Formalización y resolución de problemas

Laboratorio 4
Lógica para Computación
Primavera 2024

El objetivo de este laboratorio es poner en práctica una de las posibles aplicaciones de la lógica formal, específicamente en la resolución de problemas. A grandes rasgos, la idea es formalizar problemas enunciados en el lenguaje natural para hallar soluciones de forma automática mediante herramientas de decisión de satisfacibilidad. En el primer ejercicio vamos a poder aprovechar la implementación realizada en el Laboratorio 3 basada en el Método de Tableaux. Para los otros ejercicios, la complejidad computacional del problema de decisión de satisfacibilidad nos condiciona a usar una herramienta de porte industrial que está basada en otro algoritmo que viene siendo optimizado desde hace décadas. En esos casos vamos a usar el *SMT solver* CVC5 que es de código abierto y ofrece un entorno de ejecución online accesible a través del navegador web [1].

Ejercicios

1. (**Veraces y mentirosos**). En una extraña isla, viven tan sólo dos tipos de habitantes: los *caballeros*, que siempre dicen la verdad, y los *escuderos*, que siempre mienten. Además, los forasteros no pueden distinguir el tipo de los habitantes a simple vista.
   En una ocasión, usted visita la isla y se encuentra con tres habitantes A, B y C, entablando un diálogo. Hay distintas versiones de como fue exactamente dicho diálogo, las cuales se presentan a continuación. En cualquier caso, el objetivo es determinar, cuándo sea posible, el tipo de los tres habitantes.

   1.
   A: Todos somos escuderos.
   B: Solo uno de nosotros es caballero.

   2.
   A: B es caballero si y solo si yo soy escudero
   B: C es escudero.

   3.
   A: Si soy caballero entonces B es escudero.
   B: C es caballero si y solo si yo soy escudero.
   C: No les creas, ambos mienten.

[1] Ver Teorías de satisfacibilidad módulo.

Deberá completar su formalización en el archivo `Lab4.hs` de la siguiente manera:

- Declarar las variables que se usarán en la formalización, incluyendo un comentario de como se interpretan.
- Completar la formalización como una fórmula de tipo `L`.
- Completar el comentario con la respuesta al problema, según el resultado que se obtiene usando las funciones implementadas en el Laboratorio 3.

Sugerencia. En general, pensar:

- ¿Bajo qué condiciones, la declaración de un habitante es verdadera?.
- ¿Bajo qué condiciones, la declaración de un habitante es falsa?.
- ¿Cómo podemos expresar dichas condiciones formalmente?

2. (**Crimen de tía Agatha**). En un horrible crimen, tía Agatha fue encontrada muerta en su mansión donde vivía junto a su mayordomo y Charles. Luego de una investigación preliminar, el detective a cargo de la investigación está seguro de que alguien en la casa es el responsable y tiene la siguiente información en su pesquisa:

   P1. *Agatha fue asesinada por uno de los habitantes de la mansión.*

   P2. *El asesino siempre odia a su víctima.*

   P3. *El asesino nunca es más rico que su víctima.*

   P4. *Charles no odia a las personas que son odiadas por tía Agatha.*

   P5. *Tía Agatha odia a todos excepto a su mayordomo.*

   P6. *El mayordomo odia a todo aquel que tía Agatha odia o que no sea más rico que tía Agatha.*

   P7. *Nadie odia al mayordomo.*

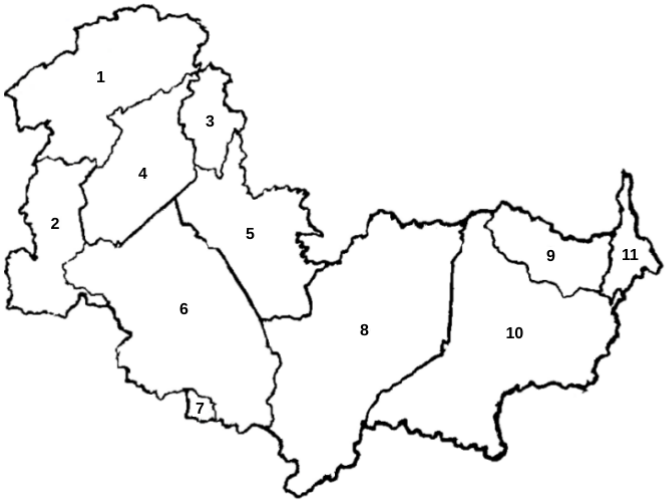
   En este ejercicio, se pide lo siguiente:

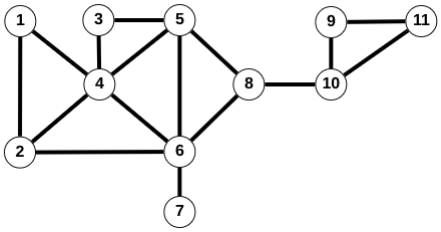
   1. Formalizar el problema y determinar quién es el culpable de la muerte de tía Agatha usando CVC5. Deberá completar su formalización en el archivo `Lab4.hs` y en el script `agatha.smt`.

   2. Confirmar de forma deductiva su hallazgo de la parte anterior. Puede armar una derivación formal en Deducción Natural o alternativamente explicarlo en palabras siempre y cuando quede bien claro de qué premisas se desprende cada una de sus inferencias.

   Nota. El problema tiene solución única.

3. **(Coloreo de mapas).** Considere un mapa con n zonas (países, estados, departamentos o etc.). Este se puede representar mediante una estructura formada por vértices (las zonas) conectados por aristas (las fronteras), lo que se denomina un *grafo*. Podemos pensar en un grafo como una abstracción de un mapa, se pierde algo de información geográfica pero se retiene información sobre las conexiones entre las zonas. Para muchas aplicaciones, como el coloreo, esa información es suficiente. Si identificamos las zonas por naturales entre 1 y n, el grafo del mapa se puede codificar mediante una relación binaria de tipo $[(\mathsf{Nat}, \mathsf{Nat})]$ donde cada pareja (i, j) nos dice que las zonas i y j son limítrofes (o adyacentes) [2].
Por ejemplo, la siguiente figura ilustra un mapa político ficticio con zonas numeradas (izquierda), su representación como grafo (arriba derecha) y su representación como relación binaria (abajo derecha).





$[(1,2),(1,4),(2,4),(2,6),$
$(3,4),(3,5),(4,5),(4,6),$
$(5,6),(5,8),(6,7),(6,8),$
$(8,10),(9,10),(9,11),(10,11)]$

El objetivo de un coloreo es incrementar la legibilidad del mapa[3]. Una *m-coloración* de un mapa G es una asignación de m colores sobre cada zona de G de tal manera que ningún par de regiones limítrofes tiene el mismo color[4]. Decimos que G es *m-coloreable* si existe una *m-coloración* para G. Además, a la mínima cantidad de colores m necesaria para hacer una coloración de G le denominamos el *número cromático* de G denotado por $\chi(G)$. Por ejemplo, para el mapa anterior tenemos $\chi = 3$, y una 3-*coloración* posible del mismo es:

$$1 \mapsto 1,\ 2 \mapsto 2,\ 3 \mapsto 1,\ 4 \mapsto 3,\ 5 \mapsto 2,\ 6 \mapsto 1,$$
$$7 \mapsto 3,\ 8 \mapsto 3,\ 9 \mapsto 1,\ 10 \mapsto 2,\ 11 \mapsto 3.$$

Es decir, podemos comenzar coloreando la zona 1 con el color 1, la zona 2 con el color 2, la zona 3 usando de nuevo el color 1 y así sucesivamente hasta colorear la zona 11. Notar que la solución no será necesariamente única.

[2] Siendo más precisos, dos zonas son limítrofes si tienen alguna frontera en común, pero un solo punto en común no se considera frontera para los propósitos de hacer un coloreo. Adicionalmente, la relación binaria se asume implícitamente simétrica.

[3] Históricamente, el interés por los coloreos surge en 1852. Se conjeturaba que dado cualquier mapa geográfico con regiones continuas, eran suficientes cuatro colores para poder colorearlo. Esto fue probado como Teorema recién en 1976, y se trata del primer Teorema matemático cuya verificación fue asistida por computadora.

[4] Formalmente, una función con dominio en zonas y rango en colores, ambos identificados por naturales.

El problema, en general, consiste en determinar si un mapa G con n zonas es *m-coloreable*. Si podemos resolver ese problema, entonces también podemos resolver el problema de determinar $\chi(G)$ viéndolo como un problema de búsqueda: hay que encontrar la *m-coloración* más pequeña posible de manera que $m = \chi(G)$. Para esto es suficiente iterar sobre un intervalo apropiado[5] de valores de m y verificar si G es *m-coloreable*.

En este ejercicio, se pide lo siguiente:

1. Formalizar el problema de coloreo de la manera más general posible, es decir para cualquier mapa con n y m arbitrarios. La respuesta quedará expresada mediante la función de Haskell:

   coloreo :: Mapa → Nat → Nat → L

2. Encontrar dos *m-coloraciones* tales que $m = \chi$ para el siguiente mapa de América Latina (LATAM) usando la parte 3.1 y CVC5.

Condiciones:

- Las formalizaciones que produzcan las soluciones solicitadas serán entregadas en los scripts `latam1.smt` y `latam2.smt`. Además, deberá marcar las soluciones encontradas sobre las imágenes `latam1.png` y `latam2.png` respectivamente.
- La codificación deberá enumerar los paises en orden alfabético, o sea Argentina = 1, Bolivia = 2, etc.

[5]Notar que m tiene trivialmente cota inferior 1, ya que se requiere al menos un color para colorear un mapa (no vacío). Por otro lado, el Teorema de Brooks en teoría de grafos nos dice que $\chi(G)$ está acotado superiormente por $\Delta(G) + 1$ donde $\Delta(G)$ es el máximo grado de G, asumiendo G es conexo. Además, por el Teorema de los cuatro colores, si G es simple y plano, $\chi(G)$ no es mayor a 4.

3. La aerolínea UruguayTieneAlas se encuentra planificando los horarios de los distintas vuelos que se ofrecerán periódicamente. Por diversos motivos, no todas los vuelos se pueden ofrecer en el mismo horario. Si un par de vuelos no se pueden ofrecer en el mismo horario (quizá por falta de aviones, o por indisponibilidad de personal de vuelo, etc.) decimos que tienen *conflicto*.

   Cada vuelo está identificado por un código. La siguiente tabla resume los vuelos a considerar y sus conflictos:

Vuelo (Origen-Destino)	**Código**	**Conflicto con**
MVD-POA	1	2, 3, 7, 9
POA-BSB	2	1, 4, 5
MAE-MVD	3	1, 4, 6, 10
EZE-MVD	4	2, 3, 5, 7
EZE-POA	5	2, 4, 8, 9
EZE-AEP	6	3, 7
CBA-AEP	7	1, 4, 6
CBA-SCL	8	5, 9
SCL-MVD	9	1, 5, 8
STY-MVD	10	3

   Por razones de organización, la aerolínea requiere determinar la cantidad mínima de horarios de vuelo necesarios para acomodar todas los vuelos de tal manera que no haya conflicto.

   En esta parte se pide:

   a. Explicar como modelar el problema de planificación como un problema de coloreo.

   b. Encontrar una solución al problema usando la parte 3.1 y CVC5. La formalización que produzca la solución solicitada será entregada en el script `plan.smt`.

   c. Mostrar como queda la asignación de vuelos, según su solución, en el siguiente formato:

      Horario 1: Código vuelo A, Código vuelo B, ...

      Horario 2: Código vuelo C, Código vuelo D, ...

      ⋮

Sugerencia. Para el ejercicio 3 se recomienda implementar las siguientes funciones de soporte que le permitirán construir en Haskell fórmulas de LP con tamaño arbitrario. Además, estas se apoyan en el trabajo realizado en el Laboratorio 3, por lo cual se asume que el módulo de dicho laboratorio se encuentra disponible en el mismo directorio que el laboratorio actual.

- **bigAnd :: [Nat] → (Nat → L) → L**
  Dada una lista de índices I y una fórmula indexada α, devuelve la conjuntaria de α sobre I, o sea $\bigwedge_{i \in I} \alpha_i$.
  Ejemplo: Para representar $\bigwedge_{i \in [1,2,3]} p_i$ en Haskell escribimos

  $$\mathsf{bigAnd}\ [1,2,3]\ (\lambda i.\, \mathsf{V}\ (\text{“p”}\mathsf{++}\ \mathsf{show\ i})) \ =\ (\mathsf{V}\ \text{“}p1\text{”})\ \mathsf{And}\ (\mathsf{V}\ \text{“}p2\text{”})\ \mathsf{And}\ (\mathsf{V}\ \text{“}p3\text{”})$$

- **bigOr :: [Nat] → (Nat → L) → L**
  Dada una lista de índices I y una fórmula indexada α, devuelve la disyuntoria de α sobre I, o sea $\bigvee_{i \in I} \alpha_i$.
  Ejemplo: Para representar $\bigvee_{i \in [1,2,3]} p_i$ en Haskell escribimos

  $$\mathsf{bigOr}\ [1,2,3]\ (\lambda i.\, \mathsf{V}\ (\text{“p”}\mathsf{++}\ \mathsf{show\ i})) \ =\ (\mathsf{V}\ \text{“}p1\text{”})\ \mathsf{Or}\ (\mathsf{V}\ \text{“}p2\text{”})\ \mathsf{Or}\ (\mathsf{V}\ \text{“}p3\text{”})$$

- **v2 :: Var → Nat → Nat → L**
  Dada una variable p y dos índices i y j, devuelve la variable p indexada en i y j. Los índices deberán quedar separados por una barra baja.
  Ejemplo: Para representar la variable $p_{1,2}$ en Haskell, escribimos $\mathsf{v2}\ \text{“}p\text{”}\ 1\ 2$ que produce el valor $\mathsf{V}\ \text{“}p1\_2\text{”}$.

Ilustramos con un ejemplo como se pueden combinar las funciones anteriores. Para cualquier par de índices $I = [1..n]$ y $J = [1..m]$, podemos por ejemplo representar en Haskell la siguiente conjunción de disyunciones de literales indexados (una FNC)

$$\bigwedge_{i \in I} \bigvee_{j \in J} p_{i,j}$$

escribiendo

$$\mathsf{bigAnd}\ [1..n]\ (\lambda i.\ \mathsf{bigOr}\ [1..m]\ (\lambda j.\ \mathsf{v2}\ \text{“}p\text{”}\ i\ j))$$

En particular, si $n = 2$ y $m = 2$, esto construye en Haskell la fórmula de LP

$$\mathsf{Bin}\ (\mathsf{Bin}\ (\mathsf{V}\ \text{“}p1\_1\text{”})\ \mathsf{Or}\ (\mathsf{V}\ \text{“}p1\_2\text{”}))\ \mathsf{And}\ (\mathsf{Bin}\ (\mathsf{V}\ \text{“}p2\_1\text{”})\ \mathsf{Or}\ (\mathsf{V}\ \text{“}p2\_2\text{”}))$$

que en la sintaxis del formato SMT-LIB (requerido para trabajar con CVC5) se escribe con notación prefija y parentizada

$$(\mathsf{and}\ (\mathsf{or}\ p1\_1\ p1\_2)\ (\mathsf{or}\ p2\_1\ p2\_2))$$

y en nuestra sintaxis concreta (“notación de pizarra”) se escribe

$$(p_{1,1} \vee p_{1,2}) \wedge (p_{2,1} \vee p_{2,2})$$

Afortunadamente, nunca vamos a necesitar escribir a mano en la sintaxis del formato SMT-LIB, porque disponemos de la función auxiliar toPrefix que convierte a dicha sintaxis. De esta manera, Haskell nos servirá como interfaz para usar CVC5.