiones de laboratorio sobre el asistente de pruebas `Coq` y del lenguaje funcional contenido en este (`Gallina`), así como el que los alumnos refuercen el entendimiento de estos y puedan realizar programas complejos utilizando este lenguaje.

3. Justificación

El motivo de ser de esta práctica, es el hecho de que como se mencionó anteriormente, `Coq` será una herramienta de trabajo durante el curso, por lo que es de gran importancia que los alumnos adquieran un nivel de entendimiento sobre sus conceptos para el poder desarrollar las prácticas posteriores de manera correcta.

Es importante recalcar, este no es un curso de `Coq`, este será nuestra herramienta para poner en práctica conceptos de lógica computacional. Además de que los asistentes de prueba, son objeto de estudio de la lógica computacional.

4. Desarrollo de la práctica

4.1. Instalación de CoqIDE

Para trabajar con `Coq`, debemos realizar la instalación de `Coq` en sí, además de ser altamente recomendado la utilización de un IDE. Para esto, utilizaremos `CoqIDE`, el cual se distribuye junto con Coq, y nos permite utilizar este en modo interactivo.

Para el caso de Windows y MacOS existen archivos binarios que se pueden descargar en `https://coq.inria.fr/download`. En el caso de Linux se utiliza `Snap`, los detalles de la instalación se encuentran en `https://snapcraft.io/coq-prover`.

4.2. Programación funcional en `Coq`

Props

Recordemos que `Coq` es un lenguaje tipado, lo que significa que cada expresión tiene un tipo asociado. Las afirmaciones lógicas no son una excepción: cualquier afirmación que intentemos demostrar en Coq tiene un tipo, Prop, el tipo de las proposiciones. Podemos ver esto con el comando `Check`: `Check (3 = 3) : Prop.`

Teniendo en cuenta que `Coq`, nos brinda ya un tipo que nos permite trabajar con fórmulas proposicionales, y este además soporta conectivos proposicionales (conjunción, disyunción y negación), utilizaremos este para comenzar a trabajar con fórmulas proposicionales, por lo que es importante comprender su funcionamiento y utilización.

Para empezar a poner en práctica nuestra, nuestras habilidades realizaremos los siguientes ejercicios:

1. (1 punto) `Fixpoint In {A : Type} (x : A) (l : list A) : Prop`
   Dada una lista de tipo 'A' y un elemento del mismo tipo, construye una fórmula proposicional que nos permita saber si el elemento está contenido en la lista. **Hint:** debe ser una fórmula de 'or´s' anidados.

2. (1 punto) `Fixpoint All {T : Type} (P : T → Prop) (l : list T) : Prop`
   Las funciones que devuelven proposiciones pueden verse como propiedades de sus argumentos. Por ejemplo, si P tiene tipo `nat → Prop`, entonces P n afirma que la propiedad P es válida

para n. Implementa `All` que afirma que una propiedad P se cumple para todos los elementos de una lista l.

Streams

En la búsqueda de la representación de del tiempo, consideraremos una estructura infinita, en la cual, el tiempo tendrá una naturaleza secuencial, donde cada momento tiene exactamente un posible futuro. Esta estructura, puede ser vista como una línea infinita. Además, esta estructura de tiempo la podemos describir con las siguientes propiedades.

- El tiempo es discreto.
- El tiempo tiene un momento inicial (sin predecesores).
- Cada momento, tiene exactamente un sucesor.

Con esto en mente, implementaremos en `Gallina` el tipo `TStream`, el cual poseerá todas las propiedades descritas anteriormente. La definición en `Coq` de un TStream utilizando `Gallina` es la siguiente:

```
 Variable L : Type
Inductive TStream : Type := TScons : L ->TStream ->TStream
```

Para el tipo `TStream`, realizar los siguientes ejercicios:

1. (.5 puntos)
   `Definition TShead (s : TStream) : TStream`
   `Definition TStail (s : TStream) : TStream`
   Implementar los 'destructores' `TShead` y `TStail`, que obtienen la cabeza y cola respectivamente, para tipo `TStream`.

2. (.5 puntos) `Fixpoint TSnth (n : nat) (s : TStream) : L`
   Dado un natural 'n' y un TStream 's' regresa el n-esimo elemento de s.
3. (.5 puntos) `Fixpoint TSnth_tail (n : nat) (s : TStream) : TStream`
   Aplica 'n' veces `TStail` a 's'.
4. (1 punto) `Fixpoint TSnth_conc (n : nat) (s1, s2 : TStream) : TStream`
   Concatena los primeros 'n' elementos del TStream 's1' con 's2'

4.3. Demostraciones asistidas con `Coq`

Props

Retomaremos las funciones que realizamos en la sección anterior, sobre el tipo `Prop`, demostrando las siguientes propiedades sobre ellas.

1. (.5 puntos) `Example In_example_1 : In 4 [1; 2; 3; 4; 5].`
2. (1 punto)
   ```
   Lemma In_map :
   forall (A B : Type) (f : A → B) (l : list A) (x : A),
   ```

```
In x l →
In (f x) (map f l).
```

3. (1 punto)
```
Lemma All_In :
forall T (P : T → Prop) (l : list T),
(forall x, In x l → P x) ↔
All P l.
```

Streams

Retomando los `TStream` definidos en la sección anterior de esta práctica. Demostrar utilizando `Coq`, lo siguiente:

1. (.5 puntos)
```
Lemma one_step_nth_tail :
forall (s:TStream)(n:nat) ,
(TStail (TSnth_tail n s)) = (TSnth_tail (S n) s).
```

2. (.5 puntos)
```
Lemma multi_step_nth_tail :
forall (s:TStream)(n:nat) ,
(TSnth_tail (S n) s) = (TSnth_tail n (TStail s).
```

3. (.5 puntos)
```
Lemma multi_step_nth_conc :
forall (n:nat)(s1 s2:TStream) ,
(TSnth_conc (S n) s1 s2) =
(TSnth_conc n s1 (TScons (TSnth n s1) s2)).
```

4. (.5 puntos)
```
Lemma cons_head_tail :
forall (s : TStream) ,
TScons (TShead s) (TStail s) = s
```

5. (.5 puntos)
```
Lemma nth_tail_with_nth_conc :
forall (n:nat)(s1 s2:TStream) ,
(TSnth_tail n (TSnth_conc n s1 s2)) = s2.
```

6. (.5 puntos)
```
Lemma nth_conc_with_nth_tail :
forall (n:nat)(s:TStream) ,
(TSnth_conc n s (TSnth_tail n s)) = s.
```

5. Especificaciones de entrega

- **GitHub Classroom:** Para realizar la entrega de la práctica se utilizará la plataforma de GitHub Classroom. `https://classroom.github.com/a/p5YWIom3`

- **Equipos:** La práctica puede realizarse en equipos de hasta tres personas. Esto siempre y cuando se respete la política de trabajo colaborativo y podamos ver que ambos miembros del equipo contribuyen en la elaboración de la práctica. En caso contrario, las prácticas serán exclusivamente individuales.
- **Fecha de entrega:** La fecha de entrega será la indicada al principio de este documento. No se recibirá ninguna práctica en fechas posteriores a la fecha indicada, al menos que el profesor indique una prórroga.
- **Flujo de trabajo:** Se recomienda llevar un flujo de trabajo adecuado, en el cual se realicen commits constantemente y no hacer todo al final y en un solo commit. Esto ya que de esta manera podemos comprobar la colaboración de todos los miembros del equipo, dar feedback a lo largo del tiempo de trabajo para mejorar la entrega de las prácticas, facilitar la resolución de dudas, etc.
- **Evaluación:** Para la evaluación la práctica se someterá a un conjunto de pruebas, y además se realizará una revisión del código. Comprobándose que se cumplan las condiciones indicadas para cada ejercicio.
- **Compilado:** Cualquier práctica que al ser descargada no compile, **será evaluada con una calificación de 0**.
- **Datos personales:** Se debe agregar un documento `README.md` en el directorio raíz de sus repositorios, con sus datos personales. En caso de no ser indicados, la calificación no podrá ser asignada.
- **Limpieza y estructura:** en caso de no tener una estructura clara y limpieza dentro de sus repositorios, la calificación se verá afectada negativamente.

6. Sugerencias y Notas

- **Dudas**: Pueden preguntar sus dudas a través de telegram, o correo por el canal que ustedes deseen. Pero les recomendamos preguntar a través del grupo de telegram para que sus demás compañeros también puedan aclarar dudas similares a la suya.
- **Limitaciones**: Pueden utilizar funciones predefinidas siempre y cuando su utilización no derive en que se pierda el objetivo académico del ejercicio que se está realizando. Si se tiene duda acerca del poder utilizar algo, pueden preguntarlo.

Buena suerte a todos! ☺☺☺