Instituto Tecnológico de Costa Rica
Administración de Tecnologías de Información
TI-3404 Lenguajes de Programación
Prof. Andréi Fuentes L.
Setiembre, 2014

TAREA PROGRAMADA 2

Introducción

La tarea consiste en implementar un intérprete para un lenguaje de programación lógico similar a Prolog.

Descripción del programa

El funcionamiento del programa es el siguiente:

1. El usuario abre el programa, y se le debe mostrar una consola similar al intérprete de Prolog
2. El usuario ingresa una instrucción, el intérprete la procesa, y muestra los resultados.

Las instrucciones que debe soportar el lenguaje son definición de predicados, y consultas.

Definición de predicados

Cuando el usuario ingrese la instrucción `<define>`, el intérprete entrará en modo de definición de predicados. En este modo, se permitirá que el usuario agregue hechos o reglas a la base de conocimientos. Para agregar hechos y reglas, se permitirá la sintaxis sencilla de Prolog, como se muestra en los siguientes ejemplos:

- Para reglas: `predicado(arg1,arg2,…,argn):-ant-1,…,ant-n.`
- Para hechos: `hecho(arg1,arg2).`

Es importante recalcar varios puntos:

- Toda regla o hecho válido termina con un punto.
- Toda regla o hecho puede tener de ***0*** a ***n*** argumentos
- Toda regla puede tener ***n*** antecedentes

El usuario podrá ingresar cualquier cantidad de hechos y reglas a la base de conocimientos, y estos deberán ser almacenados en memoria temporal, para poder hacer las consultas respectivas posteriormente.

Adicionalmente a los predicados ingresados por el usuario, el sistema deberá tener implementados los siguientes Built-in-predicates: `write(arg)`, `nl`. El comportamiento de estos predicados deberá ser igual que en Prolog.

Para salir del modo de definición de predicados, el usuario ingresará la instrucción `</define>`.

Modo consulta

En este modo, el usuario interactuará con el sistema por medio de consultas. La idea es que se puedan hacer consultas similares a las que se hacen en el sistema interactivo de Prolog, y el sistema deberá revisar los hechos y reglas de la base de conocimientos, con lo que responderá ***YES*** o ***NO***, dependiendo de la información que haya en la base de conocimientos.

En el caso de que una regla tenga antecedentes, deberán validar los antecedentes antes de poder responder, al igual que lo hace Prolog. Para esto, deberán mantener un tipo de pila con los diferentes predicados. Además, deberán poder manejar backtracking, para el proceso de resolución de metas.

Aspectos técnicos

El programa podrá ser escrito en alguno de los siguientes lenguajes: C, C++, Ruby o Python.

El proyecto completo deberá funcionar en el sistema operativo Linux, por lo que es importante que los grupos tomen eso en consideración a la hora de elegir el lenguaje de programación, y las librerías que se usen.

No es necesario tener interfaz gráfica de usuario, ya que la interacción será completamente en modo consola.

Documentación

La documentación externa deberá incluir:

- Descripción del problema

- Diseño del programa: decisiones de diseño, algoritmos usados, descripción de principales predicados, lenguaje de programación escogido, y las razones detrás de dicha escogencia. Es importante que en esta sección documenten bien todas las decisiones tomadas a la hora de desarrollar la tarea.

- Librerías usadas: librerías de GUI usadas, etc.

- Análisis de resultados: descripción de las tareas que se completaron y de las tareas que no se lograron completar, además de algunas propuestas de solución para los problemas que no se pudieron resolver

- Manual de usuario: instrucciones de uso de modalidad administración y consulta, instrucciones de compilación (en caso de requerir librerías adicionales, deberán detallar todos los pasos que se deben ejecutar para compilar y ejecutar el código, además de incluir los archivos de las librerías necesarios dentro del código fuente que entregarán)

- Conclusión personal

Evaluación

La siguiente rúbrica especifica cómo se calificarán los diferentes elementos que forman parte de la evaluación.

Criterio	Peso	Excelente (100%)	Adecuado (70%)	Regular (30%)	Deficiente (0%)
Documentación interna	3%	Los comentarios de las clases y funciones explican claramente lo que hacen. Los nombres de funciones, variables y clases son claros, y permiten entender perfectamente el código.	Una o dos secciones podrían tener mejores comentarios, tanto en clases como en funciones.	Hay muchas secciones del código sin comentar. correctamente Muchos de los comentarios no son lo suficientemente claros. Usan nombres de variables poco claros.	El código fuente no está comentado del todo
Documentación externa	7%	Usan buena redacción. La documentación tiene todas las secciones. Las secciones tienen suficiente detalle para entender.	Errores de redacción menores. La documentación tiene todas las secciones, pero falta detalle en al menos una de las secciones.	Errores de redacción frecuentes. La documentación no tiene todas las secciones, o falta profundizar en alguna de las secciones.	No entregan documentación.
Definición de predicados	30%	Validan correctamente la sintaxis de los predicados ingresados por el usuario. Usan correctamente las instrucciones <define> y </define> para delimitar la definición de predicados. Guardan correctamente los predicados en la base de conocimientos.	No validan totalmente la sintaxis de los predicados ingresados. Hay errores menores a la hora de ingresar los predicados.	No validan la sintaxis de los predicados ingresados, o hay errores graves a la hora de ingresar los predicados.	No se pueden ingresar predicados nuevos a la base de conocimientos.
Modo de consulta	40%	Validan correctamente la sintaxis de las consultas. El motor de inferencia funciona adecuadamente, y devuelve los resultados correctos todas las veces.	Errores menores en la validación de la sintaxis. El motor de inferencia tiene errores menores a la hora de devolver los resultados	El motor de inferencia tiene errores graves a la hora de resolver las consultas hechas por los usuarios	El modo de consulta no funciona del todo
Revisión	20%	Responden adecuadamente a cada una de las preguntas hechas durante la revisión. Demuestran conocimiento avanzado de la materia y de la solución entregada.	Muestran conocimiento medio de la materia y de la solución entregada.	Muestran conocimiento muy básico de la materia. No responden correctamente algunas de las preguntas.	No asisten a la revisión.
Uso de C++	20% extra	Usan perfectamente la sintaxis de C++. Usan clases, métodos y funciones correctamente.	Tienen uno o dos errores causados por el uso de C++	Tienen más de dos errores causados por el uso de C++	No usan C++
Uso de Parsers y ER	20% extra	Usan perfectamente herramientas como Bison (generadores de parsers) para validar sintaxis	Tienen uno o dos errores al usar generadores de parsers	Tienen más de dos errores a la hora de usar los generadores de parsers.	No usan herramientas de generación de parsers, ni tampoco ER

Aspectos administrativos

- La tarea vale un 10% de la nota del curso

- La tarea se hará en grupos de 3 personas.

- Fecha de entrega: Martes 7 de octubre, 9 a.m. No se aceptan tareas entregadas después de esa fecha y hora, sin excepciones.

- Los grupos deberán subir el código y la documentación de sus respectivas tareas a un repositorio en Github, de manera que el profesor pueda ver las contribuciones que las diferentes personas hacen al proyecto. La idea es que apenas empiecen a desarrollar la tarea, suban las contribuciones al repositorio, y no esperar a tener todo el código listo para subirlo.

- Deberán enviar un correo a andreifu@gmail.com, con copia a evelyn.madriz@gmail.com, en donde indiquen el url del repositorio de Github en donde se encuentra el código y la documentación de la tarea. El asunto del correo enviado tendrá el siguiente formato: `TI-3404 TP2-Nombre1-...-Nombre`*`n`*. Las tareas que no sean entregadas por medio de Github tendrán nota cero.

- Las tareas deberán ser revisadas con el profesor o el asistente. Todos los miembros del grupo deberán participar de la revisión, ya que de lo contrario no se les asignará el puntaje correspondiente. La nota de la revisión es individual, el resto de la nota es grupal.

- En caso de probarse algún tipo de fraude en la elaboración de la tarea, se aplicarán **todas** las medidas correspondientes, según el reglamento del TEC, incluyendo una carta al expediente.