AAP - Trabajo Práctico 3: Sistemas de tipos

Bugni , Paulovsky , Perez (Grupo 2)

December 6, 2011

1 Ejercicio 1

Para implementar el soporte de *strings* en Pest se realizaron las siguientes modificaciones al código existente:

- la extensión del léxico y la gramática de Pest para poder escribir literales de *strings*;
- la implementación de clases para realizar la inferencia del tipo de una expresión del lenguaje.

1.1 Cambios en el lenguaje

Dado que en un principio el único tipo disponible era *int*, fue necesario agregar (además del tipo *string*) el tipo *Top* para representar una subexpresión de tipo desconocido.

Fue necesario agregar al parser de Pest (tanto en su gramática como en sus clases auxiliares):

- términos literales que representan *strings*;
- el operador |.| que devuelve la longitud de un *string*;
- la abstracción de muchas apariciones de expresiones Int por expresiones Top, como en el caso del operador . + . (AdditiveIntExp cambiada por AdditiveTopExp) que en principio "suma" dos expresiones de tipo desconocido, y que en nuestra extensión pueden ser o bien ambas String o bien ambas Int.

Los cambios mencionados fueron realizados de manera análoga para Terms (permitiendo así usar *strings* en predicados de especificación de pest).

1.2 Implementación de la inferencia de tipos

La inferencia de tipos se realiza a través de las clases desarrolladas en el paquete **budapest.pest.typeinference**. Algunas de ellas son:

- *PestTypeInferenceManager*: implementa un Visitor de código Pest que mantiene un contexto de tipos conocidos y responde si pudo o no tipar correctamente un programa; delega en Visitors especializados la inferencia de expresiones / terms / predicados.

- *PestTypedContext*: representa a un contexto de tipado de expresiones. Implementa la operación de unión de contextos que intenta unificar los tipos de los contextos a unir (lo cual podría resultar en un fallo).
- *PestTypingConstant*: implementación de variables de tipo para su uso en la unificación de tipos mencionada.
- *[Pred Trm Exp]TypeInferenceManager*: Visitors que intentan juzgar el tipo de una Pred / Trm / Exp basándose en el contexto y en las reglas de la semántica. En estas clases es donde se resuelve la inferencia del tipado de expresiones de la forma $e_1 + e_2$ y $|e|$. Para el caso de los Preds, además, se intenta inferir que la expresión sea de tipo *Bool*.
- *[Pred Trm Exp]TypeJudgement*: clases que encapsulan el resultado de juzgar el tipo de una Pred / Trm / Exp.
- *Pest[Int Bool String]Type*: clases que representan los tipos disponibles para el uso en el contexto de tipos.