

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2213 — Lógica para Ciencia de la Computación — 2024-01

TAREA 5

Publicación: Jueves 13 de junio.
Entrega: **Viernes 28 de junio hasta las 23:59 horas**.

Indicaciones

- La tarea es **individual**. Puede discutir con sus compañera/os o utilizar información externa (libros, foros, páginas web, etc...), siempre y cuando usted mismo escriba su tarea, con sus propias palabras. Debe **citar** cualquier referencia que utilice. Se considerará copia: usar soluciones de compañera/os (con o sin consentimiento del plagiado), entregar su solución para que la vea otra/o compañera/o, usar fuentes externas sin citar.
- Cada pregunta tiene una nota entre 1 y 7 (hay un punto base). Los items dentro de una pregunta valen lo mismo. La nota de la tarea es el promedio de las notas de las preguntas.
- La tarea **debe** ser escrita en LaTeX. No se aceptarán tareas escritas de otra forma. No olvide indicar su nombre, RUT y número de alumna/o.
- **Ojo:** No se admiten atrasos en esta tarea.

Pregunta 1

Sea $\mathcal{L}$ un vocabulario *relacional*, es decir, un vocabulario compuesto sólo por relaciones. Una *consulta conjuntiva* sobre $\mathcal{L}$ es una $\mathcal{L}$-fórmula $\varphi(x_1, \ldots, x_n)$ de la forma:

$$\varphi(x_1, \ldots, x_n) = \exists z_1 \cdots \exists z_\ell \bigwedge_{1 \leq i \leq m} R_i(\bar{y}_i)$$

donde, para cada $1 \leq i \leq m$, R_i es una relación en $\mathcal{L}$, e $\bar{y}_i$ es una tupla de variables en $\{x_1, \ldots, x_n\} \cup \{z_1, \ldots, z_\ell\}$ cuyo largo es la aridad de R_i.
Recordar que $x_1, \ldots, x_n$ son las variable libres de φ. En caso de que no hayan variables libres, es decir, φ sea una oración, decimos que la consulta conjuntiva es *Booleana*.

Considere el vocabulario $\mathcal{L} = \{R, S, T\}$, donde R y S son relaciones binarias, y T es una relación ternaria. Algunos ejemplos de consultas conjuntivas sobre $\mathcal{L}$:

- $\varphi(x_1, x_2) = \exists z \big(R(x_1, z) \wedge S(z, x_2)\big)$.
- $\varphi(x) = \exists z_1 \exists z_2 \big(T(x, z_1, z_2) \wedge T(z_1, x, z_2) \wedge T(z_1, z_2, x) \wedge S(z_1, z_2)\big)$.
- $\varphi(x_1, x_2, x_3) = T(x_1, x_2, x_3) \wedge R(x_1, x_3)$. (en este caso no hay cuantificación existencial.)
- $\varphi = \exists z_1 \exists z_2 \exists z_3 \big(R(z_1, z_2) \wedge S(z_2, z_3) \wedge T(z_1, z_2, z_3)\big)$. (en este caso la consulta es Booleana.)

Demuestre que toda consulta conjuntiva es satisfacible.

Solución

Sea φ consulta conjuntiva sobre un vocabulario $\mathcal{L} = \{R_i, \ldots, R_m\}$ tal que r_i es la aridad de R_i para todo $i \in \{1, \ldots, m\}$. Entonces definimos la estructura

$$\mathfrak{A}_\varphi = \langle A = \{\star\}, A^{r_1}, \ldots, A^{r_m} \rangle$$

Como φ es consulta conjuntiva, sabemos que es de la forma

$$\varphi(x_1, \ldots, x_n) = \exists z_1 \cdots \exists z_\ell \bigwedge_{1 \leq i \leq m} R_i(\bar{y}_i)$$

Entonces podemos tomar $x_i = \star$ y $z_j = \star$ para todo $i \in \{1, \ldots, n\}$ y $j \in \{1, \ldots, \ell\}$ y obtenemos que cada $\bar{y} = (\star)_{i=1}^{r_i} \in R_i$, es decir $R_i(\bar{y}_i)$ es verdadera para todo $i \in \{1, \ldots, m\}$, por lo que se tiene que $\mathfrak{A}_\varphi \models \varphi$.
Al haber construido una estructura que satisface cada consulta conjuntiva, se puede concluir que toda consulta conjuntiva es satisfacible.

La pregunta tiene un total de 6 puntos, de ellos 1 es por entender la definición de satisfacibilidad, 2 por construir la estructura correctamente y 3 por justificar adecuadamente que la oración se satisface en la estructura construida. (La estructura podría ser distinta a la presentada en la pauta)

Pregunta 2

Considere el siguiente problema

$$\text{EVAL-CC} = \{(\mathcal{L}, \varphi, \mathfrak{A}) \mid \mathcal{L} \text{ es vocabulario relacional, } \varphi \text{ es consulta conjuntiva Booleana sobre } \mathcal{L}, \\ \mathfrak{A} \text{ es } \mathcal{L}\text{-estructura con dominio finito, y } \mathfrak{A} \models \varphi\}.$$

Demuestre que EVAL-CC es NP-completo.

Solución

Para probar NP completitud, hay que probar EVAL-CC $\in$ NP y que es NP-hard.

EVAL-CC $\in$ NP

Para probar que EVAL-CC está en NP, se presentará un algoritmo no determinista que lo acepta en tiempo polinomial. Para esto, sea $w = (\mathcal{L}, \varphi, \mathfrak{A})$, omitiremos el chequeo de formato, ya que eso es un paso claramente polinomial. Como φ es consulta conjuntiva, debe ser de la siguiente manera

$$\varphi = \exists z_1 \cdots \exists z_\ell \bigwedge_{1 \leq i \leq m} R_i(\bar{y}_i)$$

donde, para cada $1 \leq i \leq m$, R_i es una relación en $\mathcal{L}$, e $\bar{y}_i$ es una tupla de variables en $\{z_1, \ldots, z_\ell\}$ cuyo largo es la aridad de R_i. Llamemos ψ a $\bigwedge_{1 \leq i \leq m} R_i(\bar{y}_i)$, luego, la máquina trabajará de la siguiente manera

1. Para cada z_i se elige algún elemento del dominio de $\mathfrak{A}$ y se reemplaza en la fórmula ψ.
2. Se revisa si cada $\bar{y}_i$ se encuentra en la relación R_i, si es así, se acepta. Si para algún i se tiene $\bar{y}_i \notin R_i$, entonces se rechaza.

Notemos que si $\mathfrak{A} \models \varphi$, entonces existe una elección de z_i's que hacen verdadera a ψ, por lo tanto, w sería aceptada por la máquina. Por otro lado, si $\mathfrak{A} \not\models \varphi$, para toda elección de los z_i's, existirá un j tal que $\bar{y}_j$ no está en R_j, es decir, w no será aceptada.
Además, este algoritmo funciona en tiempo polinomial, ya que ambos pasos requieren tiempo lineal, entonces al concatenarlos, tenemos un algoritmo polinomial.

EVAL-CC es NP-hard

Para probar esto se presentará una reducción desde 3-COL. Definimos la reducción $f : \{0,1\}^* \to \{0,1\}^*$ de la siguiente manera:

1. Si z no tiene la forma $z = (V, E)$ entonces $f(z) = 0 \notin$ EVAL-CC.
2. Si $z = (V, E)$, entonces definimos $\mathcal{L} = \{R\}$ con R relación binaria,

   $$\mathfrak{A} = \langle \{\bullet, \bullet, \bullet\}, \{(\bullet, \bullet), (\bullet, \bullet), (\bullet, \bullet), (\bullet, \bullet), (\bullet, \bullet), (\bullet, \bullet)\} \rangle$$

   y si n es el número de vértices, definimos la oración

   $$\varphi = \exists c_1 \ldots \exists c_n \bigwedge_{\{i,j\} \in E} R(c_i, c_j)$$

   Por lo tanto, tomamos $f(z) = (\mathcal{L}, \varphi, \mathfrak{A})$

Esto es computable en tiempo polinomial, ya que $\mathcal{L}$ y $\mathfrak{A}$ son fijos y para escribir φ solo hay que agregar una variable cuantificada por cada vértice y una relación por cada arista.

Probemos que esta función efectivamente es una reducción, es decir, $z \in$ 3-COL si y solo si $f(z) \in$ EVAL-CC.

($\Rightarrow$) Sea $z \in$ 3-COL, entonces $z = (V, E)$ con (V, E) un grafo 3 coloreable. Entonces existe una coloración $c : V \to \{\bullet, \bullet, \bullet\}$. Ahora podemos definir $c_i = c(i)$ y esto satisfacerá φ, ya que para toda arista $\{i, j\}$ tenemos que $c(i) \neq c(j)$, es decir, $(c_i, c_j) \in R$. Por lo tanto $\mathfrak{A} \models \varphi$ y $f(z) \in$ EVAL-CC.

($\Leftarrow$) Sea $f(z) \in$ EVAL-CC, entonces $f(z) = (\mathcal{L}, \varphi, \mathfrak{A})$ como definimos anteriormente y $\mathfrak{A} \models \varphi$. Es decir, existe una elección de c_i's tal que se cumple $\bigwedge_{\{i,j\} \in E} R(c_i, c_j)$, entonces podemos definir la función $c : V \to \{\bullet, \bullet, \bullet\}$ tal que $c(i) = c_i$. Notemos que por φ tenemos que $c(i) \neq c(j)$ para todo $\{i, j\} \in E$, es decir, la función definida es una 3-coloración. Concluimos así que el grafo es 3-coloreable y $z \in$ 3-COL.

Como sabemos que 3-COL es NP-hard y construimos una reducción a EVAL-CC, podemos construir que EVAL-CC es NP-hard y como además tenemos que está en NP se concluye que EVAL-CC es NP-completo.

En esta pregunta se dan 2 puntos por probar que el lenguaje está en NP (puede ser con cualquiera de las dos definiciones vistas en clases) y 4 puntos por probar que es NP-hard. De los 4 puntos, se dan 2 por construir la reducción y 1 por cada lado de la doble implicancia. (La reducción se podría hacer desde otro lenguaje, pero la distribución de puntaje sería la misma)

Pregunta 3

Considere el vocabulario $\mathcal{L} = \{E\}$ para grafos dirigidos, donde E es una relación binaria. Decimos que una $\mathcal{L}$-estructura $G = (V, E^G)$ contiene un ciclo finito si existen elementos $u_1, \ldots, u_n \in V$, con $n \geq 2$, tal que para todo $1 \leq i < n$, se tiene que $(u_i, u_{i+1}) \in E^G$, y además $(u_n, u_1) \in E^G$.

Aplicando el Teorema de Compacidad, demuestre que no existe una $\mathcal{L}$-oración φ tal que para toda $\mathcal{L}$-estructura G, se cumple:

$$G \text{ contiene un ciclo finito si y sólo si } G \models \varphi.$$

Solución

En primer lugar definimos las oraciones ψ_n de la siguiente manera,

$$\psi_n = \exists u_1 \ldots \exists u_n \; E(u_n, u_1) \wedge \bigwedge_{i=1}^{n-1} E(u_i, u_{i+1})$$

Luego, supongamos que existe una oración φ tal que $G \models \varphi$ si y solo si G contiene un ciclo finito. Entonces podemos definir el conjunto de oraciones

$$\Sigma = \{\varphi\} \cup \{\neg\psi_i : i \in \mathbb{N}\}$$

Sea $\Sigma' \subset \Sigma$ finito, entonces existe un $n \in \mathbb{N}$ máximo tal que $\neg\psi_n \in \Sigma'$, luego tomamos $G = (V, E^G)$ como el grafo que consiste en un $n+1$ ciclo, es decir $V = \{u_1, \ldots, u_{n+1}\}$ y

$$E^G = \{(u_n, u_1), (u_1, u_n)\} \cup \{(u_i, u_{i+1}) : 1 \leq i \leq n-1\} \cup \{(u_{i+1}, u_i) : 1 \leq i \leq n-1\}$$

Por lo tanto, G no tiene ciclos de largo menor o igual a n (ie. $G \models \neg\psi_i$ para todo $i \leq n$) y $G \models \varphi$, ya que contiene un ciclo de largo $n+1$. En particular, como sabemos que $\Sigma' \subseteq \{\varphi\} \cup \{\psi_i : i \leq n\}$ tenemos que $G \models \Sigma'$.
Por Teorema de Compacidad, podemos concluir que Σ es satisfacible, por lo tanto existe un grafo $G = (V, E^G)$ tal que no tiene un ciclo de largo n para todo $n \in \mathbb{N}$ y, sin embargo, tiene un ciclo finito, llegando así a una contradicción.
En conclusión, no puede existir la oración φ tal que $G \models \varphi$ si y solo si G contiene un ciclo finito.

En esta pregunta se da 2 puntos por construir la familia de oraciones Σ, 3 puntos por probar que Σ es finitamente satisfacible y 1 punto por aplicar compacidad correctamente para concluir lo pedido.