PRÁCTICA HASKELL

TEORÍA DE LOS LENGUAJES DE PROGRAMACIÓN

JORGE LUIS GRANADOS VILLANUEVA

Cuestiones sobre la práctica

1) **(1’5 puntos). Supongamos una implementación de la práctica (usando la misma estructura aquí presentada) en un lenguaje no declarativo (como Java, Pascal, C…). Comente qué ventajas y qué desventajas tendría frente a la implementación en Haskell. Relacione estas ventajas desde el punto de vista de la eficiencia con respecto a la programación y a la ejecución. ¿Cuál sería el principal punto a favor de la implementación en los lenguajes no declarativos? ¿Y el de la implementación en Haskell? Justifique sus respuestas.**

La principal ventaja de la implementación del Trie en un lenguaje no declarativo es la capacidad de optimización a la que se puede llegar para una ejecución más rápida y un mejor uso de la memoria ya que tenemos un mayor control de flujo de la ejecución, por ejemplo, C nos permite una asignación manual de los datos en memoria pudiendo optimizar el almacenamiento y el acceso a los distintos nodos el árbol (Trie).
Como desventajas podemos encontrar la mayor cantidad de código que puede requerir un programa y que estos son más verbosos, y el menor nivel de abstracción del que estos lenguajes están dotados.

Una de las principales ventajas del uso de lenguajes declarativos destaca que la solución de un problema se puede realizar con un nivel de abstracción considerablemente alto, sin embargo esto puede llegar a ser una desventaja, ya que a medida que crece la complejidad del programa es más difícil que sea entendido por otras personas.
Otra ventaja que nos podemos encontrar en Haskell, como lenguaje declarativo, es la expresividad de la que este tipo de lenguajes está dotado.
Como desventajas podemos encontrar la limitación en el control y rendimiento de la ejecución debido a que en los lenguajes declarativos se enfocan en especificar qué se debe a lograr en lugar de cómo hacerlo, lo que implica que el control y ejecución del programa está determinado por el entorno, esto puede limitar la capacidad de realizar optimizaciones específicas o tener un control detallado sobre el rendimiento del programa en situaciones donde sea necesario un control más detallado del flujo.

2) **(1’5 puntos). Indique, con sus palabras, cómo afecta el predicado predefinico no lógico corte (!) al modelo de computación de Prolog. ¿Cómo se realizaría este efecto en Haskell? ¿Considera que sería necesario el uso del corte en una implementación en Prolog de esta práctica? Justifique sus respuestas.**

La estrategia utilizada por Prolog para hallar las distintas soluciones a un problema es el Backtracking, con el predicado de corte podemos alterar la ejecución del Backtracking, ya que con el podemos parar la evaluación de una rama para que continúe con otras o incluso podemos hacer que la evaluación de soluciones se detenga cuando obtengamos una solución. Gracias a este operador podemos mejorar la eficiencia y rendimiento del programa, específicamente cuando se trabaja sobre grandes conjuntos, sin embargo, esta mejora de eficiencia y rendimiento puede llevar a que se dejen de considerar ramas con posibles soluciones.

Aunque en Haskell no podríamos implementar directamente el operador de corte, podríamos implementar el comportamiento mediante guardas, haciendo que bajo unas determinadas condiciones la ejecución vaya por distintos caminos.

Sí, considero que la utilización del corte sería necesario, ya que evitaríamos recorrer espacios de búsqueda innecesarios, ya que por la propia definición de Trie no puede haber nodos repetidos, haciendo que las cadenas sean únicas, por lo que sería correcto que no se sigan evaluando ramas cuando se haya encontrado una solución, haciendo que la implementación con el Trie sea más eficiente y tenga mejor rendimiento sobre una que no la tuviere.

3) **(1 punto). Indique qué clases de constructores de tipos (ver capítulo 5 del libro de la asignatura) se han utilizado para definir el tipo Stock y para definir los nodos del backtracking utilizado en la función listStock. Justifique sus respuestas.**

El tipo Stock esta compuesto por 3 constructores de tipo, tenemos el constructor ROOTNODE el cual hace referencia a un array de valores de el mismo, utilizando la recursividad, tenemos el constructor INNERNODE el cual esta formado por dos argumentos, uno de tipo Char y otro de un array de Stock, y por último encontramos el constructor INFONODE el cual tiene un argumento de tipo entero que indica el valor asociado a la cadena introducida en el Trie