Trabajo práctico Nº 1: Verificación de Programas

Federico Bugni, Federico Paulovsky, Gervasio Perez

September 22, 2011

1 Ejercicio 1

TODO.

2 Ejercicios 2 y 3

En esta sección trataremos los aspectos del TP relacionados a la implementación del verificador Pest.

2.1 Consideraciones generales de diseño

En nuestro TP utilizamos extensivamente el patrón *Visitor* para las recorridas de los árboles sintácticos de programas Pest, de predicados lógicos y de términos.

2.2 Desarrollo

2.2.1 Primer enfoque de verificación

Inicialmente optamos por implementar el verificador armando una fórmula lógica con cuantificadores, lo cual probó ser arduo especialmente con respecto al *debugging*. Luego de la clase práctica del Miércoles 14/9 decidimos cambiar el enfoque completamente, convirtiendo el output CVC3 en una secuencia de declaraciones de variables, y comandos ASSERT y comandos QUERY para verificar condiciones. El cambio no resultó problemático porque recién estábamos llegando a implementar el if y buena parte del código de los Visitors pudo ser reutilizada. De todas maneras

2.2.2 Versión final

Para el nuevo enfoque adoptamos el concepto de *contexto* mencionado en clase.

2.3 Clases desarrolladas

Para la realización del TP desarrollamos los paquetes **budapest.pest.pesttocvc3** y **budapest.pest.predtocvc3** que contienen el código de traducción. Éstos contienen las siguientes clases:

- PestToCVC3Translator: Implementación de Visitor que recorre un programa Pest y lo traduce, devolviendo un String con los comandos CVC3 que lo verifican.
- PestVarContext: Representación de un contexto de variables Pest versionadas.
- PredVarReplacer y TrmVarReplacer: Visitors que realizan el reemplazo de variables en Predicados y en Terms respectivamente. Hacen uso de un PestVarContext de donde toman el nombre que debe usarse para cada variable en el reemplazo.
- PredParamReplacer y TrmParamReplacer: Similares a los anteriores, pero hacen un reemplazo de un nombre de variable por un Term cualquiera. Necesarios para implementar el reemplazo de variables en una llamada a función.
- PestVarBinder: Auxiliar para calcular el mapeo de nombres de parámetro de un procedimiento a las expresiones que se usaron para invocarlo.

2.3.1 PestToCVC3Translator

Este Visitor hace uso de contextos de variables para resolver el nombre que debe tener cada "instancia" de una variable al momento de usarla en una fórmula CVC.