Compiladores

Linguagem *AGL*

Departamento de Eletrónica, Telecomunicações e Informática
Universidade de Aveiro

abril de 2024

Objetivos

O objetivo geral deste trabalho é o desenvolvimento de um ambiente de programação, consituído pela linguagem de programação AGL (*animated graphics language*) e correspondentes ferramentas de compilação, que permita a criação de programas na linguagem de programação genérica Python3, usando a biblioteca `tkinter`, que, quando executados, permitem a visualização, com possível animação e interação, de gráficos 2D.

A linguagem assume (implicitamente) a existência de uma área de desenho, *canvas*, com dimensões ilimitadas, sobre a qual se podem instanciar (desenhar) figuras gráficas 2D. Podem também ser instanciadas vistas (`View`), que permitem capturar o estado do *canvas* numa dada região, com um determinado zoom e num determinado instante no tempo. Há um conjunto base de modelos gráficos pré-definido, mas a linguagem possui mecanismos construtivos que permitem definir outros. As instâncias dos modelos gráficos podem, ao longo do tempo, mudar a sua posição e as suas propriedades, mas, apenas quando uma vista é *refrescada* é que as alterações são capturadas por essa vista.

A concretização do objetivo geral deste projeto pode ser decomposto em:

- Definição da linguagem principal AGL, associada a ficheiros com a extensão `agl`, que permite a definição de figuras gráficas 2D e a sua visualização com animação e interação. O resultado desta tarefa é a descrição em ANTLR de uma gramática adequada à análise sintática de descrições em AGL.
- Construção de um analisador semântico que detete e sinalize erros de descrição AGL que escapam à análise sintática. Compete a este analisador verificar se as variáveis são declaradas adequadamente antes de ser utilizadas, se há repetições na declarações de variáveis, se as atribuições e operações são válidas, se os atributos (propriedades) dos vários modelos gráficos são válidos, etc. O resultado desta tarefa é o código fonte (*visitor*), na linguagem Java, do analisador semântico desenvolvido, assim como a sua adequada invocação no programa principal.
- Construção de um compilador que permite transformar uma descrição AGL num programa equivalente na linguagem destino (Python3), usando a biblioteca `tkinter`, e que, quando executado, produza a visualização desejada. Encoraja-se e valoriza-se o uso de *string templates*. O resultado desta tarefa é o código fonte do compilador (*visitor*, outro que não o do analisador semântico), na linguagem Java, assim como a sua adequada invocação no programa principal, além do ficheiro com os *string templates* utilizados.

- Pretende-se que a linguagem principal permita a importação de elementos auxiliares através de descrições numa linguagem secundária, cujos elementos, em *runtime*, possam ser carregados pelo programa final. Em concreto, pretende-se a definição de uma linguagem secundária, aqui designada por $_{x}$AG$_{L}$, associado a ficheiros com a extensão `xagl`, que permita auxilar a linguagem AG$_{L}$ de alguma maneira (por exemplo, definindo modelos). O resultado desta tarefa é a gramática (ANTLR) da linguagem $_{x}$AG$_{L}$, a construção gramatical na linguagem AG$_{L}$ que permite fazer a importação de descrições em $_{x}$AG$_{L}$, o interpretador que permite fazer o *parsing* de descrições $_{x}$AG$_{L}$, e exemplos ilustrando a sua utilização. Note que este interpretador tem de ser desenvolvido na linguagem destino (neste caso Python3) e não na linguagem de trabalho, uma vez que a interpretação será feita em tempo de execução (*runtime*).

Características da solução

Há um conjunto base de modelos gráficos que são parte integrante da linguagem, podendo ser definidas outros, construídos usando esse conjunto base ou outros modelos entretanto definidos. Cada modelo (base ou não) possui propriedades, como por exemplo cor e espessura de traço, cor do preenchimento (se aplicável), etc.

Apresentam-se a seguir um conjunto de características que a solução desenvolvida deve ou pode contemplar. Essas características estão classificadas a 3 níveis:

- nível mínimo – características que a solução tem obrigatoriamente que implementar;
- nível desejável – características não obrigatórias, mas fortemente desejáveis que sejam implementadas pela solução;
- nível adicional – características apenas consideradas para avaliação se as mínimas e as médias tiverem sido contempladas na solução.

Nível mínimo

Os ficheiros `ex01.agl`, `ex02.agl` e `ex03.agl` representam descrições AG$_{L}$ que a solução de nível mínimo deve aceitar e compilar. Os ficheiros têm comentários que permitem perceber o significado de cada nova construção gramatical introduzida.

A linguagem AG$_{L}$, na sua versão de nível mínimo, deve incorporar:

- Suporte para comentários, de linha e de bloco, tal como ilustrado no exemplos.
- Instanciação de uma vista (`View`) sobre a área de desenho (*canvas*). As vistas devem suportar as ações de `move`, `refresh`, `wait` e `close`. Neste nível (mínimo), pode considerar que há apenas uma vista.
- Instanciação dos modelos gráficos base `Dot`, `Line`, `Rectangle`, `Ellipse`, `Text`, `Arc`, `ArcChord`, e `PieSlice`. Todos estes modelos têm implicitamente um ponto de referência, cuja localização no *canvas* será indicada aquando da sua instanciação. Todos os objetos gráficos devem suportar a ação de `move`. Todos os objetos gráficos possuem um conjunto de propriedades, que podem ser alteradas em tempo de execução.

- Suporte dos tipos de dados `Integer`, `Number`, `Point`, `Vector`, `String` e `Time` e da instanciação de variáveis desses tipos. Suporte de expressões envolvendo estes tipos de dados.
- Suporte da construção `with`.
- Uma construção gramatical repetitiva `for`, para iterar sobre uma sequência de valores.
- Verificação semântica no uso de variáveis e de expressões e na manipulação de propriedades.
- Definição e implementação de uma linguagem secundária.

Nível desejável

Desejável e adicionalmente à versão de nível mínimo, a linguagem AG$_L$ deve incorporar:

- Instanciação dos modelos gráficos base `Polyline`, `Spline`, `Polygon` e `Blob`. Todos estes modelos têm implicitamente um ponto de referência, cuja localização no *canvas* será indicada aquando da sua instanciação. Todos os objetos gráficos devem suportar a ação de `move`. Todos os objetos gráficos possuem um conjunto de propriedades, que podem ser alteradas em tempo de execução.
- O tipo de dados `Boolean` e a manipulação de expressões booleanas.
- Uma construção gramatical condicional.
- Uma construção gramatical repetitiva baseada em predicado (condição de termo ou de continuação).
- Suporte para a especificação de vetores (`Vector`) em coordenadas polares.
- A possibilidade de definição de novos modelos (associada à palavra-chave `model`), que permita agregar instâncias de outros modelos, base ou definidos anteriomente, numa nova entidade. Todas as propriedades destes novos modelos serão definidas aquando da sua conceção. Deve ser possível fazer depender as propriedades dos modelos incorporados de propriedades do modelo principal. O novo modelo poderá ser depois instanciado.
- Suporte para estrutura de dados iterável (*array*, lista, ...) e de mecanismos de instanciação, acesso e manipulação dos seus elementos.
- Possibilidade de aplicar as ações `move`, `refresh` e `close` diretamente a uma lista de objetos.

Nível adicional

Chama-se de novo a atenção de que acrescentos a este nível apenas serão avaliados se as características de níveis mínimo e desejável estiverem praticamente a 100%. Adicionalmente à versão que cubra os níveis mínimo e desejável, a linguagem AG$_L$ pode incorporar:

- Suporte para a rotação de objetos gráficos.
- Suporte para várias vistas.
- ...

Desafios

- Pretende-se criar uma animação interativa que permita a um utilizador resolver as torres de Hanoi.