ENG. INFORMÁTICA

COMPILADORES

COMPILADORES

ENG. INFORMÁTICA

Capítulo 1: Introdução

- **Processamento de uma linguagem**
    - Definição de linguagem, definição de alfabeto, definição de palavra, definição da linguagem em extensão/compreensão, regras léxicas e regras sintáticas
    - Linguagens actuais e linguagens formais, como se especifica uma linguagem?
    - Análise e síntese; Fases de reconhecimento
    - Exemplos de processadores de linguagem; Tarefas de um processador de linguagem; Estratégias de processamento
    - Compiladores/interpretadores
- **Génese dos Compiladores/Compiladores**
    - um pouco de história: Como tornar a programação mais produtiva? FORTRAN, exemplo de análise léxica, parsing e análise semântica. Tarefas de um tradutor (no passado e no presente), o contexto de um compilador (pré-processador, compilador, assembler, loader/linker), fases de compilação (ou reconhecimento)
- **A economia das linguagens de programação**
    - Porque existem tantas linguagens de programação?
    - Porque aparecem novas linguagens de programação?
    - Quais são as boas linguagens de programação?
- **Programas tipo compilador**

CAPÍTULO 1

PROCESSAMENTO DE UMA LINGUAGEM

O que é uma linguagem ?

Definição [Linguagem] Seja $\mathscr{V}$ um vocabulário/alfabeto.

Uma linguagem $\mathscr{L}$ sobre $\mathscr{V}$ é um conjunto de palavras em que cada palavra é uma sequência de símbolos pertencentes a $\mathscr{V}$.

Definição [Alfabeto] Um conjunto finito e não vazio de símbolos.

Os símbolos do alfabeto são designados por **letras**.

Exemplo:

- Os alfabetos das linguagens naturais: alfabeto latino e alfabeto grego
- ASCII, EBCDIC e UNIcode

CAPÍTULO 1

PROCESSAMENTO DE UMA LINGUAGEM

Definição [palavra] sequência finita de letras.

$\mathscr{V}*$ representa o conjunto de todas as palavras sobre $\mathscr{V}$

ε representa a palavra vazia — não contém letras; $\varepsilon \in \mathscr{V}*$

$v \in \mathscr{V}*$, $|v|$=comprimento de v (o nr de símbolos);

e.g. $|abc| = 3$ e $|\varepsilon| = 0$

Uma linguagem não admite todas as sequências de letras: existem sequências válidas e inválidas.

CAPÍTULO 1

PROCESSAMENTO DE UMA LINGUAGEM

Todas as palavras são finitas, mas a linguagem é frequentemente um conjunto infinito, tornando impossível a sua enumeração.

Definição de uma linguagem

- por enumeração; $\{a, b, aa, ab, ba, bb\}$
- por compreensão; $\left\{v : |v| \leq 2 \text{ e } v \in \mathscr{V} = \{a, b\}\right\}$

Uma questão de linguagem: palavra, frase, string, token ...

CAPÍTULO 1

PROCESSAMENTO DE UMA LINGUAGEM

Para definir uma linguagem, indicamos:

- o alfabeto
- as regras para formar sequências válidas

As regras que definem a estrutura são de dois tipos:

- Regras léxicas: definem como se formam palavras correctas
- Regras sintáticas: definem como se formam frases correctas
- Regras semânticas: definem o significado das frases — regras que devem ser respeitadas pelos símbolos para que as frases façam sentido, i.e., para que as frases possam ser interpretadas

CAPÍTULO 1

PROCESSAMENTO DE UMA LINGUAGEM

estrutura das frases: que símbolos devem ser usados e a ordem pela qual devem ser escritos

significado das frases: valor intrínseco dos símbolos, ou inferível a partir dele; indica-se como são interpretadas as frases

significado: informação contida numa frase — o que efectivamente nos interessa

linguagens naturais: línguas faladas pelo diferentes povos no dia a dia, português, inglês, hindu, magyar, basco, …

linguagens artificiais/formais: linguagens criadas com o propósito de veicular a comunicação homem-máquina — linguagens de programação: python, c, Matlab, Java, …

CAPÍTULO 1

PROCESSAMENTO DE UMA LINGUAGEM

O formalismo das linguagens formais é muito mais rigoroso ⟶não podem existir ambiguidades

Como se especifica uma linguagem ?

As linguagens formais têm de ter a sua sintaxe e semântica rigorosamente definidas — à custa de regras adequadas

Estas regras devem ser escrita de forma sucinta, objectiva e sem ambiguidades, de forma a que possam ser facilmente interpretadas por ambos os agentes de comunicação: emissor e receptor

emissor = utilizador humano

receptor = computador

CAPÍTULO 1

PROCESSAMENTO DE UMA LINGUAGEM

A gramática é universalmente aceite como instrumento de definição de uma linguagem; serve o duplo propósito de:

- ensinar como se escrevem, ou produzem, as frases da linguagem ⟶ papel reconhecedor
- determinar como se podem analisar essas frases ⟶ papel gerador

$\mathscr{L}_G$ linguagem especificada pela gramática G

CAPÍTULO 1

PROCESSAMENTO DE UMA LINGUAGEM

Definição [Processamento de uma linguagem] Seja $\mathscr{L}_G$ **a** linguagem gerada pela gramática G. Um processador para essa linguagem, $P_{\mathscr{L}_G}$, é um programa que, tendo conhecimento da gramática G:

- lê um texto — sequência de símbolos
- verifica se esse texto é uma frase válida de $\mathscr{L}_G$
- executa uma acção em função do significado da frase reconhecida

Existe uma larga classe de programas que cumprem esta definição. Mas todos eles são constituídos por 2 módulos:

Análise: faz o reconhecimento do significado do texto fonte

Síntese: reage ao significado identificado produzindo um determinado resultado

CAPÍTULO 1

PROCESSAMENTO DE UMA LINGUAGEM

Exemplos de processadores de linguagem:

Assemblers: traduzem linguagens de programação de baixo nível, formadas por mnemónicas das instruções, para código máquina

Compiladores: traduzem linguagens de programação de alto nível para código máquina

Interpretadores: executam os programas logo após o seu reconhecimento; sem gerarem um programa executável em código máquina

Tradutores (em geral): transformam textos escritos numa qq linguagem para uma outra linguagem

CAPÍTULO 1

PROCESSAMENTO DE UMA LINGUAGEM

Carregadores: reconhecem descrições de dados e "carregam" essa informação para Base de Dados ou para estruturas de dados em memória central

Pesquisadores: reconhecem questões *query* — pesquisam em BD para encontrar as respostas encontradas

Filtros: reproduzem como saída o texto de entrada depois de lhe terem sido retiradas palavras (ou substituição de abreviaturas) ou, por ex., as minúsculas são convertidas em maiúsculas. ex:

- pré-processador de C e o par C Tangle
- Web

CAPÍTULO 1

PROCESSAMENTO DE UMA LINGUAGEM

Processadores de documentos: usados em manipulação vária de documentos, como por exemplo na formatação; ex. Latex, Bibtex, ou extracção de conhecimento

Formatadores (*pretty printers*): recebem um texto em linguagem objecto e produzem o mesmo texto formatado, por exemplo realçando os principais construtores da linguagem.

Cronologicamente, os assembladores foram os primeiros a surgir, a que se seguiram os compiladores/interpretadores

CAPÍTULO 1

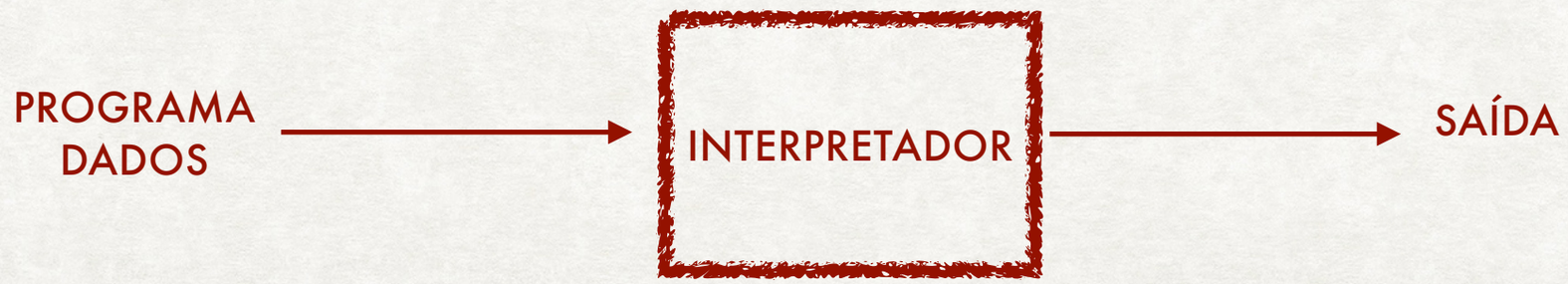
PROCESSAMENTO DE UMA LINGUAGEM

2 abordagens distintas para implementar/processar uma linguagem de programação:

- compiladores
- interpretadores

interpretador: não requer qualquer processamento do programa antes da sua execução ⟶ correr o programa nos dados

O interpretador funciona *online*, no sentido em que o processador do programa faz parte integrante de correr o programa



CAPÍTULO 1

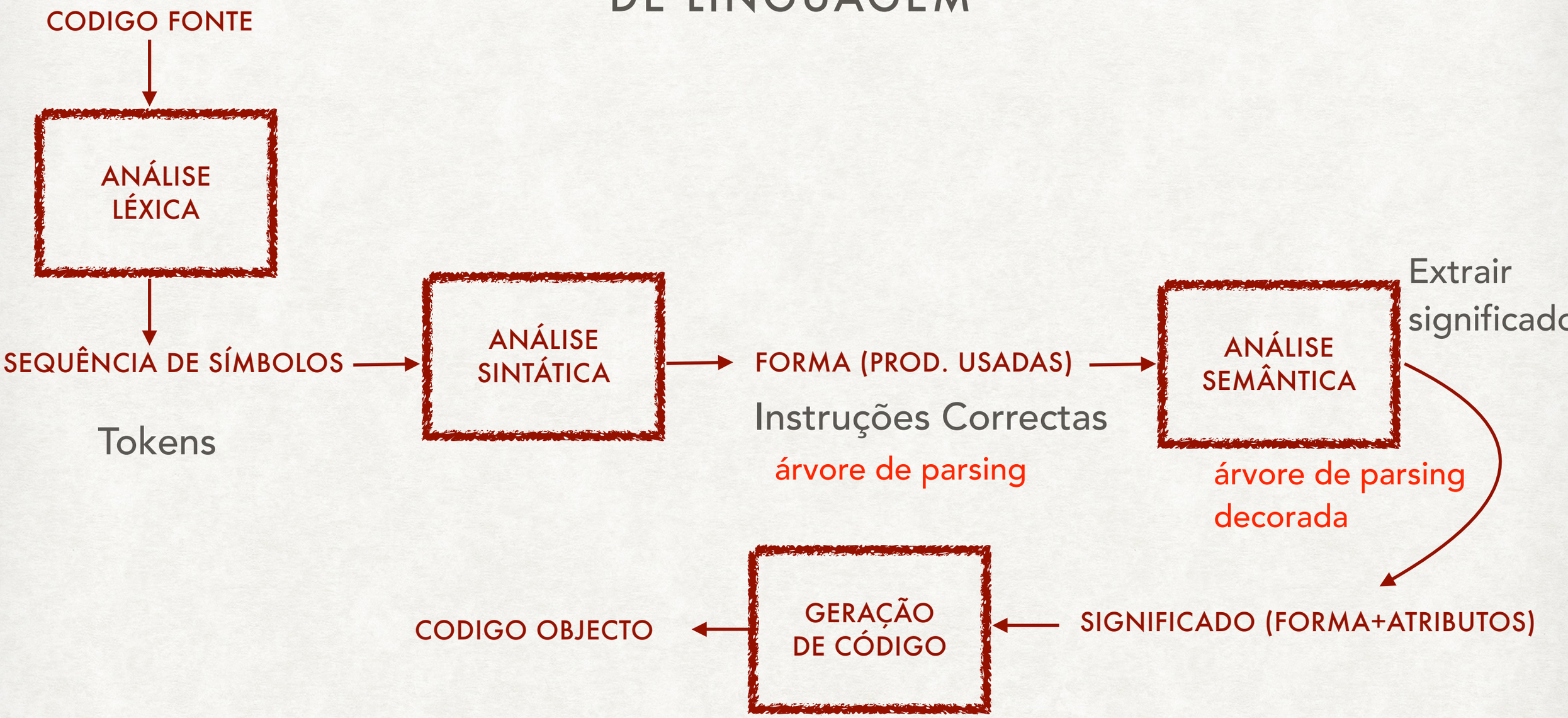
PROCESSAMENTO DE UMA LINGUAGEM

compilador: pode ser escrito em diferentes linguagens de programação; uma vez gerado o executável, *exec* é invocado nos dados e produz a saída

O compilador funciona *offline*, no sentido em que o programa é primeiro pré-processado, antes de ser aplicado nos dados e produzir a saída; o compilador é somente um pré-processador de código que produz um código executável passível de ser executado com diferentes dados

CAPÍTULO 1

TAREFAS DE UM PROCESSADOR DE LINGUAGEM



CAPÍTULO 1

FASES DE RECONHECIMENTO

Análise léxica (AL): responsável pela leitura sequencial dos caracteres que forma o texto fonte, pela sua separação em palavras e pelo reconhecimento dos vocábulos (lexemas, tokens) — símbolos terminais — representados por cada palavra

Análise sintática (AS): encarregado de agrupar os símbolos terminais verificando se forma uma frase sintaticamente correcta, i.e., composta de acordo com as RS da linguagem

Análise semântica (ASem): destinada a verificar se as regras semânticas da linguagem são satisfeitas e calcular os valores associados aos símbolos, de modo a poder conhecer-se o significado completo da frase

CAPÍTULO 1

FASES DE RECONHECIMENTO

O AL passa ao AS os símbolos terminais que reconheceu no texto fonte — tokens;

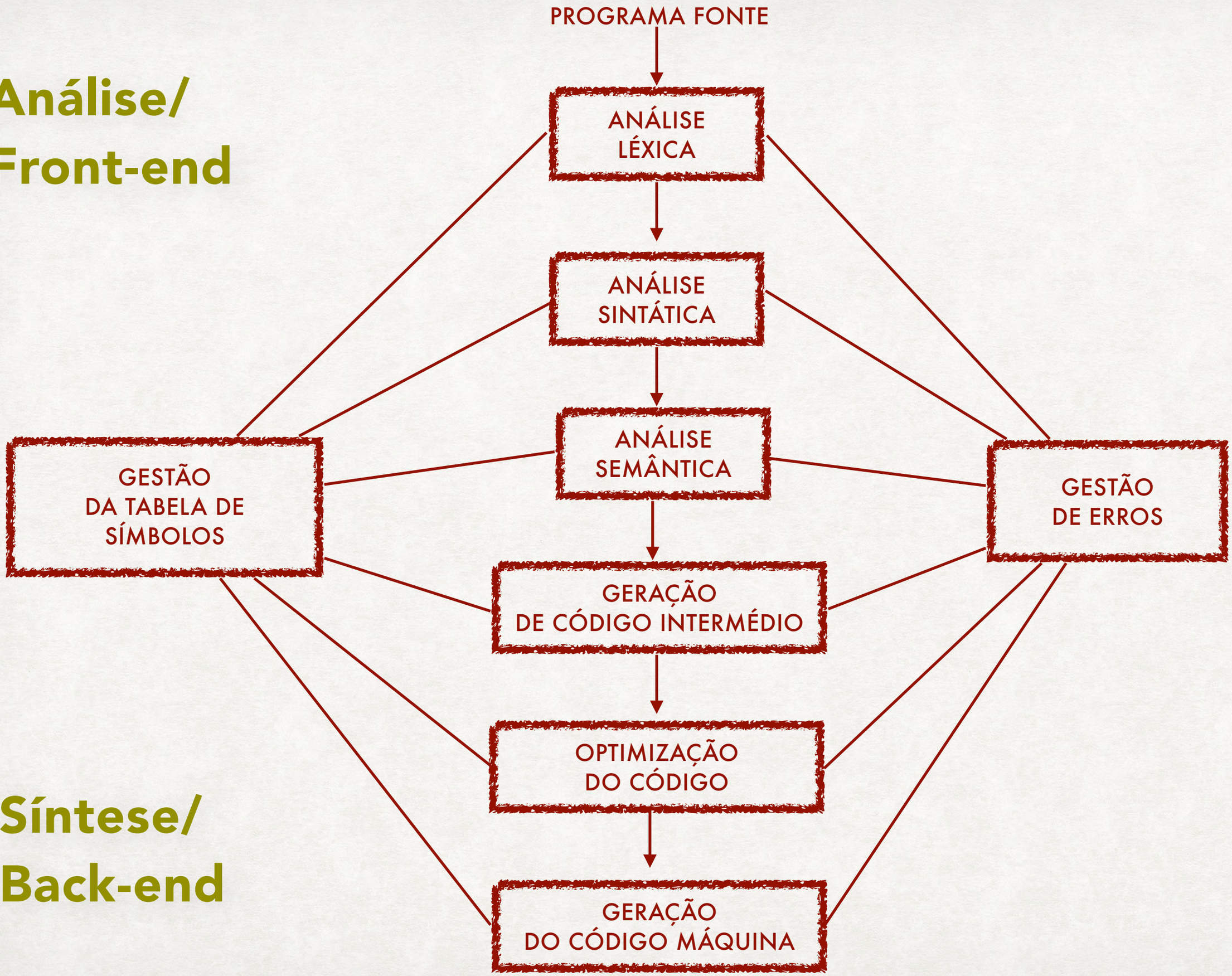
O AS envia ao Asem uma representação interna da forma da frase (estrutura sintática) ⟶ árvore de derivação para suportar essa representação

O ASem passa ao transformador uma árvore de sintaxe decorada que representa o significado da frase

Estas 3 fases distintas no seu conjunto designam-se por *front-end (ou Análise)*: o texto do programa é separado em todas as suas partes constituintes; cria-se uma representação intermédia do programa fonte

Back-end (ou Síntese): constrói-se o programa destino a partir da representação intermédia

Análise/ Front-end



Síntese/ Back-end

CAPÍTULO 1

PROCESSAMENTO DE UMA LINGUAGEM

Geração do código intermédio: transformação da frase (árvore) — análise — numa frase do código intermédio — síntese;

Tabela de símbolos (TS): A informação dos tokens do programa tem de fluir entre as várias fases da compilação. A gestão da tabela de símbolos é de primordial importância e vai cruzar todas as fases do compilador.

Gestão de erros: Trata-se de detectar os erros enviando uma mensagem apropriada para o utilizador: local do erro, tipo do erro, causa provável;

Para aumentar a produtividade é importante tentar recuperar automaticamente do erro, de forma a poder dar o máximo de continuidade à tarefa de compilação; Por ex., ao detectar a falta de “;” entre duas instruções correctas, o compilador deverá fazer a introdução automática da palavra em causa (i.e., do “;”) de forma a que a compilação possa prosseguir.

CAPÍTULO 1

O CONTEXTO DE UM COMPILADOR

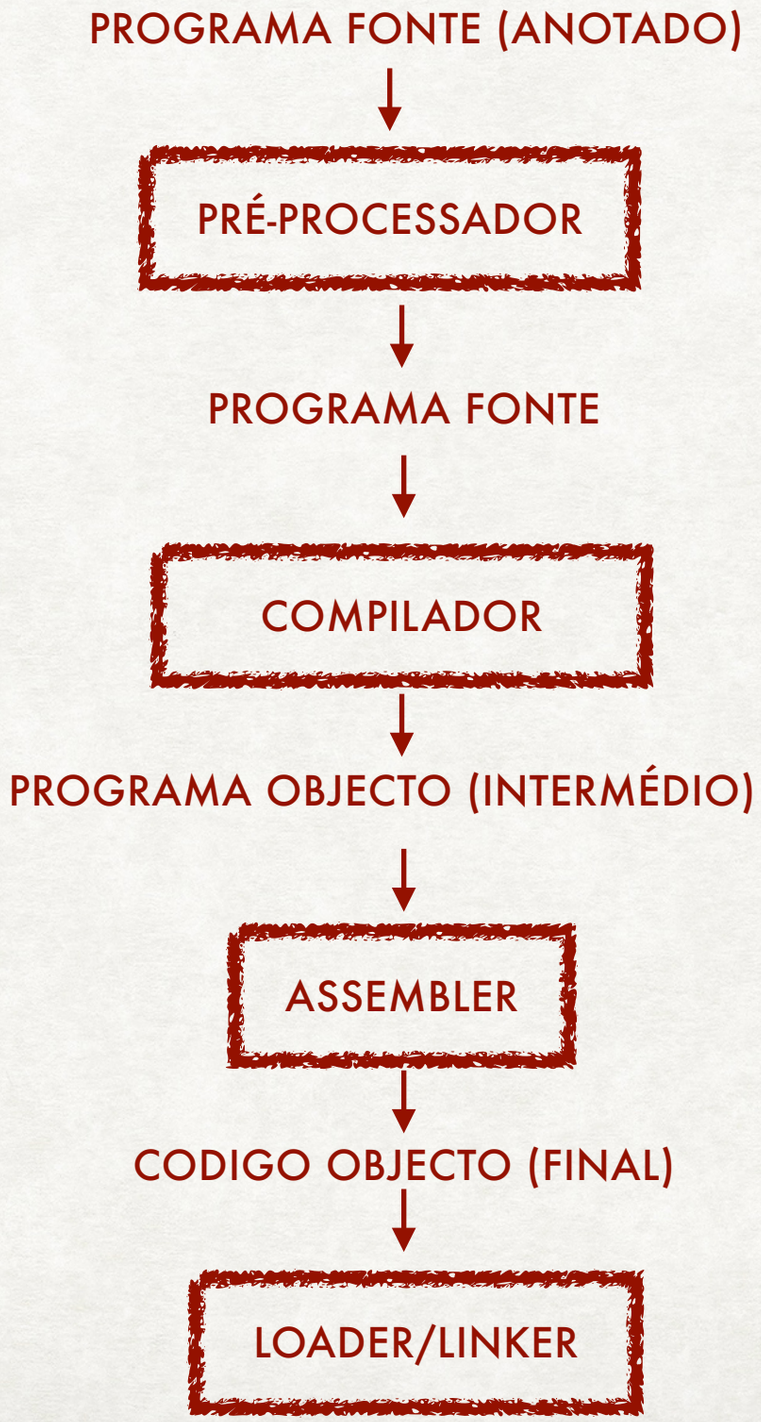
O que é usualmente designado por compilador é de facto um conjunto de programas, que no seu conjunto formam o **compilador**

Num dado sistema de compilação podem não estar presentes todas as componentes

Por vezes, a decomposição de um sistema de compilação nas suas várias componentes não é tornada explícita ao utilizador

CAPÍTULO 1

O CONTEXTO DE UM COMPILADOR



CAPÍTULO 1

O CONTEXTO DE UM COMPILADOR

pré-processador: processa directivas; retira os comentários; agrupa ficheiros se tal for necessário, entre outras tarefas; ex. no sistema GNUC, o cpp GNU C-compatible compiler pre-processor

compilador: faz a análise do texto escrito na linguagem fonte e faz a sua transcrição para a linguagem destino por razões de economia — há a possibilidade de escrever um compilador que seja adaptável a diferentes sistemas computacionais destino — a língua destino é tão somente uma linguagem genérica intermédia; ex. no sistema GNUC, o ccc “the GNU project C-compiler”

assembler: faz a transcrição da linguagem intermédia para a linguagem final (máquina); é um programa que está fortemente ligado a um e um só sistema computacional

loader/linker: no caso de se querer um programa executável, os loader/linker fazem a junção do código máquina produzido pelas anteriores fases a um conjunto de serviços — *run-time routines* — que permitem a criação de um programa independente.

CAPÍTULO 1

ESTRATÉGIAS DE PROCESSAMENTO

Tradução orientada pela sintaxe (TOS): Todo o processamento é controlado pelo AS; não há uma nítida separação entre todas as tarefas e nunca se chega a construir integralmente a árvore de sintaxe decorada, nem tão pouco a árvore de derivação; abordagem mais antiga. Gramáticas Tradutoras (GT)

Tradução orientada pela semântica (TOSem): Todas as tarefas são executadas separadamente, não se distinguindo nenhuma em relação às outras. A árvore de derivação é construída explícitamente para que todas as outras etapas possam trabalhar sobre a árvore de sintaxe decorada. Gramáticas de Atributos (GA)

CAPÍTULO 1

ESTRATÉGIAS DE PROCESSAMENTO

Estas 2 abordagens diferem:

- na estratégia de representação da informação em memória
- na técnica de desenvolvimento dos algoritmos
- no formalismo de especificação usado para descrever o processador de linguagens

TOS: mais simples de especificar; menos exigente em termos de requisitos de hardware/software

TOSem: maior rigor na descrição formal do processador a desenvolver; traz vantagens do ponto de vista da programação.