Errores conceptuales

Primer Parcial 2024

En este documento presentamos algunos de los errores más comunes encontrados durante la corrección del primer parcial.

Ejercicio 1

Programa 5:

- No mencionar o explicar la función del módulo *re* utilizada para ese programa.
- No incluir ningún ejemplo de expresión regular utilizada para encontrar los patrones pedidos en el ejercicio.
- No incluir el símbolo de búsqueda no greedy (?) en la expresión regular. Por ejemplo: r'#(.*)\\n en vez de r'#(.*?)\\n.
- Mezclar los patrones de búsqueda (##texto## o **texto\n).
- Confundir las expresiones de html. Por ejemplo, "h1" por "h2" o "strong" por "em".

Programa 1:

- Confundir el programa 1 con el programa 2. Es decir, las menciones de usuarios, que comenzaban con el signo @ por la participación de los usuarios en el foro, marcada por el campo user.
- No mencionar o explicar la función del módulo *re* utilizada para ese programa.
- No poner la expresión regular utilizada para encontrar las menciones.
- Incluir en el final de la expresión regular \s o \n. Recordar que las menciones de los usuarios eran alfanuméricas, por lo que terminaban cuando no se encontraban más caracteres alfanuméricos y no necesariamente ante la aparición de un espacio o un \n.

Expresión regular:

- No escapear los asteriscos. Recordar que para referirse al asterisco como símbolo a reconocer, se debe usar "*" en vez de únicamente "*".
- Confundir los * por #
- El ejercicio estaba pensado originalmente para matchear cada ocurrencia de **texto** en una entrada más grande, donde este patrón ocurriera más de una vez. Por lo tanto, la expresión regular ** (.*) ** no contempla este caso, donde el punto (.) también puede matchear asteriscos (*). La expresión correcta para ese caso sería: ** (.*)? ** . Por ejemplo, para la entrada "**hola**como**estas**" la primer expresión regular retorna una ocurrencia de "hola**como**estas" mientras que la segunda encuentra dos ocurrencias "hola" y "estas".

Ejercicio 2

Pasaje de AFND-ε a AFND:

- **Olvidar estados en los** ε_{cl}. Recordar que se deben agregar todos los estados a los cuales se puede llegar a través de una transición épsilon, incluyendo el estado mismo.
- **No poner** q_0 **como estado final del AFND**. El algoritmo de pasaje de AFND-ε a AFND nos dice que el conjunto de estados finales del autómata resultante se conforma de: F' = $\{q_f\}$ (estado final del AFND-ε) o F' = $\{q_f\} \cup \{q_0\}$ si $\varepsilon_{cl}(q_0) \cap F \neq \emptyset$.
- **Agregar más estados que** q_0 **a los estados finales del AFND**. Como se dijo anteriormente, F' puede ser $\{q_f\}$ o $\{q_f\} \cup \{q_0\}$, nunca puede agregarse otro estado que no sea q_0.
- **No poner el desarrollo de los algoritmos.** Al no desarrollar los algoritmos de AFND-ε a AFND y AFND a AFD y solo dibujar los autómatas, no se puede evaluar los pasos en el razonamiento del estudiante y evaluar la gravedad de sus posibles errores.
- **No escribir todas las** $\tilde{\delta}$ **para el algoritmo de AFND-ε a AFND**. La mayoría de los estudiantes que no las escriben terminan teniendo un error de cálculos que, de haberlas desarrollado, seguramente no hubieran tenido.

Relación R_M :

- **Definir la relación** R_M **sobre estados**. La relación R_M es una relación sobre tiras, donde aquellas que al ser procesadas por el autómata determinista terminan en el mismo estado pertenecen a la misma clase de equivalencia.
- **No mencionar que el autómata debe ser determinista al definir la relación** R_M. Si el autómata no es determinista, no se puede definir la relación R_M .
- **No definir x e y al definir la relación** R_M, que son tiras de Σ*.
- **No agregar el estado pozo al autómata**. Las tiras que al ser procesadas llegan al estado pozo, también tienen una clase de equivalencia asociada. La cantidad de clases de R_M es igual a la cantidad de estados del autómata completo.
- **Minimizar el autómata para hallar la cantidad de clases de** R_M. La cantidad de clases de R_M es igual a la cantidad de estados del autómata completo, que no necesariamente tiene que ser mínimo. Recordar que puede haber distintos autómatas que reconozcan un mismo lenguaje, incluso con distinta cantidad de estados y por ende distinta cantidad de clases de R_M. En un caso diferente al de este ejercicio, minimizamos cuando queremos encontrar las clases de R_L usando el

método de clases de equivalencia de R_M , ya que cuando el autómata es mínimo y completo las clases de R_M y R_L coinciden.

Ejercicio 3

- **No incluir el pozo al minimizar el autómata**. El algoritmo de minimización siempre tiene en cuenta al estado pozo, por lo que no incluirlo explícitamente puede llevar a errores de cálculo. Además, es imprescindible incluirlo al calcular las clases de equivalencia de R_L: al dividir a Σ* en diferentes clases de equivalencia, las tiras que no pertenecen al lenguaje también deben estar representadas por alguna clase.
- **Usar el Análisis de Kleene para hallar las clases de equivalencia de R_M.** El análisis de Kleene solamente calcula una expresión regular que genera el lenguaje aceptado por el autómata, por lo que no sirve para hallar las clases de equivalencia.
- **No incluir ε en la expresión correspondiente a la clase del estado inicial**, al plantear el sistema de ecuaciones para hallar las expresiones regulares asociadas a las clases de R_M. Recordar que las Y_i representan "qué forma tienen las tiras que llegan a q_i", por lo que la tira vacía llega al estado inicial por definición.
- **No aplicar bien la distributiva en expresiones regulares**. Este error se da comúnmente cuando la expresión incluye un *pipe* y ε. Por ejemplo: "(a|ε).b = (ab|b)" es correcto, pero "(a|ε).b = ab" no lo es.
- **No usar la expresión regular asociada a los estados finales para verificar si el ejercicio estaba bien**. Analizando la expresión regular hallada se puede ver el tipo de tiras que genera. Si esa expresión genera tiras que no están en el lenguaje *(sobregenera*), o no genera tiras que deberían estar en el lenguaje (*subgenera*), entonces hay algún error. Por ejemplo, algunas soluciones reconocían ε y otras no reconocían la tira más chica.

Ejercicio 4

- **No justificar la respuesta**. Por ejemplo, en caso de dar una ER, debería decirse cómo se deduce (en función de la estructura de sus tiras).
- **Plantear incompleto y/o mal el contrarrecíproco del Pumping Lema al elegir la tira Z**, que lleva a que luego no se consideren las restricciones de los substrings.
- **Elegir mal la tira**. En el caso del lenguaje de ese ejercicio y a modo de ejemplo, z=#a^{2N}# . Notar que las tiras pueden ser reconocidas por un AF. En el estudio de las descomposiciones de *z=uvw*, cuando "*v*" está en las a's no lo puede probar.
- **No justificar la NO pertenencia al lenguaje en el estudio de casos**. Poner algo del estilo: "como p>1 la tira z_i no pertenece" no es suficiente. Se debe aclarar por qué la tira no pertenece al lenguaje, argumentando según las características del lenguaje mismo. Por ejemplo, para el caso prototípico {$a^k b^k$: k>0}, puede ser algo como "la tira tiene más a's que b's, por lo cual no pertenece al lenguaje, que exige que la cantidad de a's y b's sea la misma".

- **No “cerrar” la demostración de que un lenguaje no es regular**. Al final se debe aclarar que se analizaron todas las familias posibles y por el CR del PL el lenguaje no es regular.

Ejercicio 5

Máquina con Salida

- **Estados finales en una máquina con salida**. Las máquinas con salida nunca pueden llevar estados finales, porque aceptar una tira no es su objetivo y por lo tanto no forman parte de su definición.
- **No determinista.** Las máquinas con salida vistas en el curso son deterministas. Si dado un estado y una entrada, existe más de una transición posible, entonces la máquina no es determinista. Notar, entonces, que si hay una transición epsilon definida para un estado, entonces este estado no puede tener otra transición saliente.
- **Saltear patrones atípicos en el lenguaje de entrada**. Las máquinas con salida reciben tiras de un lenguaje. Asumir que la máquina no se tranca ante patrones extraños en la entrada, es un error grave. Lo correcto es dejar las transiciones inesperadas sin definir o incluir un estado pozo, lo cual es conceptualmente equivalente.
- **No detallar la salida en cada caso**. Una máquina de Mealy genera una salida con cada transición y una máquina de Moore lo hace al llegar a cada estado. Es incorrecto asumir que, si no se indica lo contrario, una máquina produce ε ante un estímulo.
- **Las entradas son tiras**. Las máquinas con salida que vemos en el curso reciben símbolos en su entrada, no tiras. Por lo tanto, procesar más de un símbolo con una misma transición es considerado un error.

Máquina de dos cintas

- **Ausencia de estados finales en una máquina de dos cintas**. Las máquinas de dos cintas tienen como objetivo aceptar un par de tiras según un par de lenguajes. No puede haber una máquina de dos cintas que no tenga estados finales.
- **No determinista.** Las máquinas de dos cintas vistas en el curso son deterministas. Si dado un estado (en cualquier cinta) y una entrada, existe más de una transición posible, entonces la máquina no es determinista.
- **Transiciones épsilon**. Las máquinas de dos cintas vistas en el curso no disponen de transiciones épsilon.
- **No indicar el estado inicial**. Si bien no es un error grave, porque se puede asumir que siempre se inicia en el estado de la izquierda con etiqueta alfanumérica más baja, no indicar cuál es el estado inicial es un error. En los autómatas de dos cintas esta decisión es sumamente crítica, pues la elección del estado inicial impacta en cuál es la cinta desde la cual se comenzará a leer.
- **Considerar a los símbolos “<” y “>” como parte del alfabeto de entrada**. A no ser que se estipule lo contrario, los símbolos < y > se usan, como en el resto de áreas relacionadas a la matemática, para definir tuplas.
- **Estados que pertenecen a las dos cintas**. No hay estados que sean comunes a las dos cintas. O se consume entrada desde la cinta izquierda, o se hace desde la derecha.