Práctica 1:
Preliminares matemáticos

Grado en Inteligencia Artificial
Autómatas y Lenguajes Formales

2025/2026

1. Construya un programa de nombre `q` que escriba en la salida estándar los n primeros elementos de $\mathbb{Q}$, el conjunto de los racionales (cuando decimos "los n primeros elementos" nos referimos a aquellos que son imagen de los naturales 0 a $n-1$, en la aplicación biyectiva más usual que se suele construir para demostrar que $\mathbb{N}$ y $\mathbb{Q}$ tienen la misma cardinalidad). Este programa debe recibir en la línea de comandos la opción `-a` (*all*) y el número n.

   Ejemplo de ejecución:

   ```
   $ python q.py -a 15
   1/1 1/2 2/1 3/1 2/2 1/3 1/4 2/3 3/2 4/1 5/1 4/2 3/3 2/4 1/5
   ```

   Apartado opcional: Revise la implementación anterior de manera que no escriba todos y cada uno de los posibles quebrados, sino sólo los que realmente sean irreducibles. Incorpore este comportamiento a través de una nueva opción `-u` (*unique elements*).

   Ejemplo de ejecución:

   ```
   $ python q.py -u 15
   1/1 1/2 2/1 3/1 1/3 1/4 2/3 3/2 4/1 5/1 1/5 1/6 2/5 3/4 4/3
   ```

   Compare los tiempos de ejecución de `q -a` y de `q -u` para valores muy altos de n (por ejemplo, $n = 10{,}000{,}000$), y comente las conclusiones que pueden ser extraídas a este respecto.

2. En matemáticas, un *sistema formal* es un modelo abstracto compuesto básicamente por símbolos primitivos, axiomas y reglas de producción (y, a veces, también reglas de inferencia y una semántica), que se utiliza para deducir teoremas y proporcionar una definición rigurosa del concepto de demostración. Cuando se crea un sistema formal, se pretende abstraer la esencia de determinadas características del mundo real en un modelo conceptual expresado en un determinado lenguaje formal. Algunos de los sistemas formales más conocidos son la lógica proposicional, la lógica de primer orden y la lógica modal.

   En el siglo XX, Hilbert y otros sostuvieron que la matemática era un sistema formal. Pero en 1931, Kurt Gödel demostró que ningún sistema formal con suficiente poder expresivo como para capturar la aritmética de Peano puede ser a la vez consistente y completo. El teorema de la incompletitud de Gödel, junto con la demostración de Alonzo Church y de Alan Turing de que la matemática y los procesos computacionales tampoco son decidibles, terminó con las premisas de Hilbert. Sin embargo, el concepto sigue siendo ampliamente utilizado, básicamente porque no se ha encontrado ninguna alternativa mejor al enfoque formalista de Hilbert y a su pretensión de trabajar siempre en el seno de teorías matemáticas explícitamente axiomatizadas, aun con sus limitaciones.

Los sistemas formales también han encontrado aplicación dentro de la informática y de la teoría de la información, y de hecho guardan una relación muy estrecha con los lenguajes formales (y con las gramáticas que los generan) que veremos a lo largo de esta asignatura.

Un ejemplo sencillo de sistema formal es el sistema `mg~` ideado por Douglas R. Hofstadter en su libro *"Gödel, Escher, Bach: un Eterno y Grácil Bucle"*:

- El sistema cuenta con tres símbolos: las letras `m` y `g` y el guión ~.
- Este sistema tiene una cantidad infinita de axiomas. Así pues, ante la imposibilidad de enunciarlos todos, y ante la necesidad de un mecanismo que indique si una determinada cadena de símbolos es o no un axioma, hacemos la siguiente definición: x`m~g`x`~` es un axioma, siempre que x esté compuesto sólo por guiones.

  Por ejemplo, la cadena ~~`m~g`~~~ es un axioma (en este caso, $x = {\sim\sim}$).
- Este sistema solamente tiene una regla de producción, que es la siguiente. Supongamos que x, y y z son cadenas formadas exclusivamente por guiones. Y supongamos que x`m`y`g`z es un teorema (es decir, una cadena perteneciente al lenguaje formal asociado a este sistema formal). Entonces, x`m`y`~g`z`~` es también un teorema.

  Por ejemplo, si ~~`m~~~g`~~~~~ es un teorema, entonces ~~`m~~~~g`~~~~~~ también lo es (en este caso, $x = {\sim\sim}$, $y = {\sim\sim\sim}$ y $z = {\sim\sim\sim\sim\sim}$).

Construya un programa de nombre `mg` que escriba en la salida estándar los n primeros teoremas de este sistema formal. Se debe garantizar que, dado un teorema cualquiera del sistema, si el programa no tuviera limitaciones y se le permitiera funcionar indefinidamente, dicho teorema siempre sería escrito en algún momento (es decir, debe asegurarse de que el procedimiento de escritura no presente comportamientos sesgados).

Ejemplo de ejecución:

```
$ python mg.py 15
m~g~
m~~g~~
~m~g~~
m~~~g~~~
~m~~g~~~
~~m~g~~~
m~~~~g~~~~
~m~~~g~~~~
~~m~~g~~~~
~~~m~g~~~~
m~~~~~g~~~~~
~m~~~~g~~~~~
~~m~~~g~~~~~
~~~m~~g~~~~~
~~~~m~g~~~~~
```

Sea libre de diseñar y utilizar, si así lo desea, una codificación más compacta para todas estas cadenas. Eso sí, asegúrese de que dicha codificación haga corresponder de manera unívoca las nuevas cadenas con las cadenas originales.

Apartado opcional: Para cualquier sistema formal, se puede considerar la existencia de un *procedimiento de decisión*, es decir, de algún mecanismo que, dada una cadena se símbolos, nos diga si dicha cadena es o no un teorema del sistema. Por supuesto, sólo lo será si se trata de un axioma, o si se trata de una cadena que pueda obtenerse a partir de un axioma aplicando las reglas de producción.

¿Sería capaz de diseñar un procedimiento de decisión para el sistema mg∼? Si es así, modifique el programa mg∼, para que, cuando en lugar de un número reciba en la línea de comandos una cadena de símbolos del sistema, indique si se trata o no de un teorema.

Ejemplos de ejecución:

```
$ python mg.py ~~m~g~~~
yes

$ python mg.py ~~m~g~~
no

$ python mg.py ~~m~m~~~g~~~~~~~
no
```

3. Otro ejemplo de sistema formal es el sistema MIU. Fue ideado también por Douglas R. Hofstadter, y tiene implicaciones más interesantes que las del sistema anterior. Sus elementos son los siguientes:

   - Este sistema cuenta con tres símbolos: las letras M, I y U.
   - Tiene un único axioma: la cadena MI.
   - Y está dotado de cuatro reglas de producción.
     - Regla I: Si se tiene una cadena cuya última letra es I, se le puede agregar una U al final.
       Por ejemplo, dado MI, podemos obtener MIU.
     - Regla II: Supongamos que tenemos Mx. En tal caso, podemos añadir Mxx a la colección de teoremas del sistema.
       Por ejemplo, dado MIU, podemos obtener MIUIU.
     - Regla III: Si en una de las cadenas de la colección aparece la secuencia III, puede elaborarse una nueva cadena sustituyendo III por U.
       Por ejemplo, dado MIIII, podemos obtener MUI (y también MIU).
     - Regla IV: Si en una de las cadenas de la colección aparece la secuencia UU, puede elaborarse una nueva cadena eliminado dicha secuencia.
       Por ejemplo, dado MIUU, podemos obtener MI.

   Construya un programa de nombre miu que escriba en la salida estándar los n primeros teoremas de este sistema formal. Igual que antes, si el programa funcionara indefinidamente, cualquier teorema del sistema siempre sería escrito en algún momento (es decir, el procedimiento de escritura no debe presentar comportamientos sesgados).

   Ejemplo de ejecución:

```
$ python miu.py 20
MI
MIU
MII
MIUIU
MIIU
MIIII
MIUIUIUIU
MIIUIIU
MIIIIU
MIIIIIIII
```

```
MUI
MIU
MIUIUIUIUIUIUIU
MIIUIIUIIUIIU
MIIIIUIIIIU
MUIU
MIUU
MIIIIIIIIU
MIIIIIIIIIIIIIII
MUIIIII
```

Apartado opcional: Como habrá podido observar, puede darse el caso de que algunas cadenas aparezcan varias veces, es decir, se trata de teoremas que pueden obtenerse mediante diferentes aplicaciones de reglas (algo que no era posible en el sistema formal del ejercicio anterior). Más adelante, nos referiremos a este fenómeno como "ambigüedad", y estudiaremos la conveniencia de considerar o no los diferentes "caminos" que conducen a una misma cadena. De momento, efectivamente, no parece demasiado interesante que una cadena pueda ser enumerada más de una vez. Así pues, como en el ejercicio 1, añada al programa `miu` algún otro modo de ejecución que aplique un procedimiento de escritura que no repita elementos, reflexionando previamente sobre la complejidad de implementar de manera adecuada dicho procedimiento.

Hofstadter, en su libro, presentó el sistema formal `MIU` en el contexto de lo que él denominó "El acertijo `MU`". Dicho acertijo consiste en responder a la pregunta de si en este sistema se puede producir o no la cadena `MU`. O, dicho de otra manera, si se puede obtener `MU` a partir de `MI` y de las cuatro reglas de producción citadas anteriormente. ¿Se puede o no? Si se puede, ¿cómo? Si no se puede, ¿por qué?

Y en lo que se refiere al *procedimiento de decisión* del sistema `MIU`, ¿cuál es su opinión? ¿Existe? ¿Es posible implementarlo? En caso afirmativo, hágalo.