Brincando às Linguagens com rigor: Engenharia Gramatical

Pedro manuel Rangel santos Henriques[1]

[1] Departamento de Informática – Escola de Engenharia – Universidade do Minho

Abril, 2012

Enquadramento

Para constituir um dos elementos requeridos para as provas de agregação, elaborei um documento intitulado

Brincando às Linguagens com rigor: Engenharia Gramatical

enquadra, *justifica* e *disserta* sobre o tema

linguagens e como lidar sistematicamente com elas

Motivação para o tema escolhido

Decidi discorrer de forma original sobre um tema que me é grato
e que julgo sintetizar a minha aprendizagem e a visão pessoal que criei
ao longo dos anos de ensino e investigação na área das Linguagens
Formais (de Programação),
reflectindo para isso sobre

- *qualidade de linguagens*,
- *qualidade e métricas para gramáticas*

Motivação para o tema escolhido

Além disso, considero que o assunto em causa pode adequadamente constituir
a matéria de uma UC a ser lecionada ao nível de uma pós-graduação em informática,
como p.ex. a Unidade Curricular em Engenharia Gramatical (EG) ontem discutida.

Motivação para o tema escolhido

Com esta dissertação pretendo, ainda, abordar e interligar tópicos que justifiquem claramente o desenvolvimento de *métodos*, *técnicas* e *ferramentas* para

- conceber,
- desenvolver (escrever, analisar e depurar),
- processar,
- avaliar

gramáticas, como forma rigorosa de refletir e manusear **linguagens**.

Estrutura do documento

Para cumprir o objetivo organizei o documento em 3 grandes capítulos

- Qualidade das Linguagens – classificação e caracterização;
- Gramaticando as Linguagens – definições sobre gramáticas e processamento de linguagens;
- Qualidade das Gramáticas (GIC e GA) – caracterização e métricas.

Objetivo da apresentação

Nesta apresentação pretendo

- divagar sobre a 1ª Parte: qualidade das linguagens formais;
- justificar e sintetizar a 2ª Parte: qualidade e métricas para gramáticas;
- apresentar a lição aprendida.

Tópicos

Parte I: Linguagens de Programação

- Refletindo sobre Linguagens de Programação
- Classificando as Linguagens de Programação

Parte I: Qualidade das Linguagens de Programação

- Porque Avaliar
- O Que Avaliar
- Tentado Definir
- Caraterísticas que influenciam a Qualidade

Parte II: Qualidade e Métricas para Gramáticas

- Definições
- Qualidade de Gramáticas
- Métricas para Gramáticas

Conclusão

Porque estudar Linguagens de Programação

Citam-se as 6 razões apresentadas por Sebesta em 2009 para se estudar uma Linguagens de Programação:

1. aumentar a capacidade de expressão (escrita) de ideias;
2. melhorar a sabedoria básica para se poder escolher a linguagem apropriada para cada situação;
3. aumentar a capacidade de aprender fácil e rapidamente novas linguagens;
4. aumentar a capacidade de conceber/desenhar novas linguagens;
5. melhorar a percepção da relevância que a implementação tem no uso da linguagem;
6. melhorar a compreensão sobre todo o domínio da programação (resolução de problemas por computador).

LPs e os Conceitos básicos da programação

Os dois conceitos essenciais são:

- **valor** (representado por uma **constante**, **variável** ou **expressão**)
- **operação**.

O primeiro para descrever os *dados* de um problema
e o segundo para realizar as *transformações* que conduzem aos *resultados*.

Linguagens e Paradigmas da programação/comunicação

Dimensões de classificação:

- quanto à cronologia e grau de abstracção;
- quanto ao processamento;
- quanto ao estilo de programação;
- quanto à organização do código;
- quanto à forma de execução;
- quanto ao sistema de tipos;
- quanto à forma de expressão;
- quanto ao propósito, ou âmbito.

Procurando o impossível

Perante tanta diversidade e uma oferta efectivamente grande em cada caso,
é necessário escolher a *melhor linguagem* para cada contexto.

> Porém, tentar decidir de acordo com as vantagens citadas por cada utilizador, é uma discussão interminável, que não leva a qualquer resultado prático . . . . . .

Justifica-se, portanto, que seja relevante discutir com rigor a **qualidade das linguagens**, quer no sentido da *legibilidade*, quer no sentido da *eficiência*.

Legibilidade

No campo da **legibilidade** para o humano (quer no papel de programador que vai escrever frases da linguagem, quer no papel de analista que vai ler essas frases) distinguem-se 3 factores:

- *facilidade de aprendizagem*;
- *facilidade de escrita*, para desenvolvimento de uma descrição completa;
- *facilidade de compreensão*, para se perceber essa descrição completa.

Eficiência

No campo da **eficiência** associada ao seu processamento,
analisa-se apenas a questão do reconhecimento automático das frases,
visto que a questão da execução depende de muitos factores externos à própria linguagem.

Definição de Qualidade de uma Linguagem

A **qualidade de uma linguagem** afere-se em termos da *facilidade* com que se **aprende**, se **usa** e se **compreende** e em termos da *eficiência* com que as suas **frases são processadas**.
Diz-se, então, que uma **linguagem tem qualidade** se *facilita a legibilidade* (numa das três vertentes acima identificadas) *sem degradar o processamento*.

É, então, fundamental encontrar um conjunto de *características*
que permitam um raciocínio claro e objectivo
sobre as propriedades (legibilidade e eficiência) que contribuem para a qualidade das linguagens.

Caracterização de LPs

Foram estudadas várias abordagens, nomeadamente de

- Hoare (73)
- Howatt (95)
- Watt (04)
- Oliveira e Silva (06)
- Sebesta (09)

Caraterísticas que proponho

Características para avaliar a Qualidade de uma Linguagem ou permitir compará-la com outras:

- (CL1) **expressividade**
- (CL2) **documentação**
- (CL3) **unicidade**
- (CL4) **consistência**
- (CL5) **extensibilidade**
- (CL6) **escalabilidade**
- (CL7) **fiabilidade (confiança)**
- (CL8) **modularidade**

Caraterísticas que proponho (cont.1)

De entre as características referidas pelos vários autores citados, foram excluídos da lista anterior os seguintes factores:

- *ambiente / ferramentas de suporte* — apesar de enorme importância para o sucesso de utilização de uma linguagem, é claramente um factor externo ao desenho da linguagem.
- *portabilidade* — tendo também enorme influência na escolha e aplicabilidade de uma linguagem, reduzindo o custo da programação, quer ao nível da escrita, quer da manutenção, depende do compilador (tem essencialmente a ver com o código-máquina gerado).

Caraterísticas que proponho (cont.2)

De entre as características referidas pelos vários autores citados, foram excluídos da lista anterior os seguintes factores:

- *segurança na execução* — depende do compilador, i.é, do código-máquina extra que é gerado para *controlar erros durante a execução* ou do conjunto de condições de prova geradas (extra código-máquina) que permite verificar se o *contrato* inicial é cumprido (Técnica designada por PCC).
- *familiaridade do programador com essa família de linguagens* — de novo um factor que tem grande influência no custo da programação (agora a nível de treino e de escrita), mas que só depende dos hábitos de quem usa a linguagem, sendo independente do seu desenho.

Os 4 factores críticos

Nos próximos diapositivos explicam-se as características enumeradas e mede-se a influência de cada um sobre os **4 factores críticos** para a qualidade de uma linguagem, cf. definição atrás:

- aprendizagem
- escrita
- compreensão
- eficiência no reconhecimento

(CL1) Expressividade

É a *facilidade* e *naturalidade* com que se exprime a mensagem a comunicar, ou seja, a satisfação com que a linguagem permite escrever as instruções a transmitir ou explicitar as operações a realizar.

Além da **disponibilidade dos operadores básicos** necessários para resolver problemas no universo de discurso ao qual a linguagem se aplica, dois outros factores influenciam esta característica:

- **Clareza** – o léxico usado e a sintaxe das operações.
- **Abstracção** – o nível de complexidade dos operadores e operações, em relação ao código-máquina.

(CL1) Expressividade

No caso específico das linguagens de programação, são ainda factores que concorrem para a expressividade:

- A existência de tipos de dados estruturados, nomeadamente o enumerado, o agregado, o conjunto e o ficheiro de acesso directo.
- A possibilidade de manusear listas ligadas dinâmicas sem qualquer referência à alocação de memória, ou aos apontadores.
- A possibilidade de manusear funções-finitas (ou *mappings*), implementadas como tabelas-de-hashing ou de outra forma.

(CL1) Expressividade

A expressividade influencia positivamente a produtividade porque acelera o tempo de

- aprendizagem,
- escrita,
- compreensão.

e não interfere significativamente na

- eficiência do reconhecimento

pois os factores que melhoram a clareza requerem, normalmente, um esforço de análise ligeiramente superior, mas um nível de abstracção elevado não degrada de forma relevante o processo de reconhecimento.

(CL2) Documentação

> É a capacidade de incluir no programa (no texto da mensagem principal) alguma forma de informação extra que explique a própria mensagem.

A forma típica de documentar é através do uso de comentários.
Um **comentário** é uma construção linguística válida que permite inserir programa um texto qualquer
dum determinado ponto até ao fim-de-linha (`eol`),
ou num bloco bem delimitado (envolvendo, ou não, várias linhas).

(CL2) Documentação

Actualmente é frequente que dentro dos comentários, se possa usar uma **linguagem de documentação**.
Essa sub-linguagem permite incluir meta-informação que é usada posteriormente para gerar uma documentação mais sofisticada.

Em contrapartida, várias linguagens actuais obrigam a incluir meta-informação que, além de servir de documentação, é usada pelos processadores.

(CL2) Documentação

No âmbito do desenvolvimento de software segundo a abordagem design by contract,
chama-se *linguagem de anotação de programas* a uma linguagem declarativa (lógica ou algébrica)
que pode ser combinada com uma linguagem de programação para incluir especificações formais nos procedimentos (pré/pós-condições) e nos ciclos (invariantes)
com vista a ser possível provar matematicamente a correcção do código face a uma especificação.

(CL2) Documentação

O facto de uma linguagem permitir denominar todas as entidades descritas na frase através de identificadores longos e formados por um conjunto vasto de caracteres (e não só por letras) é outro factor que influencia positivamente a auto-documentação da frase.

(CL2) Documentação

A documentação complementa a expressividade no sentido de facilitar/acelerar a

- compreensão

e não influencia significativamente a

- aprendizagem

Apesar de ter um efeito negativo na

- escrita

tem um efeito positivo no desenvolvimento porque obriga a reflectir mais profundamente e facilita o retomar da escrita.
Quanto à

- eficiência no reconhecimento

apenas poderá haver uma muito ligeira degradação.

(CL3) Unicidade

> É a existência de uma forma única para escrever a mesma instrução ou operação.

A unicidade é, como se costuma dizer, *um pau de dois bicos*. Por um lado é, em todos os sentidos, mais simples de trabalhar quando não há escolhas, quando só há um caminho para cumprir determinado objectivo.

Por outro lado é, sem dúvida, *anti-natura*; o ser humano (por muito que se queixe) gosta sempre de poder escolher e tomar as suas opções, de modo a personalizar a sua actividade e senti-la mais adaptada, mais sua...

(CL3) Unicidade

Positivamente, a unicidade

- acelera o tempo de aprendizagem
- acelera o tempo de reconhecimento (aumenta a sua eficiência)
- facilita a compreensão

Mas interfere negativamente na

- escrita

pois não permite que cada um procure a forma que mais lhe agrada para exprimir suas ideias.
Vulgarmente, a unicidade é conseguida à custa de operações mais elementares e portanto acaba por se traduzir numa frase mais longa para dizer a mesma coisa.

(CL4) Consistência

> É a conservação da estratégia, da disciplina ou princípio de trabalho, para expressão dos conceitos análogos (da mesma família), ou afins.

Uma linguagem revela *consistência* se a forma de escrever o mesmo tipo de ideias é coerente, é sempre a mesma.

(CL4) Consistência

A consistência está relacionada com o historial da linguagem e a forma como evoluiu:
uma das maiores fontes de inconsistência é ser *estendida/completada* quando já em uso, para a adaptar a novos fins ou aumentar a expressividade.

Pode dizer-se que a consistência é tanto maior quanto mais maior for a sua **fidelidade a um paradigma** de trabalho.
Esta é uma das características mais importantes—ser muito evidente na linguagem a existência de um princípio forte de trabalho subjacente, ou seja, a adesão clara a uma determinada conduta com suas estratégias e respectiva disciplina.

(CL4) Consistência / Ortogonalidade

Inclui-se aqui uma característica distinta mas afim designada por **ortogonalidade**.

> Uma linguagem diz-se **ortogonal** se o conjunto de operadores pode ser aplicado a qualquer um dos tipos de dados disponíveis.

Ao nível do Assembly (âmbito em que surgiu) significa que os diversos modos de endereçamento de memória (para acesso aos operandos) podem ser usados com qualquer um dos operadores definidos.
Esta característica facilita consideravelmente a aprendizagem porque reduz muito os casos especiais.

(CL4) Consistência

A consistência e a ortogonalidade influenciam positivamente a produtividade porque aceleram o tempo de

- aprendizagem
- escrita
- compreensão

e não interferem de todo na

- eficiência do reconhecimento

(CL5) Extensibilidade

É a capacidade que uma linguagem oferece para ser modificada (acrescentada) pelo utilizador.

Este conceito está fortemente associada ao uso de *macros*.

Uma **macro** é uma regra que define como um texto de entrada sucinto (mnemónico) é expandido para outro texto de saída (fixo ou variável) mais longo ou complexo.
Essa expansão pode ser textual ou pode ser procedimental.

(CL5) Extensibilidade

Com o intuito de aliviar a tarefa de escrita (reduzindo o tempo e as hipóteses de erro),
O conceito surgiu pelos anos 60/70 no âmbito da programação em Assembly,
e também no contexto do C, LaTeX, ou dos utilitários do MS-Office,
Mas só ressurge em força por volta de 2000, muito associado à programação para a Web.

(CL5) Extensibilidade

Importa distinguir:

extensões estáticas que alargam a linguagem, durante a compilação, permitindo que se usem novos elementos ao traduzir uma dada frase (um programa)

extensões dinâmicas que permitem modificar um programa, graças a ser possível retirar ou juntar código, o qual é compilado só em *runtime*.

(CL5) Extensibilidade Dinâmica

É essencialmente uma extensão/alteração ao código durante a execução. Mas requer que a linguagem tenha primitivas para manusear essas facilidades:

- criar ou remover código (tipo `assert` e `retract`),
- modificar objectos,
- invocar o compilador na fase de execução (tipo `eval`, ou `call`).

Surge frequentemente relacionada com as *linguagens reflexivas*.

> A **reflexão** é a capacidade oferecida por certos linguagens de inspeccionar, eventualmente modificar, o comportamento de um programa durante a sua execução, quer em relação ao valor das estruturas de dados, quer em relação ao código.

(CL5) Extensibilidade Estática

No caso da extensibilidade estática, importa distinguir três formas:

Léxica acrescentando tipos e operações (procedimentos/funções), estende-se o alfabeto;

Sintática acrescentando novas construções frásicas – implica alterar a própria gramática (com novos NT e P) e reconsultá-la cada vez que vai processar uma nova frase[1];

Semântica acrescentando ou refinando/especializando o significado das construções e operações – disponível nas LOO, é assegurada pelos mecanismos de extensão associados à hierarquia de classes.

[1] Interessante desafio para o processamento de linguagens, mas sem grande relevância prática.

(CL5) Extensibilidade

A extensibilidade acelera o tempo de

- escrita

porque aumenta a abstracção e, portanto, a expressividade, permitindo adaptar a ferramenta (a linguagem) às mãos de quem trabalha com ela. Mas, claramente, tem efeito negativo na

- aprendizagem
- eficiência do reconhecimento

Quanto à

- compreensão

não é seguro afirmar se facilita ou complica. No início, dificulta mas depois pode facilitar, contudo depende muito da qualidade dessas extensões.

(CL6) Escalabilidade

> É a capacidade que uma linguagem demonstra de servir para lidar uniformemente—mantendo as suas características e qualidade—com problemas sucessivamente mais complexos.

Um sistema diz-se *escalável* quando está preparado para crescer (mudar de escala da carga) mantendo-se estável e exibindo um comportamento constante, sem degradar a sua performance.

No caso de uma linguagem, diz-se que é *escalável* se permite escrever, compreender ou processar com a mesma facilidade frases longas ou frases curtas, ou seja, se permite trabalhar igualmente bem com problemas de grande dimensão/complexidade.

(CL6) Escalabilidade

As *linguagens visuais* são um bom contra-exemplo de linguagem escalável; linguagens declarativas também não escalam bem.

Um fator que muito influencia a escalabilidade é a *verbosidade* (nos 2 sentidos).

Outro fator crítico é a política de tipos e os mecanismos de parametrização dos subprogramas.

(CL6) Escalabilidade

A escalabilidade não degrada

- eficiência do reconhecimento

e preserva a legibilidade, ou seja, não altera a facilidade de

- aprendizagem
- escrita
- compreensão

(CL7) Fiabilidade (Reliability)

É a *confiança* que se pode ter que o código que vai ser executado *realiza* as operações especificadas no texto-fonte *como esperado* e que se *comporta corretamente* face a situações *anómalas*.

A primeira faceta que contribui para a fiabilidade é *a linguagem ter uma semântica precisa e não ambígua, muito claramente definida*.

Infelizmente, na grande maioria dos casos a semântica é descrita em linguagem natural em vez de ser definida formalmente, dificultando a aprendizagem e retirando a confiança quanto ao que se passa na execução.

(CL7) Fiabilidade

Outra das facetas importantes para uma linguagem ser fiável é *ter uma politica de tipos rígida*!
obrigando a que todos os valores estejam associadas um tipo
e *fazendo uma análise de tipos estática* completa.

A existência de um *sistema de inferência de tipos* poderoso (à la ML ou Haskell)
é muito relevante porque permite manter uma análise de tipos severa, sem obrigar a declarar explicitamente todos os tipos.

(CL7) Fiabilidade

A este propósito é interessante mencionar que no seio da comunidade da Programação Orientada-a-Objectos, há uma discussão acesa entre os defensores das políticas de análise estática de tipos—que argumentam que é muito mais seguro e aumenta a eficiência durante a execução e os defensores da análise de tipos dinâmica—que sustentam a ideia que o aumento de flexibilidade do programa é de tal forma relevante para o custo de desenvolvimento que ultrapassa a perda de eficiência.

(CL7) Fiabilidade

Outra faceta relevante é a
existência de mecanismos da linguagem para o *controlo de situações de exceção durante a execução*.
Para o programador é muito mais confortável poder estabelecer como quer contornar os regimes excepcionais do que os deixar sob o controlo do sistema operativo.

(CL7) Fiabilidade

Naturalmente, a fiabilidade

- aumenta o tempo de processamento (diminuindo a sua eficiência)
- aumenta o tempo de escrita

mas facilita a

- compreensão

quanto ao

- tempo de aprendizagem

é dúbio. Se por um lado, ter uma semântica clara e precisa facilita a aprendizagem, ter um sistema de tipos forte e rígido e mecanismos para controlo e excepções implica ter mais coisas para estudar e aprender e portanto aumenta o tempo desta fase.

(CL8) Modularidade

> É a facilidade de se poder organizar partes da mensagem (instruções/tarefas afins) em componentes individuais (distintas e independentes) que sejam depois reutilizadas.

A necessidade de agrupar o código de um programa surgiu desde o início da programação (Assembly) assim que a complexidade dos problemas aumentou e começou a exigir a escrita de frases (mais longas), por duas fortes razões:

- facilitar a escrita, evitando repetir sub-frases semelhantes ($\Rightarrow$ conceito de reutilização);
- isolar sub-frases que descrevem uma determinada ideia importante ($\Rightarrow$ conceito de sub-programa).

(CL8) Modularidade

Partindo do conceito primitivo de **subrotina**, em Assembly,
evolui depois para a forma de **procedimento** ou **função**, p. ex. em Pascal,
e tomou a verdadeira forma de **módulo** em linguagens como Modula-2
onde surge associado aos conceitos de *interface* e *implementação*,
para separar a *forma como se reutiliza* da *forma como se definem as estruturas de dados* ou *se descreve o corpo dos procedimentos/funções*.

(CL8) Modularidade

Posteriormente, em linguagens como SmallTalk, Eiffel, Co, Java, C♯, este conceito de módulo foi apurado no conceito de **classe** mantendo a separação interface/implementação, mas providenciando uma metodologia de desenvolvimento baseada na noção de objeto que agrega e encapsula dados e respectivas operações.

(CL8) Modularidade

Positivamente, a modularidade

- acelera o tempo de escrita
- facilita a compreensão

pois permite estruturar melhor o código, concentrando em determinadas partes determinadas ideias, e permite reutilizar.
E interfere muito ligeiramente de forma negativa na

- aprendizagem (há mais alguns construtores a aprender)
- no tempo de reconhecimento (diminui a eficiência porque há mais componentes a processar e mais informação manter em memória e a gerir)

Síntese

Tabela-resumo da influência das 8 características escolhidas sobre os 4 fatores para avaliar a qualidade de uma linguagem.

Caract x Fatores	Aprendiz.	Escrita	Compreensão	Reconhec.
CL1:Expressividade	$+$	$+$	$+$	$-$
CL2:Documentação	X	$-$	$+$	$-$
CL3:Unicidade	$+$	$-$	$+$	$+$
CL4:Consistência	$+$	$+$	$+$	X
CL5:Extensibilidade	$-$	$+$	$+/-$	$-$
CL6:Escalabilidade	$=$	$=$	$=$	$=$
CL7:Fiabilidade	X	$+$	X	$-$
CL8:Modularidade	X	$+$	$+$	$-$

Síntese

Analisando esta Tabela de Síntese, pode-se postular que:

> A qualidade de uma linguagem, entendida como expresso na definição inicial, pode ser decidida atestando a presença das características CL1 a CL8, sendo a linguagem tanto melhor quanto mais características verificar.

Exemplo de Aplicação

Tabela que exemplifica a aferição de 4 Linguagens de Programação atendendo às 8 caraterísticas estudadas.

Caract x Ling.s	Pascal	C	Prolog	Java
Paradigma	Imp	Imp	Decl/L	OO/Imp
CL1:Expressividade	B	S	S	MB
CL2:Documentação	sm	S	sm	MB
CL3:Unicidade	N	N	N	N
CL4:Consistência	B	S	B	B
CL5:Extensibilidade	S	B	B	MB
CL6:Escalabilidade	S	B	NS	NS
CL7:Fiabilidade	B	S	sm	MB
CL8:Modularidade	sm	S	sm	MB

Exemplo de Aplicação

Analisando diretamente a Tabela, a 1ª ilação direta que se tira é que:

> Java é de todas as linguagens analisadas, a melhor.
> a que tem mais características melhor pontuadas!

Exemplo de Aplicação

A 2ª ilação que se pode tirar é relativa à clássica comparação Pascal/ C:

> Tal como a intuição nos diz, estas duas linguagens são muito complementares....

Exemplo de Aplicação

Atribuindo pesos numéricos às pontuações usadas (NS=-1; N=0; ... MB=4)
e considerando o impacto positivo (1) ou negativo (-1) que cada caraterísticas tem em cada fator,
pode-se quantificar a qualidade da linguagem em cada uma das 4 vertentes.

Java é a linguagem mais fácil de aprender e compreender e a mais cómoda para desenvolver;

Pascal e C tem cotações muito semelhantes em termos de compreensão e de reconhecimento;
mas Pascal tem melhor cotação em termos de aprendizagem e facilidade de desenvolvimento.

Conclusão da Parte I

Manualmente é fácil constatar a presença dessas características.
A presença das características CL1 a CL8 (não mensuráveis) pode ser apurada através de ensaios cujos resultados são recolhidos em inquéritos e depois tabelados.

A questão está agora em tornar mais objetiva a pontuação de cada caraterística da linguagem em estudo.

Conclusão da Parte I

O problema interessante que aqui se levanta é saber como conseguir fazê-lo de forma mais objectiva e automática ?

Em resposta a esta questão, o que se discute a seguir é
até que ponto a qualidade de uma gramática pode ser medida automaticamente e
pode ajudar a avaliar a qualidade da linguagem que define.

Gramáticas e Linguagens

Gramaticar uma linguagem significa dar-lhe **forma** e **sentido** através de uma notação rigorosa para descrever as regras sintácticas e semânticas.

Para dar estrutura à linguagem, isto é, definir o seu vocabulário e as formas como os símbolos se podem agrupar para exprimir cada ideia, defende-se o uso de uma Gramática Independente de Contexto (GIC).

Para dar significado às frases e restringir algumas construções sintácticas que não fazem sentido,
defende-se o uso de uma Gramática de Atributos (GA).

Gramáticas e Linguagens

As linguagens são, no âmbito da compilação e do processamento automático, formalmente especificadas por gramáticas.

É, pois, expectável que a qualidade da primeira dependa da qualidade da segunda,
mesmo sendo certo que a qualidade de uma especificação ou a qualidade de uma ferramenta
nunca podem implicar totalmente a qualidade do produto final.

Gramática Independente de Contexto

Uma Gramática Independente de Contexto define-se como sendo um tuplo

$$GIC =< T, N, S, P >$$

onde
T é o conjunto dos **símbolos terminais** formado pelas **Palavras-Reservadas**, os **Sinais** e os **Terminais-Variáveis**.
N é o conjunto dos **símbolos não-terminais** da gramática.
$S \in N$ é o **símbolo inicial** ou *axioma* da gramática.
P é o **conjunto de produções** ou *regras de derivação* da gramática.
Cada produção $p \in P$ é uma regra da forma

$$p : X_0 \rightarrow X_1 \ldots X_i \ldots X_n$$

Gramática Tradutora

Uma Gramática Tradutora define-se como sendo um tuplo

$$GT =< T, N, S, AS, Px >$$

onde
Cada produção $p \in Px$ passa agora a ser uma regra da forma

$$p : X_0 \rightarrow X_1 \ldots X_i \ldots X_n \ as$$

em que $as \in AS$ indica a Acção Semântica a executar após reconhecer todos os símbolos no lado direito da produção.

Gramática de Atributos

Uma Gramática de Atributos define-se como sendo um tuplo

$$GA =< GIC, A, RC, CC, RT >$$

onde
A é o conjunto dos **atributos** de todos os símbolos da gramática.
RC é o conjunto das **regras de cálculo dos atributos**.
CC é o conjunto das **condições de contexto**.
RT é o conjunto das **regras de tradução**.
Os atributos $A(X)$ de cada símbolo dividem-se em dois subconjuntos disjuntos:
$AI(X)$ é o conjunto dos **atributos herdados** do símbolo X.
$AS(X)$ é o conjunto dos **atributos sintetizados** do símbolo X.

Outras Definições Relevantes

- Produção Unitária
- GIC bem-formada
- Símbolo Terminável
- GDS
- Árvore de Derivação
- Frase Válida
- Linguagem Não-Ambigua
- GA bem-formada

Funções de uma Gramática

Uma gramática desempenha um duplo papel:

- define (ou gera) uma linguagem e, portanto, diz como se escrevem frases válidas.
- guia o reconhecimento das frases dessa linguagem que gera, sendo esta vertente usada:
  - pelo humano que tem de compreender as frases (retirar o significado),
  - pelo programa que faz o processamento (reconhecimento e transformação) dessas frases.

Assim estudo sobre a qualidade será organizado segundo estas duas óticas:

- gramática como geradora de linguagens;
- gramática como geradora de programas

Caraterização de Gramáticas

Características para avaliar a qualidade de uma Gramática como *geradora de linguagens*:

- **(CG1) usabilidade** da gramática enquanto instrumento para derivar frases de uma linguagem:
  - (CG1.1) facilidade de compreensão
  - (CG1.2) facilidade de derivação
  - (CG1.3) facilidade de manutenção

Caraterização de Gramáticas

Características para avaliar a qualidade de uma Gramática como *geradora de programas*:

- **(CG2) eficiência** da gramática enquanto instrumento para derivar processadores para uma linguagem:
  - (CG2.1) eficiência no reconhecimento (processamento) das frases da linguagem gerada.
  - (CG2.2) eficiência na geração automática do processador.

Definição de Qualidade de uma Gramática (1)

A **qualidade de uma gramática, enquanto especificação que gera uma linguagem**

afere-se em termos da facilidade com que

- se **aprende** (lê e compreende o que ela descreve)
- se **usa para derivar** frases
- se **mantém** (correctiva ou evolutivamente)

Neste sentido, diz-se que uma *gramática tem qualidade* se facilita a usabilidade.

Definição de Qualidade de uma Gramática (2)

A **qualidade de uma gramática, enquanto especificação que gera um processador**

afere-se em termos da

- **eficiência** do programa que dela deriva.
- **eficiência** do próprio processo de geração.

Neste sentido, diz-se, que uma *gramática tem qualidade* se permite gerar processadores de linguagens eficientes sem degradar a facilidade de geração automática.

Qualidade de uma GIC *(1)*

Influência dos Elementos de uma GIC na Usabilidade

Elems x Caracts	(CG1)Usabilidade		
	Compreensão	Derivação	Manutenção
Ids Símbolos claros	+	+	+
Prods Unitárias	+	−	+
Comprimento RHS	−	+	−
Notação	+/−	−p, +ex	−p, +/−ex
Esquema Recursivo	+/−	+d, −e	+/−
Modularidade	−	−	+
Complexidade Sint	*X*	*X*	−

Qualidade de uma GIC *(2)*

Influência dos Elementos de uma GIC na Eficiência

Elems x Caracts	(CG2)Eficiência	
	Reconh-L	Geraç-Rec
Ids Símbolos claros	*X*	+Te,Ta
Prods Unitárias	+Te,Ta	+Te,Ta
Comprimento RHS	*x*Te,Ta	*X*
Notação	*X*	*X*
Esquema Recursivo	−Te,Ta	*X*
Modularidade	*X*	+Te
Complexidade Sint	*X*	*X*

Qualidade de uma GA *(1)*

Influência dos Elementos de uma GA na Usabilidade

Elems x Caracts	(CG1)Usabilidade		
	Compreensão	Derivação	Manutenção
Ids Atributos claros	+	X	+
Nu°Atributos	√	X	−
Nu°Opers Atribs	√	X	−
Nu°Simbolos+Prods	√	−	−
Complexidade Atribs	−	x	−
Complexidade OpersAts	−	x	−
Colocação/clarez CCs	+	+	+
Notação	+p, −ex	x	+p, −ex
Esquema Recursivo	+/−	+/−	+/−
Modularidade	−	−	+
Complexidade Semant	X	X	−

Qualidade de uma GA *(2)*

Influência dos Elementos de uma GA na Eficiência

Elems x Caracts	(CG2)Eficiência	
	Reconh-L	Geraç-Rec
Ids Atributos claros	X	$+$Te,Ta
Nu°Atributos	xTe,Ta	$+$Te,Ta
Nu°Opers Atribs	$+$Te	$+$Te,Ta
Nu°Simbolos+Prods	xTe,Ta	$+$Te,Ta
Complexidade Atribs	xTe,Ta	X
Complexidade OpersAts	xTe,Ta	X
Colocação/clarez CCs	X	X
Notação	X	X
Esquema Recursivo	$-$Te,Ta	X
Modularidade	X	$+$Te
Complexidade Semant	X	$+$Te,Ta

Métricas para GIC

- Métricas de Tamanho:
  - (MT1) **tamanho da Gramática**, mede-se em termos de: #T, #N, #P, #PU, #Rec, §RHS, §Alt, #Mod
  - (MT2) **complexidade sintáctica da Gramática**, mede-se usando: FanIn, FanOut
  - (MT3) **tamanho do Parser**, mede-se em termos de: #RD, §TabLL, §$\mathcal{AD}$-LR + §TabsLR

Métricas para GIC

- Métricas de Forma:
  - (MF1) **forma de Recursividade**, podendo ser: RecDirecta, RecIndirecta
  - (MF2) **tipo de Recursividade**, podendo ser: RecD, RecD-LL, RecE
  - (MF3) **notação**, podendo ser: BNF, ex-BNF

Métricas para GIC

- Métricas Lexicográficas:
  - (ML1) **identificadores esclarecedores** para os símbolos não-terminais e terminais-variáveis

$$\#IdCompl/(\#IdCompl + \#IdAbrev)$$

  - (ML2) **palavras-reservadas e sinais esclarecedores** da linguagem definida por G

$$\#PRCompl/(\#PRCompl + \#PRAbrev)$$

  - (ML3) **flexibilidade dos terminais-variáveis** da linguagem definida por G

$$\#TFlex/(\#TFlex + \#TRigid)$$

  - (ML4) **tipo de comentários**,

$$inline + bloco + metaI$$

Derivação de um Identificador

Para completar a formulação das métricas lexicográficas é necessário dizer o que se entende por uma palavra *derivar* de um termo:

Diz-se que um Identificador **deriva** do Nome de um Conceito se:

1. o identificador for igual ao nome (p.ex., `Lisp` deriva de Lisp);
2. o identificador for um prefixo do nome, com 3 ou mais letras (p.ex., `num` deriva de numero);
3. o identificador tiver um prefixo que é prefixo do nome e as restantes letras poderem ser obtidas do nome por remoção de algumas letras do nome (p.ex., `SExp` deriva de Symbolic Expression).

Um Identificador **denota-bem** uma Operação, se esse identificador for formado por 1 ou mais sinais que na vida quotidiana se usam para representar a operação em causa.

Métricas para GIC

Relação das Métricas Sintáticas com as Características para aferição duma GIC e da $\mathcal{L}_{GIC}$

Metrica x Caract	CG1.1	CG1.2	CG1.3	CG2.1	CG2.2	QL
MT1	√	√	√	√	√	√
MT2	√		√			
MT3				√	√	√
MF1	√	√	√			√
MF2	√	√		√		√
MF3	√	√	√			√
ML1	√	√	√		√	√
ML2				√		√
ML3				√		√
ML4				√		√

Métricas para GIC

As métricas lexicográficas, ML1 a ML4, têm uma influência mais forte e objectiva na qualidade da linguagem gerada por G.
A interferência de ML1 na legibilidade da linguagem continua a ser por contribuir para uma maior facilidade de aprendizagem;
porém a influência de ML2, ML3 e ML4 é devida à implicação que estes parâmetros têm em algumas das características de aferição de linguagem:

ML x Caract	CL1	CL2	CL6	QL
IdSimbolos				+L,*X*R
PalReservadas	+		−	+L,*x*R
SVariaveis	+		−	+L,*x*R
Comentarios		+		+L,*x*R

Métricas para GA

- Métricas de Tamanho:
  - (MTA1) tamanho de GA:
    #A, #AI, #AS, #RC, #CC, #RT
  - (MTA2) complexidade semântica da gramática:
    FanIn, FanOut

Métricas para GA

- Métricas de Forma (1):
  - (MFA1) **complexidade dos atributos**,

$$\#AComplex/(\#AComplex + \#AAtom)$$

  - (MFA2) **complexidade das operações atributivas**,

$$\#OComplex/(\#OComplex + \#OSimples)$$

  - (MFA3) **esquema de cálculo** para escrita das RCs,
    sendo medido pela adição de dois valores, um que avalia a **forma de agregação** (RCAgreg, RCNAgreg)
    e outro que classifica a forma de acumulação de valores recursivos (RCpureS, RCpureI, RCmixIS, RCvar)

Métricas para GA

- Métricas de Forma (2):
  - (MFA4) **esquema de restrição semântica** para escrita das CCs, CCTop, CCCentrad, CCBottom, CCvar
  - (MFA5) **esquema de tradução** para escrita das RTs, RTTop, RTInterm, RTBottom, RTvar
  - (MFA6) **estilo da linguagem** para escrita das operações atributivas, *linguagem declarativa* (funcional ou lógica), ou não (se for imperativa).
  - (MFA7) **especificidade da linguagem** para escrita das operações atributivas,
    linguagem de programação standard, comum, ou não

Métricas para GA

- Métricas Lexicográficas:
  - (MLA1) **identificadores esclarecedores** para os atributos,

$$\#IdACompl/(\#IdACompl + \#IdAAbrev)$$

  - (MLA2) **identificadores esclarecedores** para operadores atributivos (nomes das funções, predicados e procedimentos invocados nas regras de cálculo dos atributos, nas condições de contexto e nas regras de tradução),

$$\#IdOCompl/(\#IdOCompl + \#IdOAbrev)$$

Métricas para GA

Relação da Métricas Atributivas com as Características para aferição duma GA e da $\mathcal{L}_{GA}$

Métrica x Caract	CG1.1	CG1.2	CG1.3	CG2.1	CG2.2	QL
MTA1	√	√	√	√	√	√
MTA2	√		√	√	√	√
MFA1	√	√	√	√		√
MFA2	√	√	√			√
MFA3	√	√	√	√	√	√
MFA4	√	√	√	√	√	√
MFA5	√		√	√	√	√
MFA6	√	√	√	√		√
MFA7	√	√	√	√	√	√
MLA1	√	√	√		√	√
MLA2	√	√	√		√	√

Sumário

Com o intuito de perceber em que consistia a noção de qualidade no âmbito das linguagens (de programação) formais e o que contribuia para ela

- foram revistos os principais conceitos associados a LPs, sua evolução e classificação;
- foi proposta uma definição para qualidade de uma linguagem;
- foram identificados os fatores que contribuiam para a qualidade;
- foram estudadas as caraterísticas da linguagem que tinham impacto nesses fatores;
- foram revistas as gramáticas (GIC e GA) e o seu uso na definição de linguagens;
- foi proposta uma definição para qualidade de uma gramática;
- foram propostas métricas e a forma de as usar para avaliar a qualidade de uma gramática.

A lição retirada

As principais evidências que resultam deste estudo são:

- as caraterísticas duma linguagem que impactam nos fatores de qualidade podem ser avaliadas (ainda que manualmente) e permitem refletir sobre a qualidade das linguagens, em termos absolutos ou comparativos;
- de forma análoga, os elementos duma gramática que impactam nos fatores de qualidade podem ser avaliadas *através do cálculo (quasi-automático) de métricas* e permitem refletir sobre a qualidade das gramáticas, em termos absolutos ou comparativos;
- as métricas propostas para avaliar a qualidade de uma gramática dão também informação sobre a qualidade da linguagem definida pela gramática, embora não total.