Transformação de Programas CMinus

Daniel Rocha and Jorge Mendes

Departamento de Informática, Universidade do Minho
pg15127@alunos.uminho.pt
a49319@alunos.uminho.pt

Resumo Um dos objectivos do modulo de Análise e Transformação de Programas é incutir aos alunos da UCE de Engenharia de Linguagens a capacidade de desenvolver software como uma tarefa de transformar programas e/ou especificações em implementações eficientes. Desta forma ficou estipulada a realização de uma pesquisa sobre o tema "Transformação de Programas", em que um dos elementos a analisar eram as ferramentas utilizadas para levar a cabo as transformações e suas aplicações.
Neste documento está apresentada uma análise de como se implementaria uma transformação utilizando o Stratego XT, uma ferramenta de análise e transformação de código, identificando claramente os passos a seguir e os raciocínios a efectuar para levar a cabo a transformação pretendida.

1 Introdução

De forma a incutir aos alunos de UCE de Engenharia de Linguagens uma análise mais aprofundada sobre as potencialidades, aplicações e formas de utilização sobre as ferramentas de análise e transformação de programas, estipulou-se um exercício que consiste em analisar quais os passos fundamentais que levariam a cabo a implementação de uma transformação em Stratego/XT sobre uma versão simplificada da gramática do C, o CMinus.

O caso de estudo consiste em explicar, de forma clara, quais os principais passos que levariam a implementação de uma transformação em Stratego/XT que cumprisse as seguintes funcionalidades, em primeiro construisse a AST correspondente ao excerto de código declarado na linguagem CMinus, de seguida detecta-se o uso de identificadores não declarados percorrendo a AST criada e por fim construísse a declaração válida desses mesmos identificadores na AST, dessa forma obteríamos uma segunda AST que corresponderia a um excerto de código com as variáveis correctamente declaradas.

Com esta análise não só se consegue obter uma análise mais séria de como se implementam transformações utilizando a ferramenta de análise e transformação de código Stratego/XT como também se obtem uma primeira experiência de como se deve proceder para se conseguir perceber e utilizar as funcionalidades de uma ferramenta de transformação e como poder levar a cabo as funcionalidades pretendidas.

2 Termos utilizados

Transformação de Programas Transformação de programas é uma técnica de suporte a actividades na área da engenharia de software. Esta consiste em alterar um programa num outro através de aplicações de transformações, aplicações essas que podem dividir em duas áreas, *Translation* e *Rephrasing*.

Strategic Term Rewriting Pode ser encarado como o paradigma que estuda as estratégias de reescrita através de padrões programáveis de código, obtendo assim as transformações pretendidas;

SDF/Syntax Definition Formalism Consiste num formalismo para a definição de uma sintaxe em concreto, em que através de processos de desambiguação declarativa, não há necessidade de introduzir não-terminais para resolver conflitos e ambiguidades de uma sintaxe;

SDL/Domain Specific Languages Em engenharia de software, uma Linguagem de Domínio Especifico é o termo dado à especificação de uma linguagem dedicada a um determinado domínio ou problema.

AST/Abstract Syntax Tree Representa a estrutura dos dados submetidos a análise sob a forma de árvore, em que se estabelece a correspondência entre os símbolos reconhecidos e as regras definidas na gramática;

Pretty-Print Consiste na conversão de conteúdos tais como estruturas de dados ou fontes de dados para um formato apresentável e fácil de entender.

3 Stratego/XT

Stratego/XT consiste numa ferramenta de implementação de sistemas de transformação com base em representações estruturais de programas.

Stratego/XT é divido em duas componentes, o Stratego e XT. O XT é um conjunto de componentes que compõe a infra-estrutura básica para a transformação, como análise e representação. Stratego, é uma linguagem que permite definir regras de reescrita e para exprimir transformações básicas, bem como outras operações mais complexas em que se baseia no paradigma *strategic term rewriting*.

Juntos, Stratego e XT têm como objectivo proporcionar uma melhor produtividade no desenvolvimento de sistemas de transformação, através do uso de componentes como e funcionalidades, rais como:

- Abstracções e representações de alto nível;
- *Domain Specific Languages*;
- Geradores de programas para os diversos aspectos dos sistemas de transformação como por exemplo ATerm que é utilizado para proceder a representações de programas. O ATerm usa uma gramática muito simples de trabalhar e dessa forma obtém-se uma base para produzir uma representação de um programa de forma simples. Um ATerm é uma expressão que segue a seguinte gramática:

t	→ bt	basic term
	| bt { t }	annotated term
bt	→ C	constant
	| C(t1, ..., t2)	n-ary constructor
	| (t1, ..., t2)	n-ary tuple
	| [t1, ...,t2]	list
	| "ccc"	quoted string
	| int	integer
	| real	floating point number
	| blob	binary large object

- SDF utilizado para a definição de sintaxe e análise
- GPP para impressões personalizadas;
- Stratego para a definir as transformações, de forma algo fácil de utilizar dado que nos comandos compilação e execução é deixada a possibilidade de utilizar pipelining, tal como no seguinte exemplo:

  ```
  $ geraAst programa.cm | PrimeiraTransformação | SegundaTransformação | pretty-print
  ```

- XTC como ferramenta de composição de transformações e as ferramentas XT para a geração de estruturas intermédias necessárias para a construção de sistemas de transformação.

O sistema de composição XTC permite a combinação de componentes encontrados na arquitetura XT com novos componentes elaborados em Stratego. Isto faz Stratego/XT um ambiente escalável e flexível para o desenvolvimento de sistemas de transformação autônomos.

4 CMinus

4.1 Análise Léxica

Para fazer uma análise semântica ao programa, precisamos de fazer uma análise léxica, para saber como é o programa. Assim, temos de descrever a estrutura do programa, i.e., descrever a linguagem CMinus.

A descrição da linguagem CMinus é feita usando a linguagem SDF. Nesta descrição temos de indicar quais são os elementos que a compõem, e como está estruturada.

Tendo a linguagem CMinus completamente descrita, geramos uma tabela de análise, que nos permitirá posteriormente fazer uma análise semântica ao programa pretendido.

4.2 Análise Semântica

Usando uma ferramenta de análise, e passando como parâmetros a tabela de análise e um programa escrito em linguagem CMinus, pode-se gerar uma AST, no formato ATerm, que nos permite fazer uma análise semântica ao programa.

Esta AST pode ter sido obtida através de uma árvore de *parsing*, e neste caso sabe-se que está bem formada, ou pode ser resultado de uma transformação, mas sem garantias que esteja bem formada. Para verificar que a AST esteja correcta, é necessário descrever a estrutura da árvore usando uma gramática, chamada de *regular tree grammar*. Tendo esta gramática, pode-se usar uma ferramenta para verificar se a AST está correcta. Isto é importante para verificar se o resultado das transformações que se efectuam no programa não o invalidam.

Tendo então uma AST para analisar correcta, verificamos se o sentido do programa também o é. Pelos requisitos enunciados para este trabalho, é necessário verificar se as variáveis do programa estão declaradas, tendo em atenção se a declaração está no mesmo nível, ou num dos níveis acima, onde a variável é utilizada. Para esta verificação, definimos uma variável no programa Stratego, em que essa variável consista numa lista das definições variáveis do programa a analisar, com a indicação do bloco onde é declarada.

Os elementos da lista podem ser *strings*, que comecem pelo nome da variável, e continua com cada um dos blocos em que a declaração da variável está contida, e em que cada elemento dessa *string* esteja separado por um ponto. A linguage Stratego permite dividir uma *string* em diversos elementos, separados por um dado elemento, pondo-os numa lista. Também permite a operação inversa.

Com esta análise, obtemos uma lista de todas as variáveis que está declaradas na programa a analisar.

4.3 Transformação do Programa

Tendo a lista da variáveis declaradas, procuramos agora as variáveis que são usadas, e verificamos se elas estão declaradas e se a sua declaração está feita num sítio válido. Caso algo não esteja correcto, é necessário adicionar a declaração ao programa.

Para saber se uma variável está declarada, e a declaração feita num sítio válido, é necessário ter o seu nome e o seu local de uso. Depois, basta verificar se está na lista das variáveis declaradas, fazendo uso do formato em que está guardado o nome e local de declaração para maior facilidade de verificação da validade do local de uso.

Ao encontrar o uso de uma variável que não esteja declarada, pode-se fazer uso de uma lista, de formato semelhante à lista das declarações das variáveis, e guardar nela todas as variáveis a serem declaradas. É claro que é necessário verificar se a variável já está na lista ou não, para não haver declarações repetidas. Tendo estes dados, pode-se fazer uma passagem posterior pela árvore, e no início de cada bloco, adicionar as declarações necessárias. Uma outra abordagem seria fazer uso da recursividade na análise, e adicionar as declarações na mesma passagem em que se verifica se as variáveis estão declaradas.

4.4 Resultado

O resultado obtido consiste numa AST, e não é útil para ser compilada pela compilador da linguagem CMinus. Por isso, é necessário passar a AST do programa para a sua representação textual na linguagem CMinus.

Portanto, para apresentar o programa na linguagem CMinus, faz-se a verificação da AST como descrito na análise semântica e, caso esteja tudo correcto, faz-e o *pretty-print* da AST. Para isso, basta usar ferramentas que o Stratego/XT disponibiliza, e a partir da descrição da linguagem no formato SDF, é gerado uma tabela de *pretty-print*.

A tabela de *pretty-print* é usada por uma ferramenta que, dada uma AST, gera o programa no formato pretendido. Contudo, o programa apresentado não é completamente legível para os seres humanos, apesar de ser sintacticamente correcto, pelo que é necessário dar uns retoques na tabela de *pretty-print* para se obter um resultado satisfatório.

Nota. Esta é apenas uma abordagem proposta, podendo serem utilizadas outras abordagens.

5 Conclusão

Neste documento estão identificados os passos e a forma de implementar uma transformação recorrendo ao Stratego/XT.

Após terem sido abordados alguns termos e ferramentas importantes para perceber o funcionamento do Stratego/XT procedeu-se à explicação de como implementar a funcionalidade pretendida recorrendo a esta ferramenta de análise e transformação de código.

Desta forma vimos neste trabalho que a implementação de uma transformação não é um acção simples, mas facilitada graças ás funcionalidades oferecidas na ferramenta Stratego/XT.

Referências

[Str10] Stratego/XT. Stratego/xt. http://strategoxt.org/, Março 2010.