Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC1253 - Matemáticas Discretas

Tarea 2

15 de abril de 2024

1º semestre 2024 - Profesores P. Bahamondes - S. Bugedo - N. Alvarado

---

Requisitos

- La tarea es individual. Los casos de copia serán sancionados con la reprobación del curso con nota 1,1.
- **Entrega**: Hasta las 23:59 del 20 de marzo a través del buzón habilitado en el sitio del curso (Canvas).
  - Esta tarea debe ser hecha completamente en LaTeX. Tareas hechas a mano o en otro procesador de texto **no serán corregidas**.
  - Debe usar el template LaTeX publicado en la página del curso.
  - Cada solución de cada problema debe comenzar en una nueva hoja. ***Hint:*** Utilice `\newpage`
  - Los archivos que debe entregar son el archivo `PDF` correspondiente a su solución con nombre `numalumno.pdf`, junto con un `zip` con nombre `numalumno.zip`, conteniendo el archivo `numalumno.tex` que compila su tarea. Si su código hace referencia a otros archivos, debe incluirlos también.
- El no cumplimiento de alguna de las reglas se penalizará con un descuento de 0.5 en la nota final (acumulables).
- No se aceptarán tareas atrasadas.
- Si tiene alguna duda, el foro de Github (issues) es el lugar oficial para realizarla.

Problemas

Problema 1

Se define el conjunto de expresiones algebraicas sobre los naturales, denotado por S, como el menor conjunto que satisface las siguientes condiciones:

- Si n es natural, entonces $n \in S$.
- Si $a, b \in S$, entonces $(a + b)$, $(a * b)$, $(a - b)$ y (a/b) son expresiones algebraicas sobre los naturales.

Por ejemplo, 5 y $((4 - (5 + 3))/6)$ son expresiones algebraicas sobre los naturales. Observe que estas expresiones se definen como secuencias de caracteres (*strings*).

(a) Demuestre por inducción estructural que si $a \in S$, entonces o bien a es un natural, o bien a empieza con "(" y termina con ")".

(b) Defina el operador largo, length : $S \to \mathbb{N}$, que cuente la cantidad de números, operaciones $(+, *, -, /)$ y paréntesis que contiene una expresión algebraica sobre los naturales. Observe que el caso base para $n \in \mathbb{N}$ cumple $\text{length}(n) = 1$.

(c) Sea $a \in S$. Definimos la profundidad de a, denotada por $\text{depth}(a)$, como:

- Si $a \in \mathbb{N}$, entonces $\text{depth}(a) = 0$
- Si $a = (a_1 \# a_2)$, para $a_1, a_2 \in S$ y $\# \in \{+, *, -, /\}$, entonces

$$\text{depth}(a) = \text{máx}\{\text{depth}(a_1), \text{depth}(a_2)\} + 1$$

Demuestre por inducción que para todo natural $n \geq 1$ se cumple que existe una expresión algebraica sobre los naturales $a \in S$ tal que

$$\text{length}(a) \geq n \text{ y } \text{depth}(a) \leq \log_2(\text{length}(a))$$

El siguiente inciso es optativo y ofrece un bonus de 1 punto a la nota de la tarea 2.

(d) Escriba un programa recursivo en `Python` para la función `is_valid_expression(s)` que, dado un string `s`, retorne un booleano que indique si `s` es una expresión algebraica sobre los naturales.

Sugerencia: puede usar las propiedades en el inciso (a) junto con las funciones `isdigit`, `startswith` y `endswith` definidas para strings en `Python` y encontrar el operador de a a menor profundidad para separar una expresión algebraica en sus subexpresiones.

Condiciones de entrega: único archivo de nombre `numero_alumno.py` tal que su solución se pruebe haciendo un llamado a la función `is_valid_expression(s)` en dicho archivo.

Ejemplos de ejecución:

```
is_valid_expression('((3-(4*((4+1)*8)))/7)')  retorna True
is_valid_expression('((3-(4*((4+1)*8))/7)')  retorna False
```

Solución

Parte (a)

Para demostrar por inducción estructural que si $a \in S$, entonces o bien a es un natural, o bien a empieza con "(ž termina con ")", consideramos los siguientes casos:

Caso base: Si a es un número natural, entonces por definición $a \in S$. En este caso, a es un natural y, por lo tanto, no necesita empezar con "("ni terminar con ")".

Paso inductivo: Suponemos que $a, b \in S$ y cumplen con la propiedad de ser naturales o de empezar con "(ž terminar con ")". Necesitamos demostrar que las expresiones $(a + b)$, $(a * b)$, $(a - b)$, y (a/b) también cumplen con la propiedad.

Para cualquier expresión formada por a y b utilizando las operaciones $+$, $*$, $-$, y $/$, la expresión resultante se encierra entre paréntesis "(ž ")". Por lo tanto, cualquier nueva expresión formada será del tipo $(a\#b)$, donde $\#$ representa una operación, y comenzará con "(ž terminará con ")".

Por lo tanto, hemos demostrado que si $a \in S$, entonces o bien a es un natural, o bien a empieza con "(ž termina con ")".

Puntaje:

- 0.2 por el caso base.
- 0.3 por plantear correctamente la hipótesis inductiva.
- 1 por demostrar la tésis de inducción.

La asignación de puntajes parciales queda a criterio del corrector.

Parte (b)

Para definir el operador largo, length : $S \to \mathbb{N}$, contamos la cantidad de números, operaciones y paréntesis en una expresión algebraica sobre los naturales de la siguiente manera:

- Si $a \in \mathbb{N}$, entonces $\text{length}(a) = 1$.
- Si $a = (a_1\#a_2)$, donde $\#$ es una operación $+, *, -, /$ y $a_1, a_2 \in S$, entonces $\text{length}(a) = \text{length}(a_1) + \text{length}(a_2) + 3$.

Caso base: Para cualquier número natural n, por definición, $\text{length}(n) = 1$ ya que n es simplemente un número sin operaciones ni paréntesis adicionales. Esto establece nuestro caso base.

Paso inductivo: Supongamos que para dos expresiones $a, b \in S$, las propiedades de *length* se mantienen, es decir, $\text{length}(a)$ y $\text{length}(b)$ cuentan correctamente la cantidad de números, operaciones y paréntesis en a y b, respectivamente.

Necesitamos demostrar que para las expresiones compuestas $(a+b)$, $(a*b)$, $(a-b)$, y (a/b), la propiedad de *length* también se mantiene.

Consideremos la expresión $(a+b)$ como ejemplo (el razonamiento es análogo para las otras operaciones). La longitud de esta expresión compuesta, $\text{length}(a+b)$, se calcula como la suma de la longitud de a, la longitud de b, más uno para la operación $+$ y dos para los paréntesis que rodean la expresión completa. Esto se puede expresar como:

$$\text{length}(a+b) = \text{length}(a) + \text{length}(b) + 3$$

Dado que asumimos por hipótesis inductiva que $\text{length}(a)$ y $\text{length}(b)$ cuentan correctamente los componentes dentro de a y b, agregando 3 por la operación y los paréntesis asegura que $\text{length}(a+b)$ también cuenta correctamente todos los componentes dentro de la nueva expresión.

Por lo tanto, la propiedad de *length* se mantiene para cualquier expresión formada por la combinación de a y b utilizando las operaciones definidas, completando así el paso inductivo.

Puntaje:

- 0.5 definir correctamente el operador lenght.

La asignación de puntajes parciales queda a criterio del corrector.

Parte (c)

Para demostrar por inducción que para todo natural $n \geq 1$ existe una expresión algebraica sobre los naturales $a \in S$ tal que $\text{length}(a) \geq n$ y $\text{depth}(a) \leq \log_2(\text{length}(a))$, consideramos la construcción de expresiones algebraicas de la forma $(a_1 \# a_2)$ con $a_1, a_2 \in S$ y demostramos que cumplen con las condiciones requeridas.

Caso base (n = 1): Para $n = 1$, consideramos cualquier número natural $a \in \mathbb{N}$. En este caso, $\text{length}(a) = 1$ y $\text{depth}(a) = 0$. Claramente, $1 \geq 1$ y $0 \leq \log_2(1) = 0$, por lo que el caso base se cumple.

Paso inductivo: Supongamos que la afirmación es verdadera para algún $k \geq 1$, es decir, existe una expresión $a_k \in S$ tal que $\text{length}(a_k) \geq k$ y $\text{depth}(a_k) \leq \log_2(\text{length}(a_k))$. Necesitamos demostrar que la afirmación es verdadera para $k+1$.

Considere la expresión $a_{k+1} = (a_k + 1)$, donde 1 es un número natural y $+$ es una operación binaria. Por la definición de length y depth, tenemos:

$$\begin{aligned}\text{length}(a_{k+1}) &= \text{length}(a_k) + \text{length}(1) + 3 \geq k + 1 + 3 > k + 1\\ \text{depth}(a_{k+1}) &= \text{depth}(a_k) + 1 \leq \log_2(\text{length}(a_k)) + 1\end{aligned}$$

Para que la desigualdad $\text{depth}(a_{k+1}) \leq \log_2(\text{length}(a_{k+1}))$ se mantenga, necesitamos demostrar que:

$$\log_2(\text{length}(a_k)) + 1 \leq \log_2(\text{length}(a_{k+1}))$$

Dado que $\text{length}(a_{k+1}) > \text{length}(a_k)$, esto implica que $\text{length}(a_{k+1})$ es al menos el doble de $\text{length}(a_k)$, lo cual satisface la desigualdad anterior.

Por lo tanto, hemos demostrado que para todo $k \geq 1$, existe una expresión algebraica a_{k+1} tal que $\text{length}(a_{k+1}) \geq k + 1$ y $\text{depth}(a_{k+1}) \leq \log_2(\text{length}(a_{k+1}))$, completando así el paso inductivo.

Puntaje:

- 0.5 Por el caso base
- 1.5 por plantear correctamente la hipótesis inductiva.
- 2 por demostrar la tésis de inducción.

La asignación de puntajes parciales queda a criterio del corrector.

Parte (d):

La función `is_valid_expression`(s) toma como entrada uns string s y retorna un valor booleano indicando si s es una expresión válida en S.

El algoritmo sigue estos pasos:

1. Si s es un número (todos los caracteres son dígitos), retorna `True`.
2. Si s no comienza con '(' o no termina con ')', retorna `False`.
3. Elimina los paréntesis externos y busca la operación central (una de $+, *, -, /$) que no esté dentro de otros paréntesis. Esto divide s en dos subexpresiones.
4. Aplica recursivamente `is_valid_expression` a las subexpresiones. Si ambas subexpresiones son válidas, retorna `True`; de lo contrario, retorna `False`.

Pseudocódigo:

```
def is_valid_expression(s):
    if s.isdigit():
        return True
```

```
    if not (s.startswith('(') and s.endswith(')')):
        return False
    s = s[1:-1]
```

Este algoritmo se basa en la estructura recursiva de las expresiones en S y utiliza la inducción estructural implícitamente para validar la expresión.

La asignación de puntajes parciales queda a criterio del corrector.

Problema 2

Considere el funcionamiento de un semáforo en instantes discretos de tiempo que llamaremos estados, tal que la cantidad de estados totales es finita.

(a) Defina un conjunto P de variables proposicionales adecuadas que permitan definir un lenguaje $\mathcal{L}(P)$ de fórmulas proposicionales para modelar este escenario. Explique brevemente el significado de cada variable definida. *Sugerencia:* examine los incisos (b), (c) y (d) para determinar qué necesita incluir en su diseño.

Con el lenguaje definido en (a), proponga una fórmula proposicional φ para cada uno de los siguientes incisos. Su fórmula debe ser satisfacible si y solo si la propiedad descrita se cumple para un semáforo dado. Explique brevemente el signficado de las partes de su fórmula. No necesita demostrar la correctitud de su fórmula.

(b) La luz del semáforo en todo estado es, o verde, o roja, o amarilla.

(c) Los únicos cambios de color de luz del semáforo ocurren entre estados sucesivos y pueden ocurrir de verde a amarilla, de amarilla a roja y de roja a verde.

(e) La luz puede tener el mismo color en, a lo más, 3 estados sucesivos.

Solución

Parte (a)

Vamos a considerar un total de T tiempos o estados, además de los tres colores típicos de un semáforo, verde, rojo y amarillo. Definimos entonces las siguientes variables proposicionales para modelar un semáforo:

v_t: 1 si el semáforo es verde en el tiempo $t \in \{1, ..., T\}$.

r_t: 1 si el semáforo es rojo en el tiempo $t \in \{1, ..., T\}$.

a_t: 1 si el semáforo es amarillo en el tiempo $t \in \{1, ..., T\}$.

Parte (b)

Lo denotamos por la siguiente fórmula, que indica lo anterior en todo tiempo

$$\varphi_1 := \bigwedge_{t=1}^{T} (v_t \vee r_t \vee a_t)$$

Parte (c)

Lo denotamos por la fórmula $\varphi_2 \wedge \varphi_3$ donde φ_2 denota que solo hay un único estado de un color en un tiempo dado, y φ_3 denota que la transición respeta la secuencia descrita, según las definimos a continuación:

$$\varphi_2 := \bigwedge_{t=1}^{T}((v_t \to (\neg r_t \wedge \neg a_t)) \wedge (r_t \to (\neg v_t \wedge \neg a_t)) \wedge (a_t \to (\neg v_t \wedge \neg r_t)))$$

$$\varphi_3 := \bigwedge_{t=1}^{T}((v_t \to (v_{t+1} \vee a_{t+1})) \wedge (r_t \to (r_{t+1} \vee v_{t+1})) \wedge (a_t \to (a_{t+1} \vee v_{t+1})))$$

Parte (d)

Lo denotamos por la fórmula φ_4 según definimos a continuación:

$$\varphi_4 := \bigwedge_{t=1}^{T}(((v_t \wedge v_{t+1} \wedge v_{t+2}) \to \neg v_{t+3}) \wedge ((r_t \wedge r_{t+1} \wedge r_{t+2}) \to \neg r_{t+3}) \wedge ((a_t \wedge a_{t+1} \wedge a_{t+2}) \to \neg a_{t+3})$$

Puntaje:

- 0.5 Parte (a)
- 0.5 Parte (b)
- 2 Parte (c)
- 1 Parte (d)

La asignación de puntajes parciales queda a criterio del corrector.