Primeiro Teste

Linguagens Para Domínio Específicos
2013/14
Segundo Teste – 13 de Maio de 2014
Departamento de Informática
Universidade Nova de Lisboa
(duração 2 horas)

– Sem consulta –

Boa Sorte!

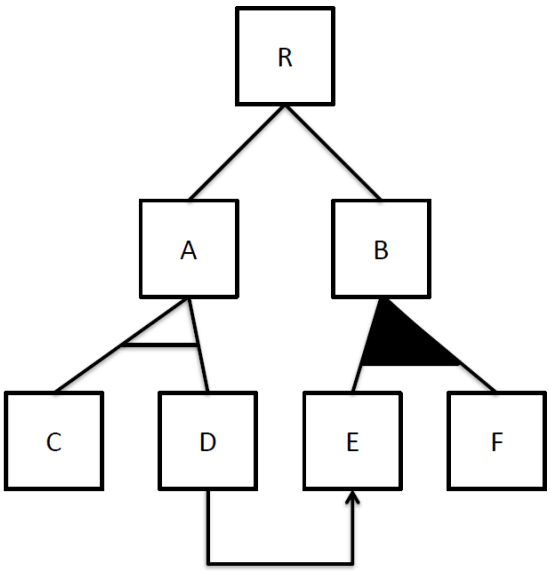
Parte I – Definições em LDE / DSL

1. O que é uma DSL? (definição, como de define enquanto linguagem o seu “triângulo dourado”, como se distingue duma GPL, benefícios, desvantagens)

2. Porque é que é relevante um Feature Model para definição de uma DSL? Em que fase de Engenharia de Linguagens é que o usamos?

3. O que se espera da Análise de Domínio? Quais os inputs e outputs desta fase?

Parte II – Feature Models



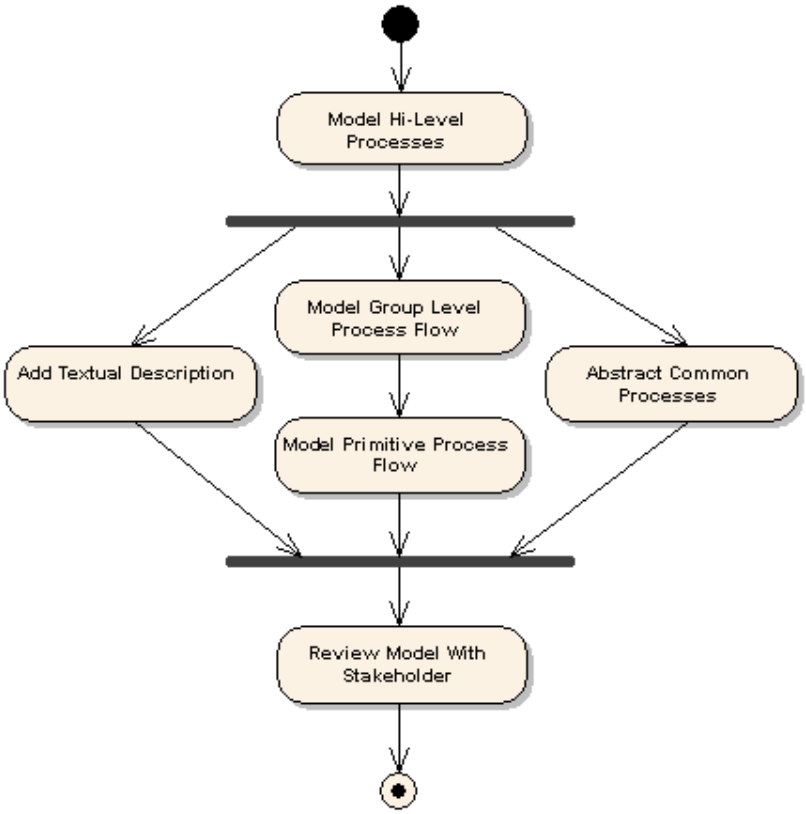
3.Quantas variantes de produtos resultam de todas as configurações possíveis (instâncias) deste Feature Model?

4. Traduza a semântica deste Feature Model para uma expressão em lógica proposicional

Parte III – Modelos e Metamodelos

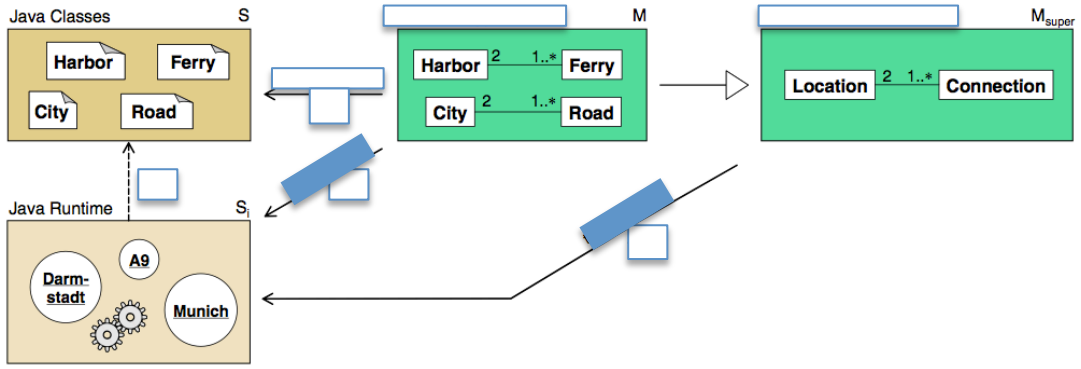
5. Desenhe o esquema de quatro camadas de modelação (como apresentado por Thomas Khuenne ou como previsto nos standards da OMG) e apresente os quatro níveis instanciados para o caso dos Traffic Lights apresentados nas aulas.

6. Considere uma linguagem para especificar Activity Diagrams. Sendo a figura seguinte uma instância (frase) nessa linguagem:



Proponha um metamodelo (sintaxe abstracta), podendo para tal usar notação inspirada no Ecore, da referida linguagem que compreenda todos os tipos de elementos da figura.

7. O trabalho desenvolvido por Thomas Khune, apresentado nas aulas de LDE apresentam dois conceitos: Token Model e Type Model. Identifique no esquema seguinte as relações de "token model of" e "type modelo of" para um programa java que implemente um mapa (ex. GPS).



Parte I – Resposta múltipla

Faça corresponder apenas os elementos da coluna da esquerda que digam respeito a usos típicos de transformações de modelos com a descrição correspondente da coluna da direita.

1	Parser		the target model is an “updated” version of the source model: no new model is created
2	Translational Semantics		Translate complex language constructs into more primitive language constructs
3	Synchronization		The semantics of the source formalism is given in terms of the semantics of the target formalism.
4	Approximation		Transformations are used to refine the search using rute force
5	Optimization		Extracts higher level specifications from lower level ones
6	Analysis		Change the internal structure of the model to improve certain quality characteristics without changing its observable behaviour
7	Reverse Engineering		Transform from a lower level specification to a higher level description
8	Decrease Syntatic Complexity		Mapping from the abstract syntax to possibly several concrete representations (textual, graphical, ...)
9	Migration		Improve certain operational qualities of the model while preserving its semantics
10	Backtracking		Integrate models that have been produced in isolation into a compound model
11	Abstraction		Automatically generate random models that conform to the language
12	Rendering		Transform all uses of a language construct in a normal or canonical form
13	Operational Semantics		Map a modeling language to a formalism that can be analyzed more appropriately than the original language
14	Meta-model instance generation		Transform from a higher level specification to a lower level description
15	Query		Refinement with respect to negated properties
16	Refinement		Transform from a software model written in one language or framework into another, but keeping the same level of abstraction
17	Refactoring		Integrate models that have evolved in isolation but that are subject to global consistency constraints
18	Normalization		Mapping from the concrete syntax to the corresponding abstract syntax (graph)
19	Composition		algorithms It is a projection, obtained by CRUD operations on the properties of M.
20	Model-to-code		Model is synthesized into a well-defined language format that can be stored, such as in serialization

Parte II – Preencher

1- Como classifica transformações para “Composition”, “Optimization” e “Reverse Engineering”? Coloque nos quadros seguintes as transformações mencionadas nos respectivas células de classificação.

	Exogenous	**Endogenous**	**Either**
Out-Place			
In-Place			
Either			

	Exogenous	**Endogenous**	**Either**
Vertical			
Horizontal			

Parte III – Gramáticas de Grafos

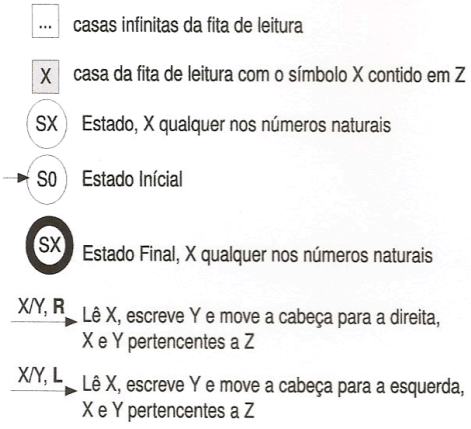
1- Descreva o que são em gramáticas de grafos as situações de a) “dangling edges” e b) “matches” não injectivos, e indique quais são as diferença principais para a resolução destas situações entre uma gramática de grafos que usa regras Double-Pushout (DPO) ou Single-Pushout (SPO) .

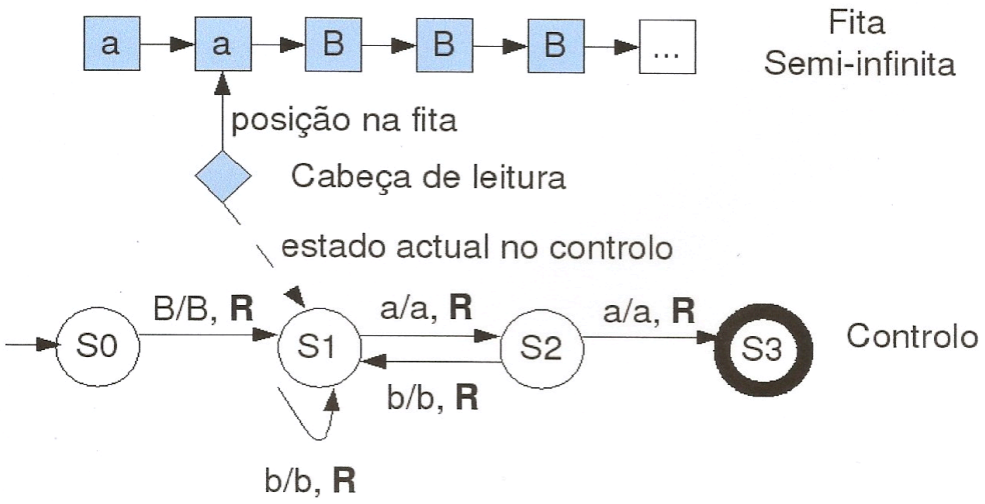
A Máquina de Turing (MT) foi introduzida por Alan Turing em 1936 como modelo de computação algorítmica. O seu modelo básico compreendia uma fita semi-infinita, uma unidade de controlo finita e uma cabeça de leitura e escrita.

M = (T, Q,Z, qI, δ)

– **Q** conjunto finito de estados.
– **Z** alfabeto da fita, contendo B (branco).
– **T** Z – {B} alfabeto de entrada.
– **qI** ∈ Q estado inicial.
– **δ** função de transição – É uma função parcial de Q X Z em Q X Z X {L,R}.

Considere agora a seguinte notação visual para descrição de Máquinas de Turing e o exemplo de modelo para aceitar palavras descritas pela seguinte expressão regular (a+b)*aa(a+b)*:





Especifique com regras transformação de grafos baseadas em DPO a semântica operacional da MT. (atenção que, se a cabeça de leitura alcançar na fita o nó referente às casas infinitas, precisamos de adicionar mais um nó de leitura normal com o B -Branco - lá contido)

Nota: Apresente as regras com o formato: L-K-R ou NAC(s)-L-K-R

FIM!