PCS-3566 / PCS-3866
Linguagens e Compiladores

Prof. João José Neto

Aula 03

Plano geral e primeiro detalhamento didático do projeto de um compilador

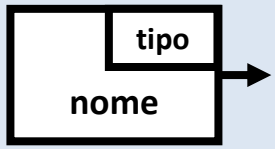
Plano estrutural do compilador

- Os slides desta apresentação mostram um esquema inicial da estrutura de um compilador típico, que pode ir sendo refinado progressivamente após adequação às especificações do compilador particular que se deseja desenvolver.
- Com base na experiência, em informações históricas e em depoimentos registrados na literatura, busca-se identificar e caracterizar, ainda de forma qualitativa, os principais elementos componentes de um compilador.
- Os esquemas assim levantados constituirão o ponto de partida em direção à estruturação definitiva do programa compilador a ser desenvolvido.
- Lembrar que, não havendo motivo em contrário, as peças do compilador devem ser implementadas de acordo com o modelo guiado por eventos, assim, desde a concepção, todas elas devem ser planejadas em consonância com esse modelo.

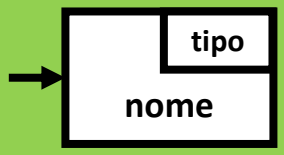
Diagramas utilizados

- Adiante, são apresentadas propostas do software a ser desenvolvido, através de diagramas elaborados segundo a seguinte convenção (informal e qualitativa):

- Representa uma peça do software, identificada por **nome**
- Suas entradas e saídas são representadas como os blocos abaixo, que se ligam a este por meio de arcos orientados de acordo com o fluxo.



- Identifica por **nome** o material de entrada que alimenta a peça de software apontada pela seta
- O **tipo** indica a natureza dessa entrada (texto, programa executável, etc)
- A seta orienta o arco que liga este bloco a um outro, que deve representar a peça de software que recebe essa entrada.

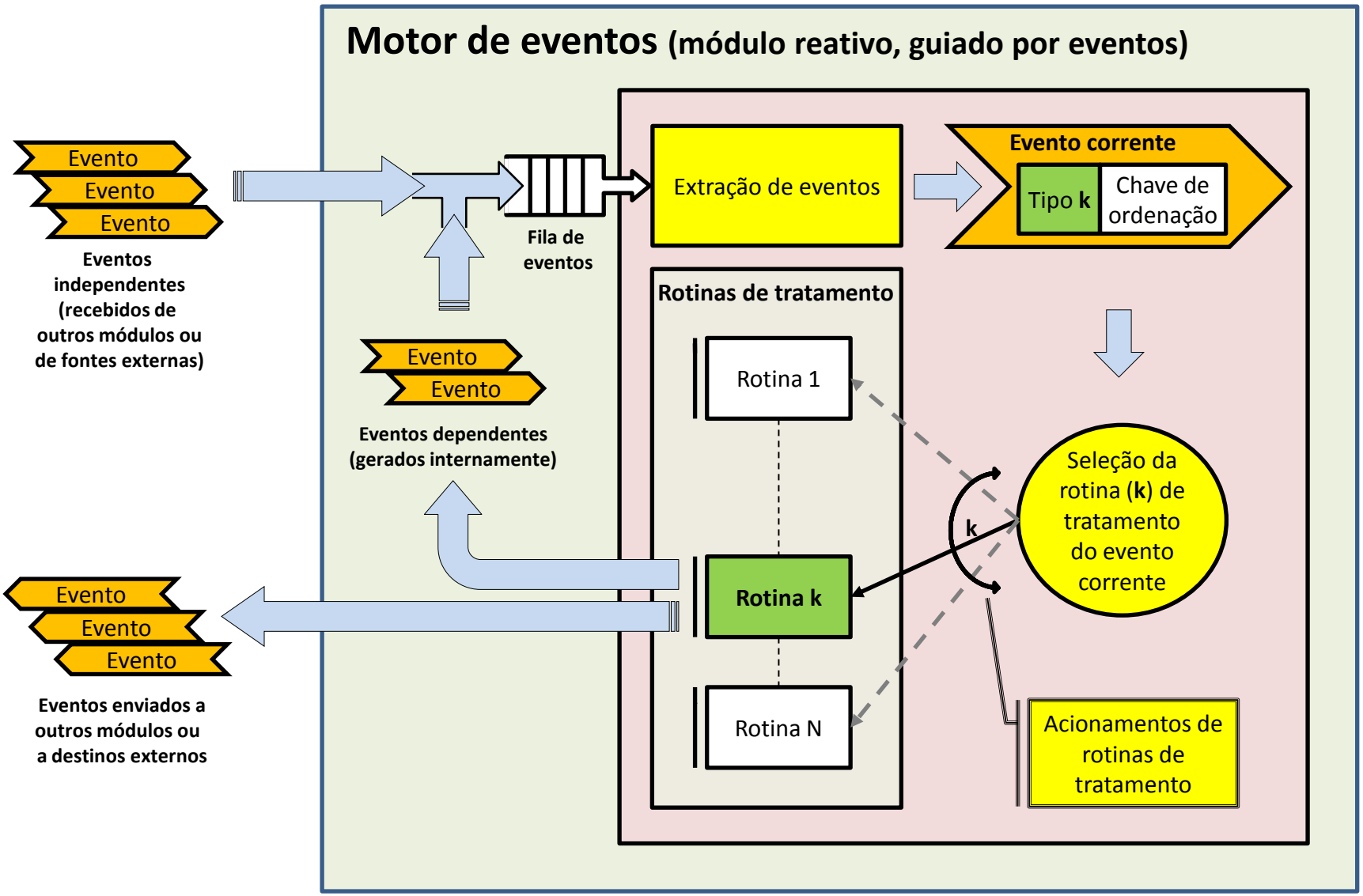


- Identifica por **nome** o material de saída gerado pela peça de software representada pelo bloco em que se origina a seta
- O **tipo** indica a natureza dessa saída (texto, programa executável, etc)
- A seta deve se originar no bloco que representa a peça de software que gera como saída esse material.

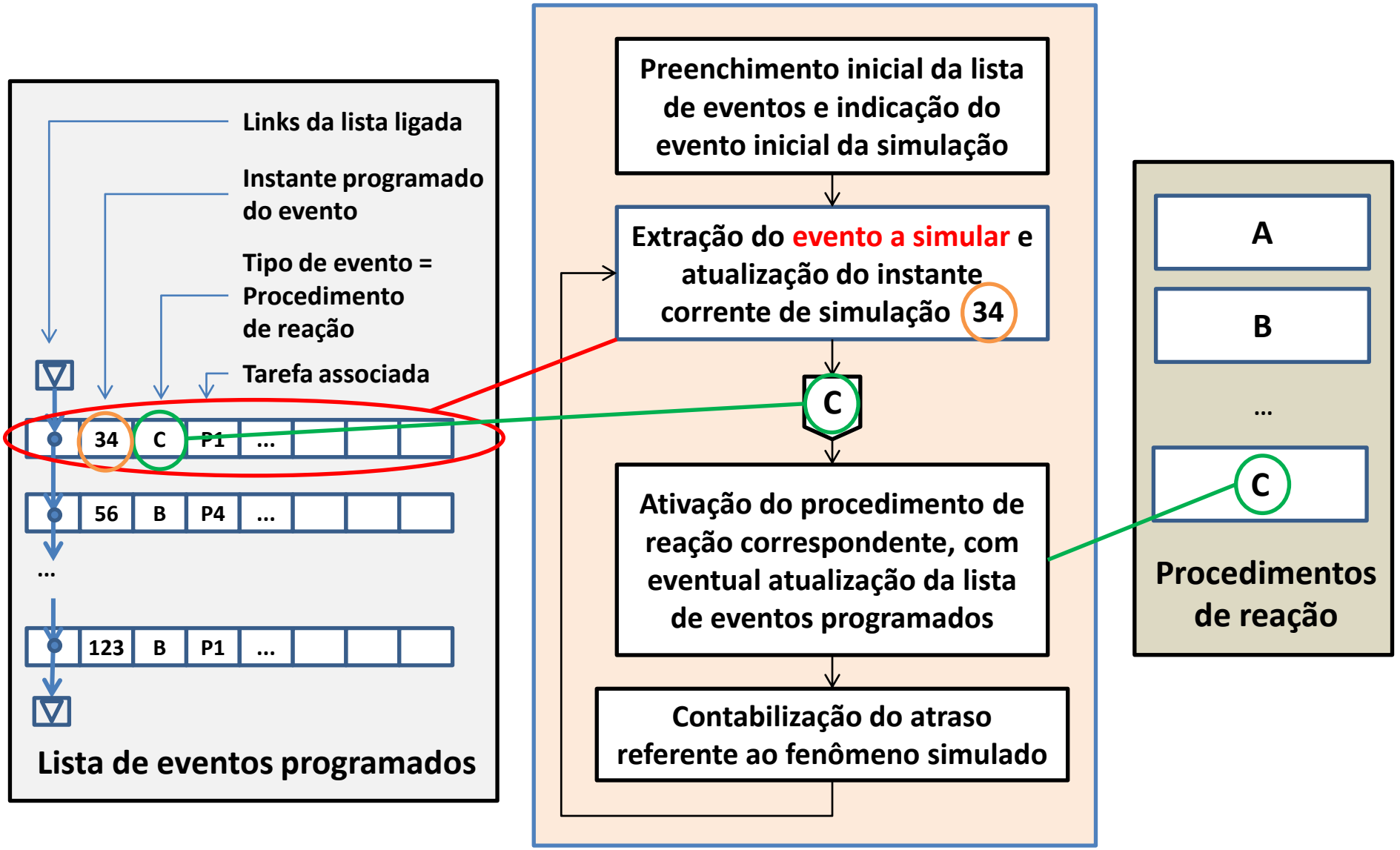
Detalhamento

- Nos detalhamentos, são também utilizadas convenções já comentadas anteriormente, para a representação de processos de simulação guiados por eventos.
- Por razões didáticas, e para ajudar a fixar os conceitos, nos slides seguintes são apresentadas representações esquemáticas diversas para um mesmo software, variando só nos detalhes apresentados:
  - Esquema de um motor de eventos genérico, em sua formulação modular potencialmente hierárquica;
  - Esquema de funcionamento de um motor de eventos, independente da aplicação;
  - Pequeno detalhamento do fluxo de eventos em um motor empregado em simulações guiadas por eventos;
  - Detalhamento inicial do uso de um motor de eventos voltado para um tipo particular de aplicação: manipulação de textos.

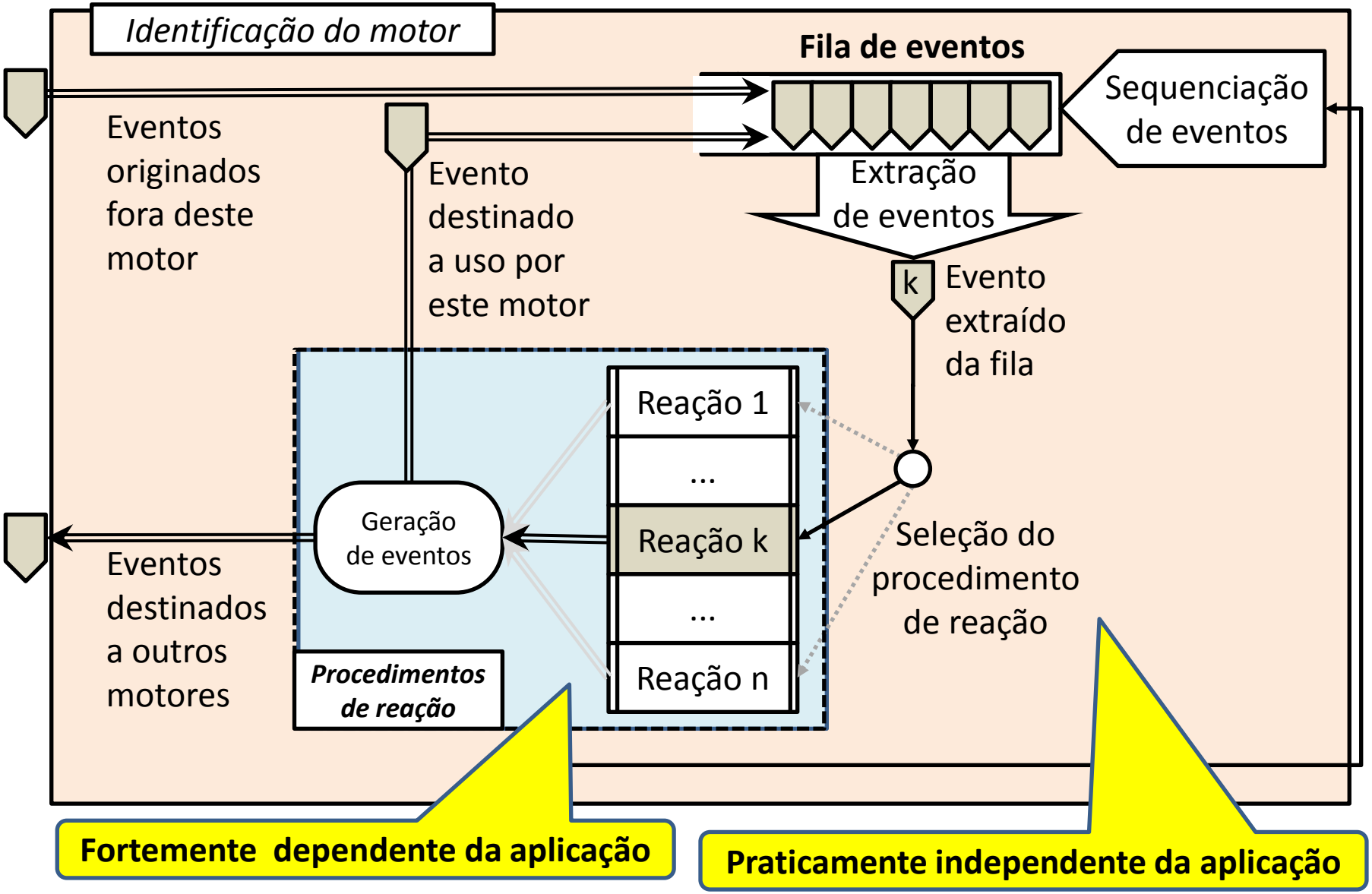
Esquema de um módulo guiado por eventos



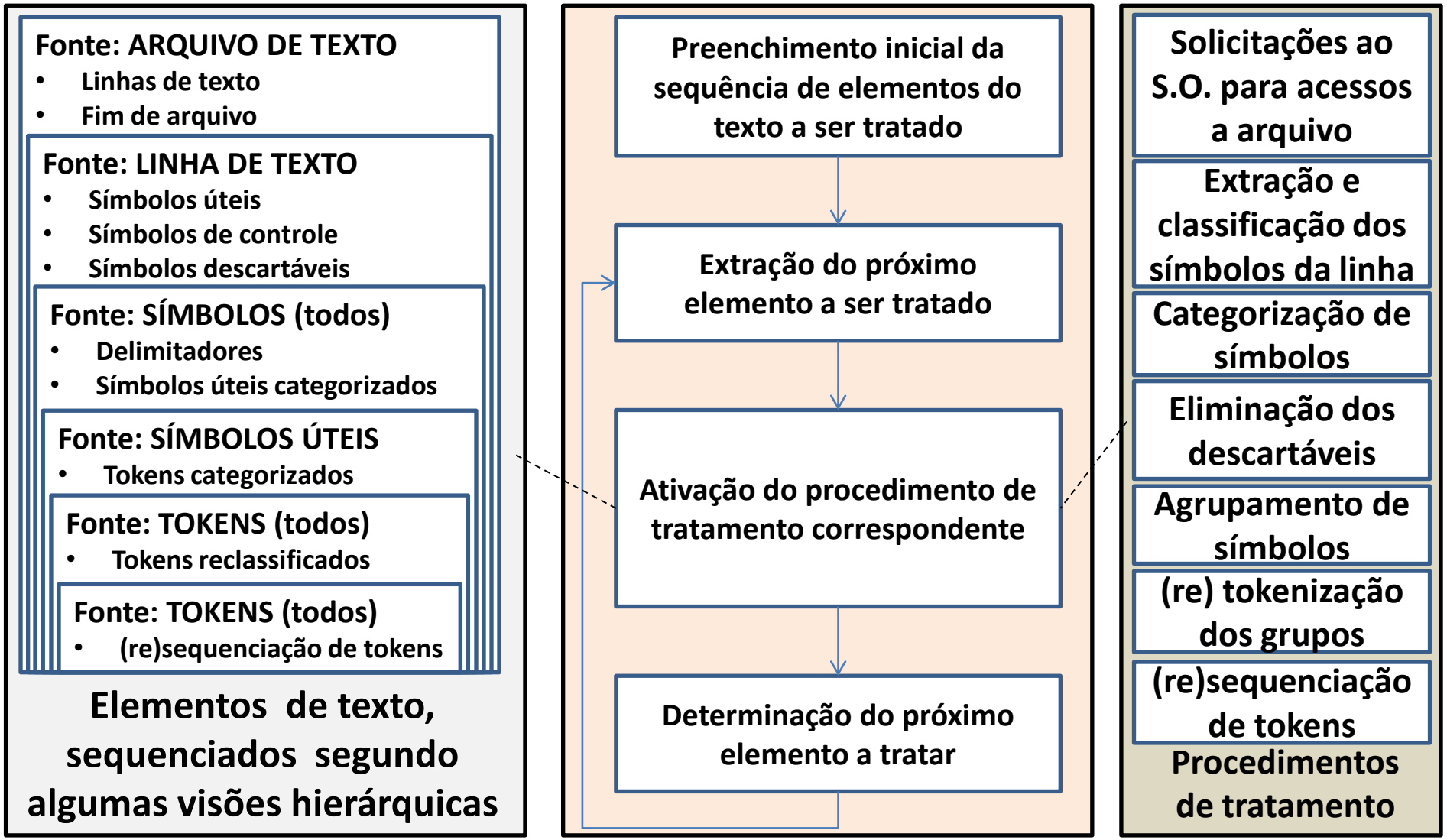
Esquema de um motor de eventos



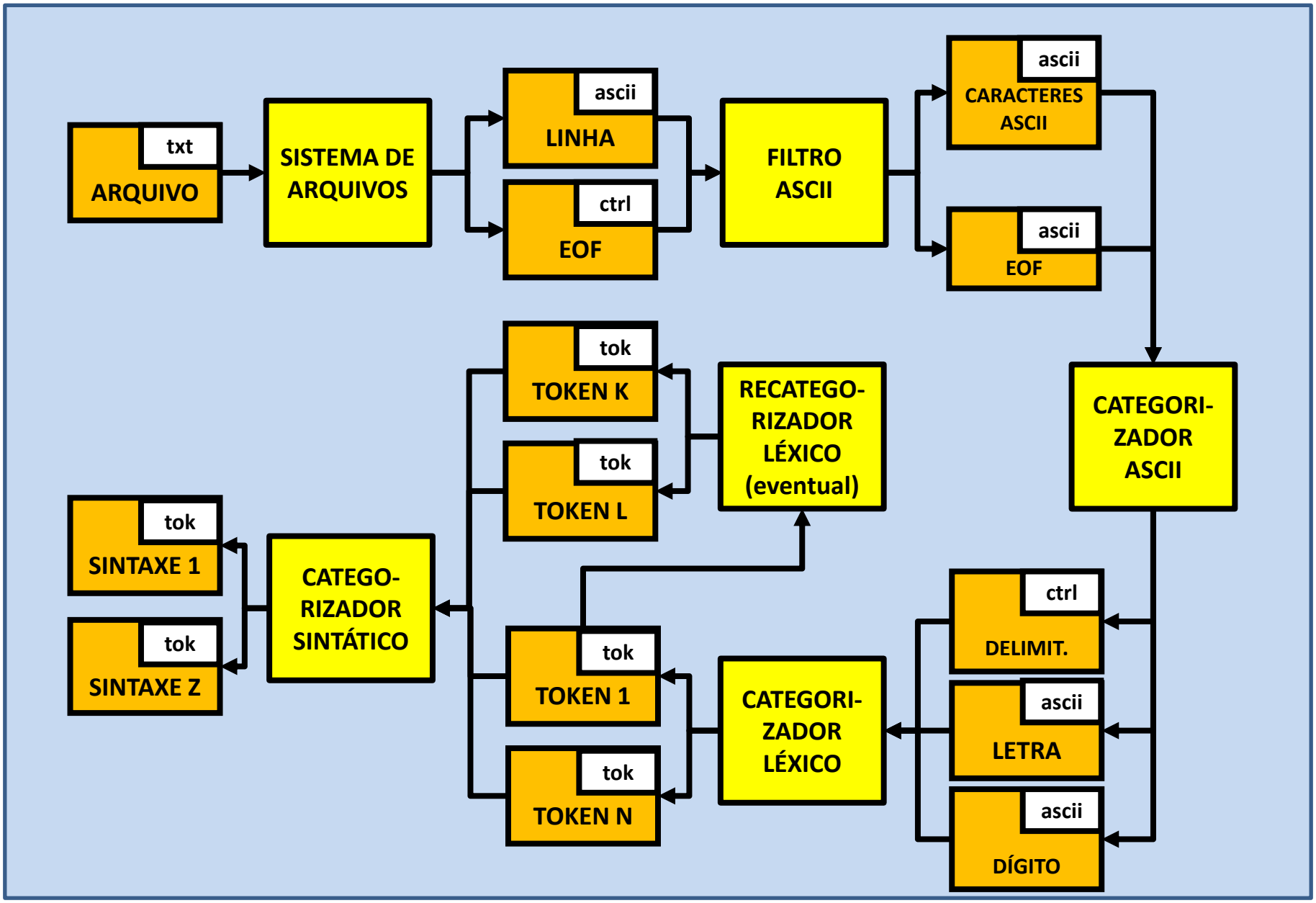
Simulação usando motor de eventos geral



Motor de eventos, voltado para aplicações em processamento textual

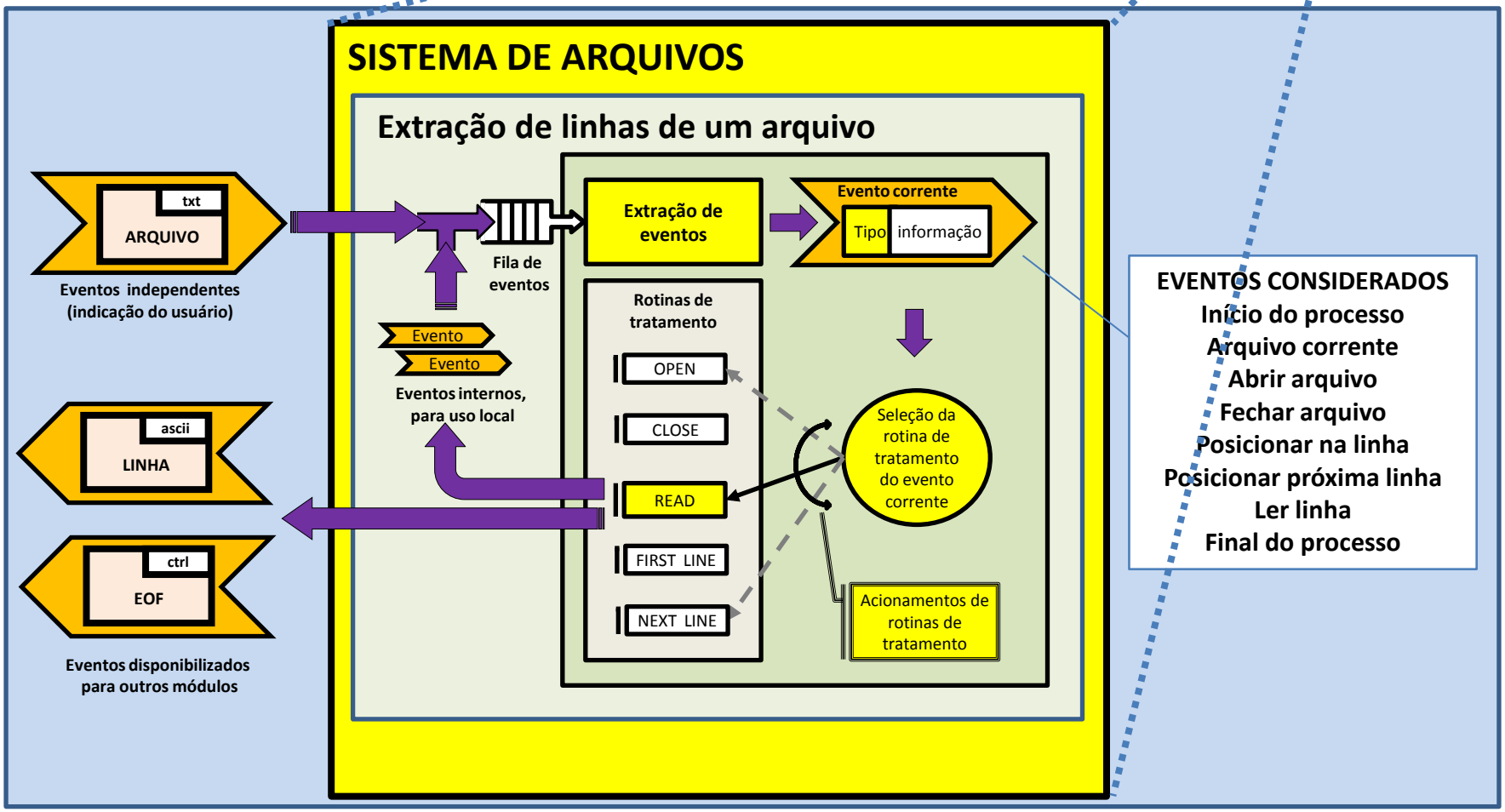


Extração Hierárquica dos elementos do texto



Detalhamento parcial de um dos blocos

Em relação ao projeto, este diagrama é parcial, impreciso e incompleto. Sua finalidade é apenas a de ilustrar o tipo de raciocínio atrelado a esta técnica de implementação guiada por eventos. Corrija-o e complete-o para adequar-se precisamente ao projeto que você está desenvolvendo.



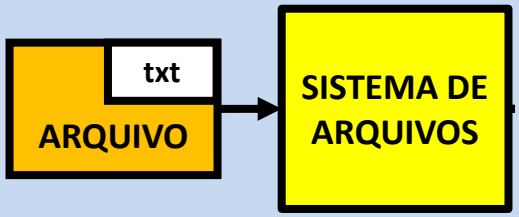
Nível 0 de abstração (arquivo ASCII)

- Neste nível, o texto-fonte está contido em um arquivo do sistema operacional, a respeito do qual se sabe apenas o nome e o tipo, nada mais.
- Não se dispõe ainda, neste nível de abstração, sequer da informação se tal arquivo realmente está presente no sistema ou não.

Exemplo:

Texto-fonte: arquivo **FONTE.TXT**

Nível 0 de abstração



Nível 1 de abstração (linhas de texto ASCII)

- Neste nível, o arquivo de texto já é visto como uma sequência de linhas contendo o texto-fonte, e terminada por um controle END-OF-FILE:

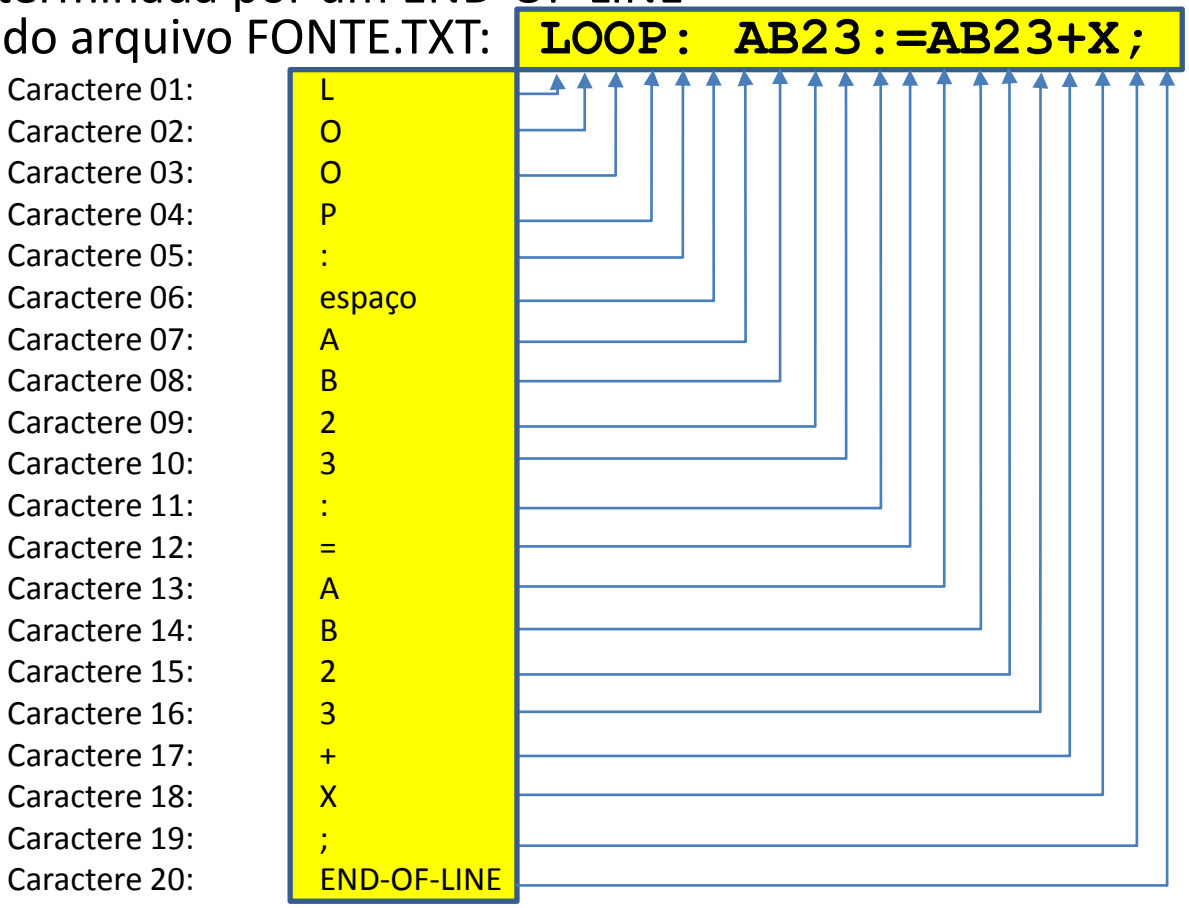
Exemplo: Arquivo **FONTE.TXT**:

Linha 1	`{ int AB23, X;`
Linha 2	`X:=200; AB23:=0;`
Linha 3	`LOOP: AB23:=AB23+X;`
Linha 4	`PRINT X, AB23;`
Linha 5	`X:=X-1;`
Linha 6	`IF X=0 STOP;`
Linha 7	`GO TO LOOP`
Linha 8	`}`
Linha 9	`END-OF-FILE`

Nível 1 de abstração

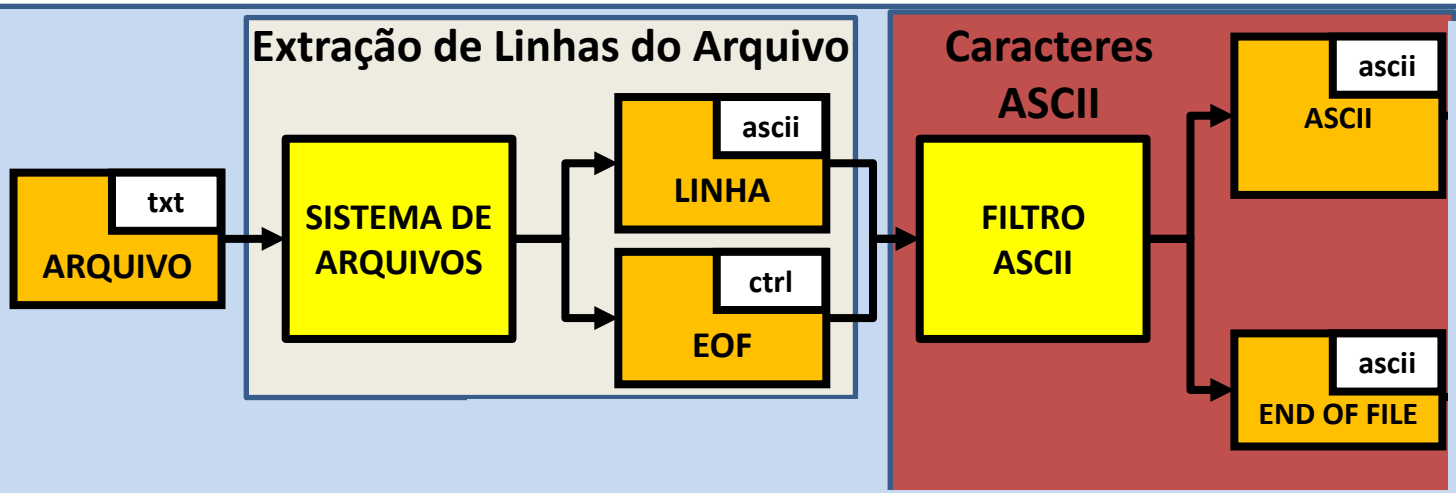
Nível 2 de abstração (caracteres ASCII)

- Neste nível, cada uma das 8 linhas do arquivo é vista como uma cadeia de caracteres ASCII, terminada por um END-OF-LINE
  Exemplo: Linha 3 do arquivo FONTE.TXT: **`LOOP: AB23:=AB23+X;`**



- Após as 8 linhas úteis do arquivo, o controle END-OF-FILE torna-se a única informação disponível
  Exemplo: linha 9 do arquivo `FONTE.TXT`:

Nível 2 de abstração



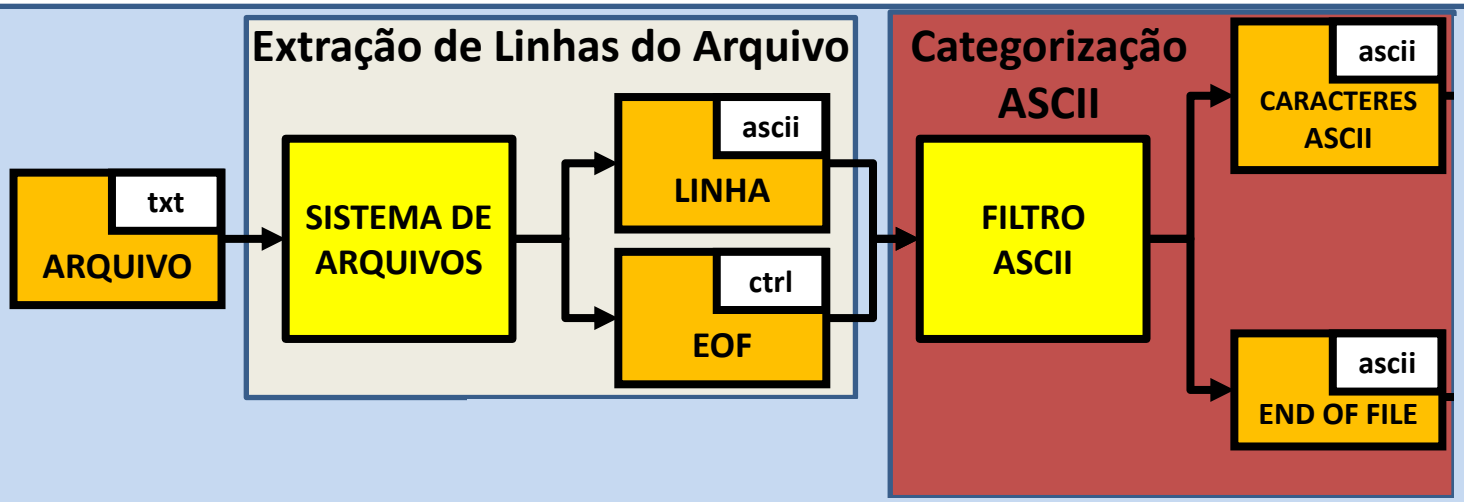
Nível 3 de abstração (filtragem ASCII básica)

- Aqui os caracteres ASCII úteis são classificados em úteis, descartáveis e controles, preparando-os para classificação.
  Exemplo: Linha 3 do arquivo FONTE.TXT: **`LOOP:AB23:=AB23+X;`**

Caractere 01:	L	útil
Caractere 02:	O	útil
Caractere 03:	O	útil
Caractere 04:	P	útil
Caractere 05:	:	útil
Caractere 06:	espaço	descartável
Caractere 07:	A	útil
Caractere 08:	B	útil
Caractere 09:	2	útil
Caractere 10:	3	útil
Caractere 11:	:	útil
Caractere 12:	=	útil
Caractere 13:	A	útil
Caractere 14:	B	útil
Caractere 15:	2	útil
Caractere 16:	3	útil
Caractere 17:	+	útil
Caractere 18:	X	útil
Caractere 19:	;	útil
Caractere 20:	END-OF-LINE	controle

- Para que seja possível essa classificação preliminar, efetuada pelo filtro, é preciso que, para cada um dos caracteres, seja executado um procedimento de verificação da pertinência daquele caractere a um conjunto completo de caracteres pertencentes às diversas possíveis categorias .
- Isso pode ser feito de várias maneiras, como por exemplo, uma busca em tabelas, ou então um cálculo efetuado sobre o código numérico que representa o caractere no sistema de codificação adotado (por exemplo, ASCII), verificando se o código do caractere testado se encontra em determinados intervalos numéricos

Nível 3 de abstração



Nível 4 de abstração (categorização ASCII)

- Aqui os caracteres ASCII úteis são classificados em dígitos, letras e especiais, e os caracteres ASCII descartáveis são convertidos em delimitadores.
  Exemplo: Linha 3 do arquivo FONTE.TXT: **`LOOP:AB23:=AB23+X;`**

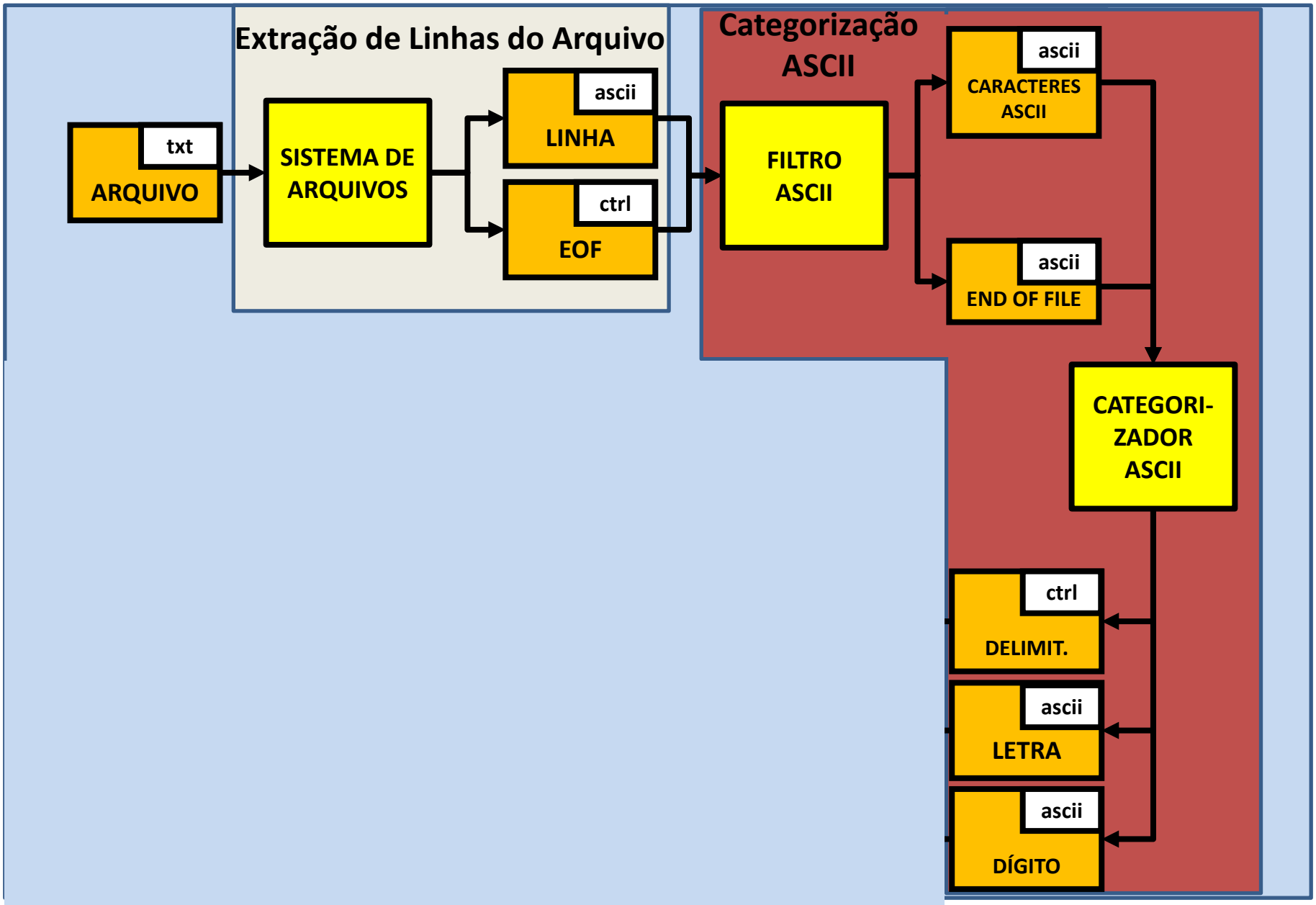
Caractere 01:	L	útil	letra
Caractere 02:	O	útil	letra
Caractere 03:	O	útil	letra
Caractere 04:	P	útil	letra
Caractere 05:	:	útil	especial
Caractere 06:	espaço	descartável	delimitador
Caractere 07:	A	útil	letra
Caractere 08:	B	útil	letra
Caractere 09:	2	útil	dígito
Caractere 10:	3	útil	dígito
Caractere 11:	:	útil	especial
Caractere 12:	=	útil	especial
Caractere 13:	A	útil	letra
Caractere 14:	B	útil	letra
Caractere 15:	2	útil	dígito
Caractere 16:	3	útil	dígito
Caractere 17:	+	útil	especial
Caractere 18:	X	útil	letra
Caractere 19:	;	útil	especial
Caractere 20:	END-OF-LINE	controle	controle

- Aqui vale o mesmo que foi dito em relação ao nível 3. Procedimentos simples de classificação, que consistem de buscas em tabelas e pequenos cálculos envolvendo os códigos numéricos utilizados para a representação dos caracteres bastam para realizar essa categorização.

Observação de ordem prática

- Se observarmos os níveis 2, 3 e 4 de abstração, é fácil perceber que não há mudanças do ponto de vista estrutural na passagem de um nível para o outro.
- Assim sendo, por questão de desburocratização do programa que se está construindo, fica evidente que não há necessidade de cada um desses níveis ser implementado de forma independente e separada.
- Portanto, torna-se interessante que na implementação desses três níveis de abstração isso seja feito utilizando um único motor de eventos e acumulando as funções das rotinas de tratamento dos eventos, o que pode tornar mais ágil e econômica essa implementação.

Nível 4 de abstração



Nível 5 de abstração (extração e categorização léxicas)

- Neste nível nascem os tokens da linguagem, a partir do agrupamento de símbolos adjacentes, de acordo com a parte léxica da gramática da linguagem cujo compilador se deseja produzir.
- No nosso caso, os tokens principais a serem categorizados são os identificadores, os números inteiros e os símbolos especiais compostos de dois ou mais caracteres. Todos os demais caracteres especiais formam cada qual uma categoria própria.
  Exemplo: Linha 3 do arquivo FONTE.TXT: **`LOOP:  AB23:=AB23+X;`**

Caractere 01:	L	útil	letra	.
Caractere 02:	O	útil	letra	.
Caractere 03:	O	útil	letra	.
Caractere 04:	P	útil	letra	identificador “LOOP”
Caractere 05:	:	útil	especial	especial “:”
Caractere 06:	espaço	descartável	delimitador	.
Caractere 07:	A	útil	letra	.
Caractere 08:	B	útil	letra	.
Caractere 09:	2	útil	dígito	.
Caractere 10:	3	útil	dígito	identificador “AB23”
Caractere 11:	:	útil	especial	.
Caractere 12:	=	útil	especial	especial “:=“
Caractere 13:	A	útil	letra	.
Caractere 14:	B	útil	letra	.
Caractere 15:	2	útil	dígito	.
Caractere 16:	3	útil	dígito	identificador “AB23”
Caractere 17:	+	útil	especial	especial “+“
Caractere 18:	X	útil	letra	identificador “X”
Caractere 19:	;	útil	especial	especial “;“
Caractere 20:	END-OF-LINE	controle	controle	controle END-OF-LINE

Lógica de extração dos tokens

- Os procedimentos de categorização léxica são menos triviais que os da classificação dos caracteres ASCII, dada a complexidade estrutural dos elementos léxicos (tokens), que, embora não seja tão grande, por certo é bem maior que a dos caracteres isolados.
- Por outro lado, essa complexidade se limita à das linguagens regulares, formadas apenas usando concatenações, uniões de conjuntos e repetições.
- Isso indica o uso de autômatos finitos para representar as estruturas válidas que formam os tokens.
- Assim, cada símbolo encontrado no texto de entrada do compilador pode, após as devidas classificações e filtragens, alimentar um autômato finito que represente o conjunto dos tokens permitidos. O resultado do processamento assim efetuado fornece a classificação dos tokens que forem sendo encontrados.

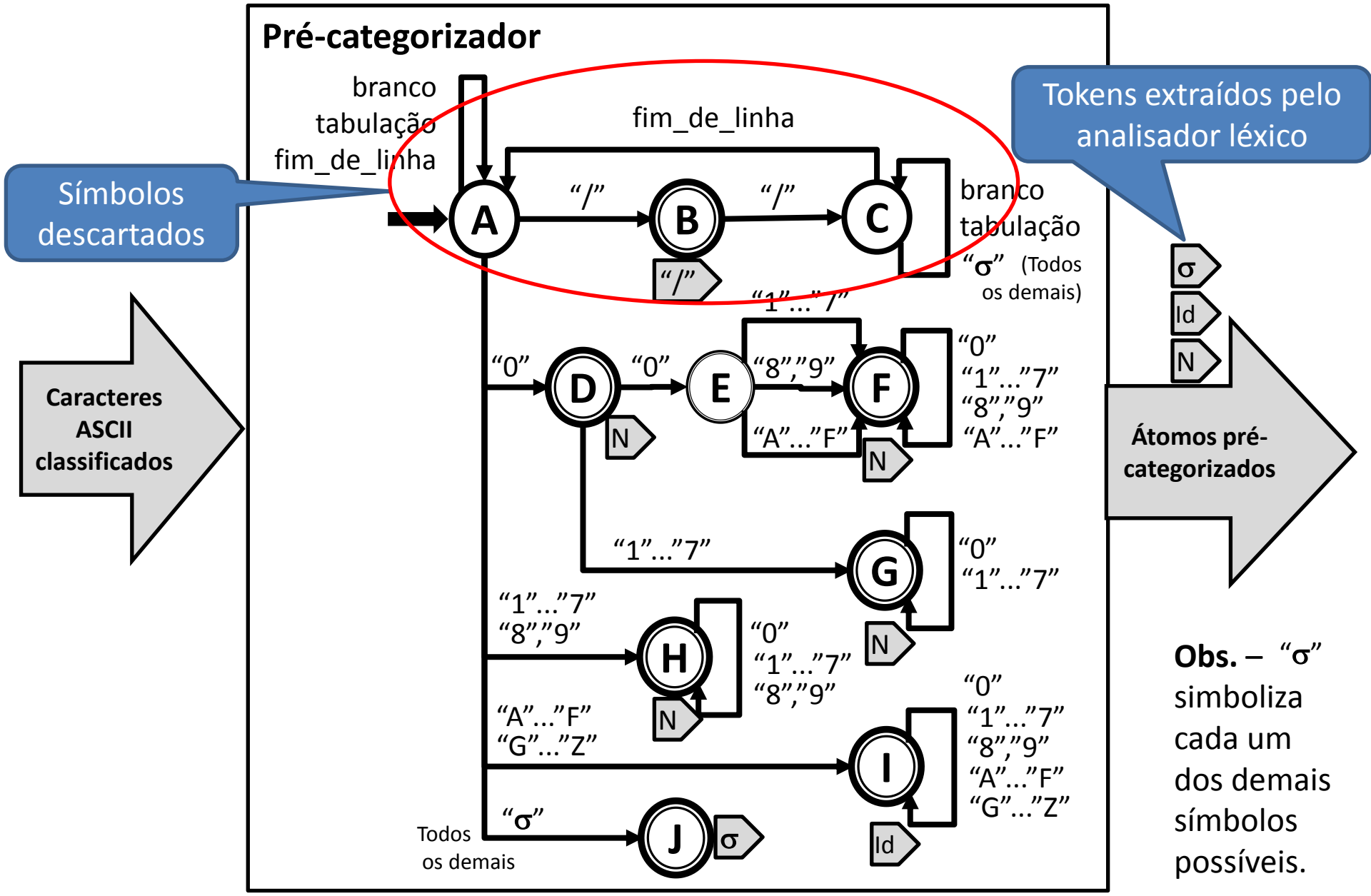
Autômato de análise léxica

- Em particular, a análise léxica para uma linguagem de programação é relativamente padronizada, e envolve essencialmente a extração de: identificadores (nomes de variáveis), palavras reservadas, números, símbolos especiais, simples e compostos, etc.
- O autômato relativamente simples do slide seguinte representa esse conjunto de tokens, e pode ser usado como base para a construção de um extrator de tokens específico para a nossa linguagem.
- Em outra apresentação, não apresentada em aula, mas disponibilizada no site da disciplina, é detalhado um pouco mais o processo de análise léxica voltado para as finalidades da compilação.

Atividades complementares

- Além da função principal de extrair e classificar tokens, é usual que os analisadores léxicos efetuem algumas outras atividades que não lhe seriam próprias, mas que são por ele realizadas por motivos práticos.
- É o caso, principalmente, da emissão de listagens, da conversão de numerais para um formato interno e da criação e manutenção de tabelas de símbolos para o programa.
- Dado o caráter sequencial dessas atividades, recomenda-se que o seu desenvolvimento também seja feito de modo incremental, com o apoio de motores de eventos.

Categorização em um analisador léxico



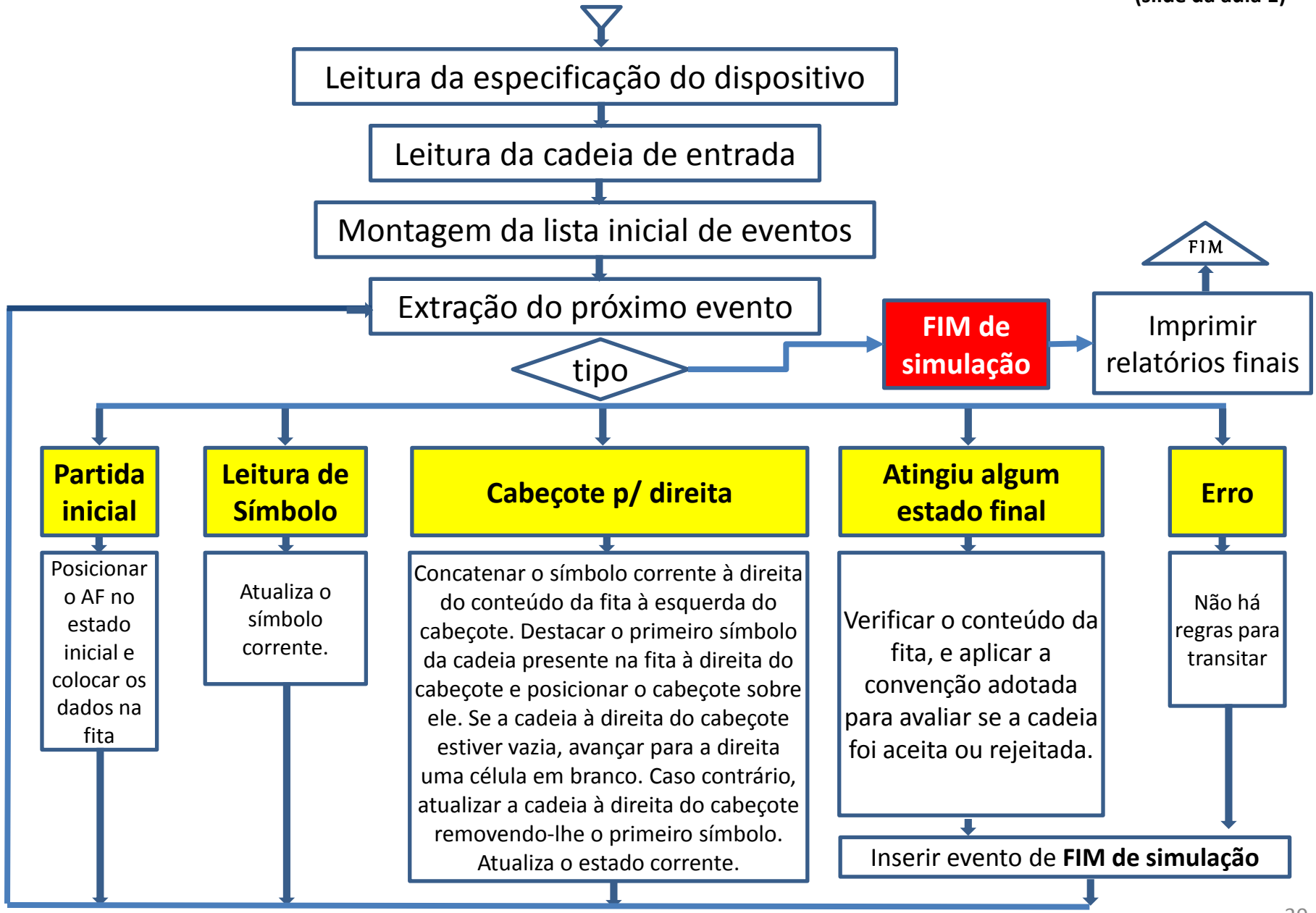
Obs. – "σ" simboliza cada um dos demais símbolos possíveis.

Observação sobre autômatos

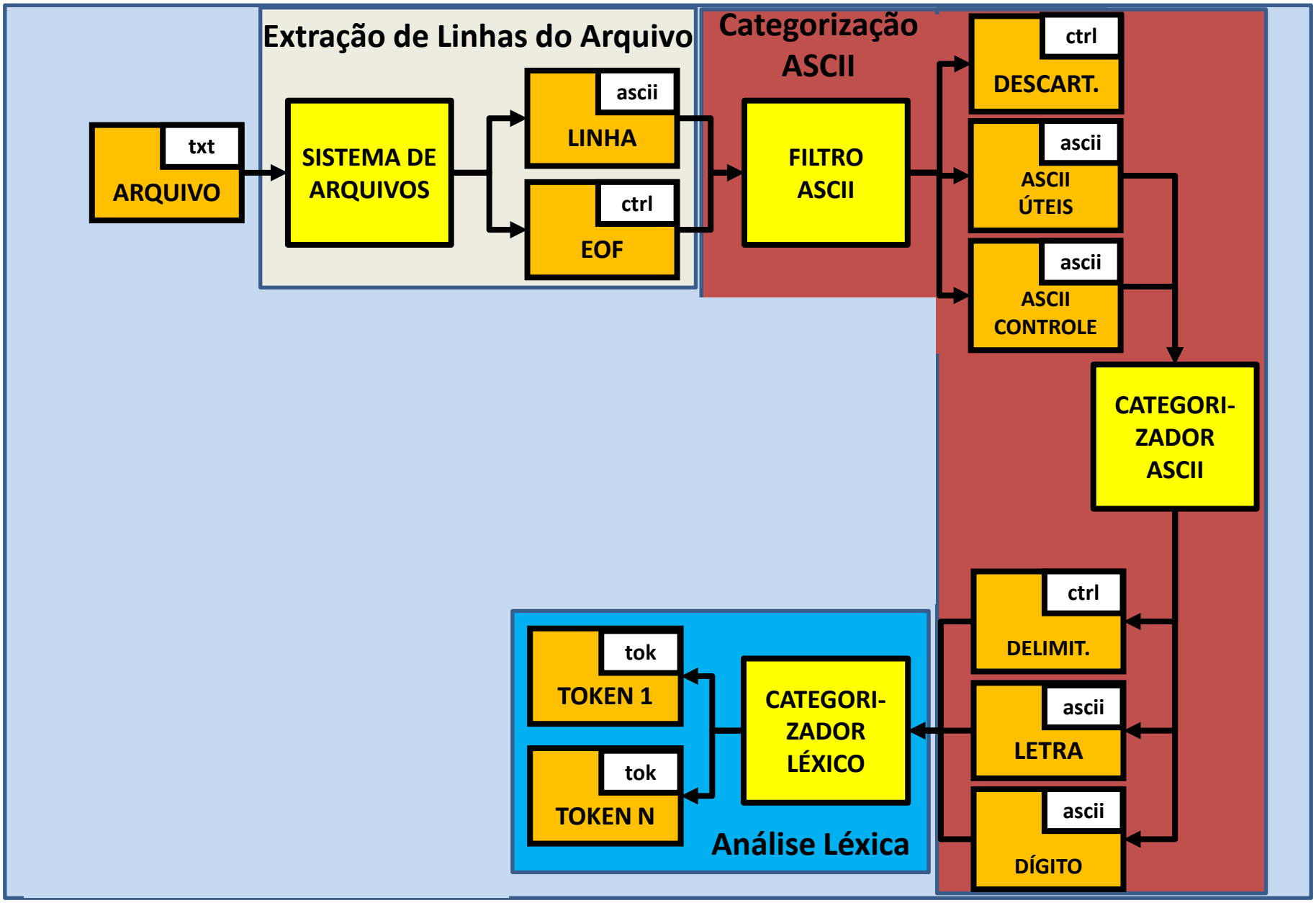
- Mais uma vez, a ocorrência de uma atividade estritamente sequencial, e bastante afinada com a programação guiada por eventos, leva-nos a considerar o uso desta técnica para a simulação das máquinas de estados em geral.
- Em particular, os autômatos finitos, como este da categorização léxica, podem ser representados em notação tabular, bastante apropriada para uso interno por parte dos programas que implementam o compilador.
- Nesse cenário, o motor de eventos se restringe a executar sucessivas transições com base nas tabelas que definem o autômato, e a cada transição, pode ser acionada uma rotina de tratamento, que execute atividades semânticas, ou seja, de manipulação dos elementos que compõem o compilador, suas entradas e saídas.
- Repetido da primeira aula, o slide seguinte mostra o processo de simulação, guiada por eventos, de um autômato finito

Simulação de autômato finito, guiada por eventos

(slide da aula 1)



Nível 5 de abstração



Nível 6 de abstração (recategorização léxica)

- Uma vez extraídos e classificados os tokens do nível 5, pode suceder que alguma categoria em que foram classificados permita que, para certas linguagens, seja utilizada diretamente tal classificação, enquanto para outras linguagens possa ser necessária uma reclassificação.
- Um exemplo muito significativo é o caso da classe dos identificadores. Linguagens que possuem palavras reservadas (atualmente, quase todas as linguagens mais usadas) necessitam reconsiderar a provável classificação prévia de palavras reservadas como identificadores, incluindo-as devidamente na correspondente classe de tokens a parte.
- Em nosso caso, isso é um pouco diferente, devido ao formato restrito dos identificadores.

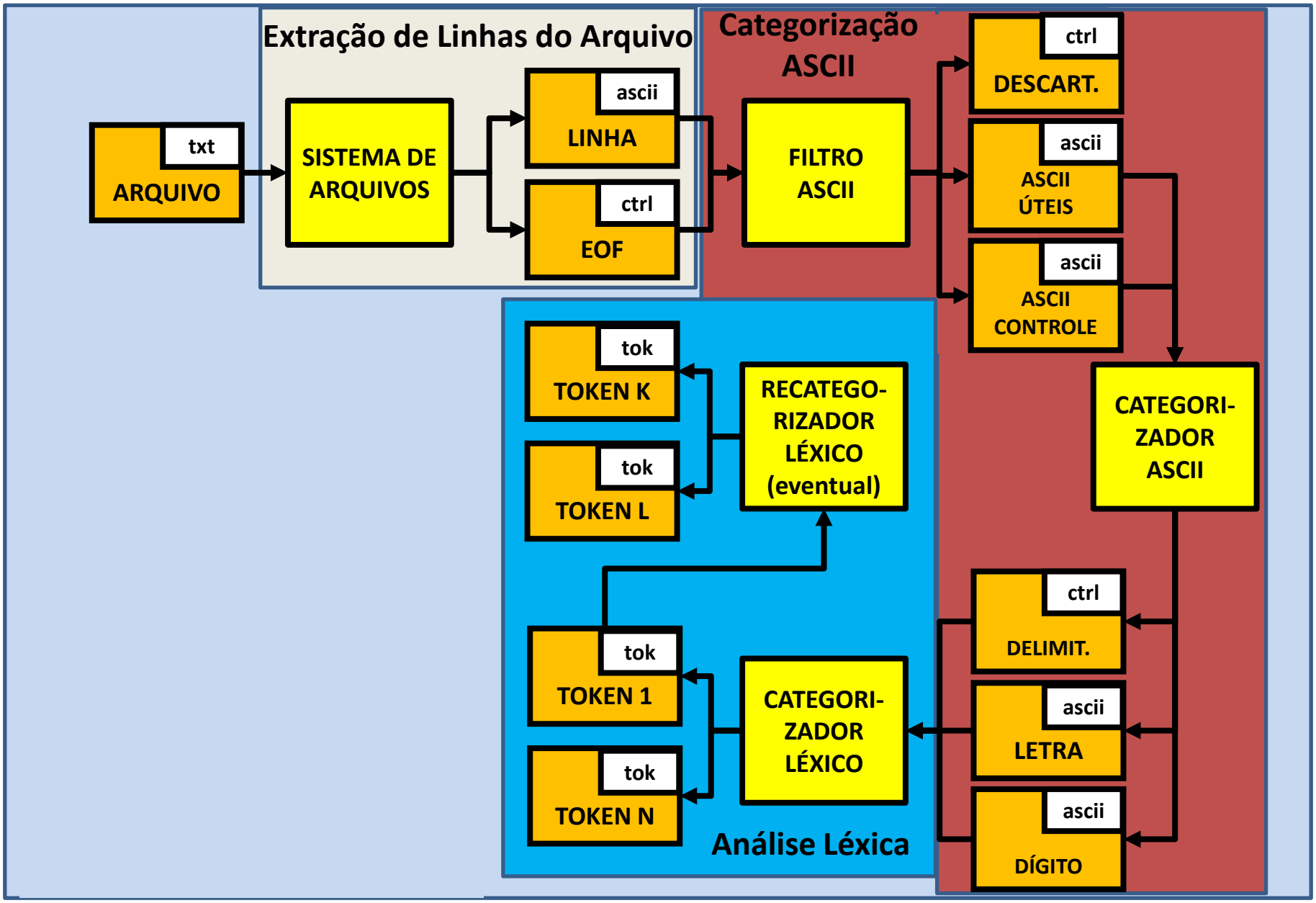
Exemplo: `Linha 7` **`GO TO LOOP`**

Caractere 01:	espaço	descartável	.		
Caractere 02:	espaço	descartável	.		
Caractere 03:	espaço	descartável	.		
Caractere 04:	espaço	descartável	.		
Caractere 05:	espaço	descartável	.		
Caractere 06:	espaço	descartável	delimitador		
Caractere 07:	G	útil	.		
Caractere 08:	O	útil	identificador	identificador “GO”	.
Caractere 09:	espaço	descartável	delimitador		
Caractere 10:	T	útil	.		
Caractere 11:	O	útil	identificador	identificador “TO”	reservada ”GOTO”
Caractere 12:	espaço	descartável	delimitador		
Caractere 13:	L	útil	.		
Caractere 14:	O	útil	.		
Caractere 15:	O	útil	.		
Caractere 16:	P	útil	identificador	identificador “LOOP”	identificador “LOOP”
Caractere 17:	END-OF-LINE	controle	controle	controle END-OF-LINE	controle END-OF-LINE

Reclassificação de identificadores

- Apenas identificadores poderiam ser recategorizados em nossa linguagem.
- Para isso, bastaria uma busca em uma tabela contendo todas as palavras reservadas da linguagem para determinar se um suposto identificador não seria na realidade uma dessas palavras reservadas.
- Quando fosse este o caso, bastaria reclassificar o identificador como sendo a palavra reservada em questão.
- Entretanto, em nossa linguagem há apenas identificadores em dois formatos restritos: uma letra, ou uma letra e um dígito. Nenhuma das palavras reservadas conflita com qualquer desses identificadores, portanto nunca haverá o que reclassificar.
- Apesar disso, o analisador léxico fica preparado para uma eventual modificação da linguagem, bem como para atender as necessidades de outras linguagens com características diferentes.

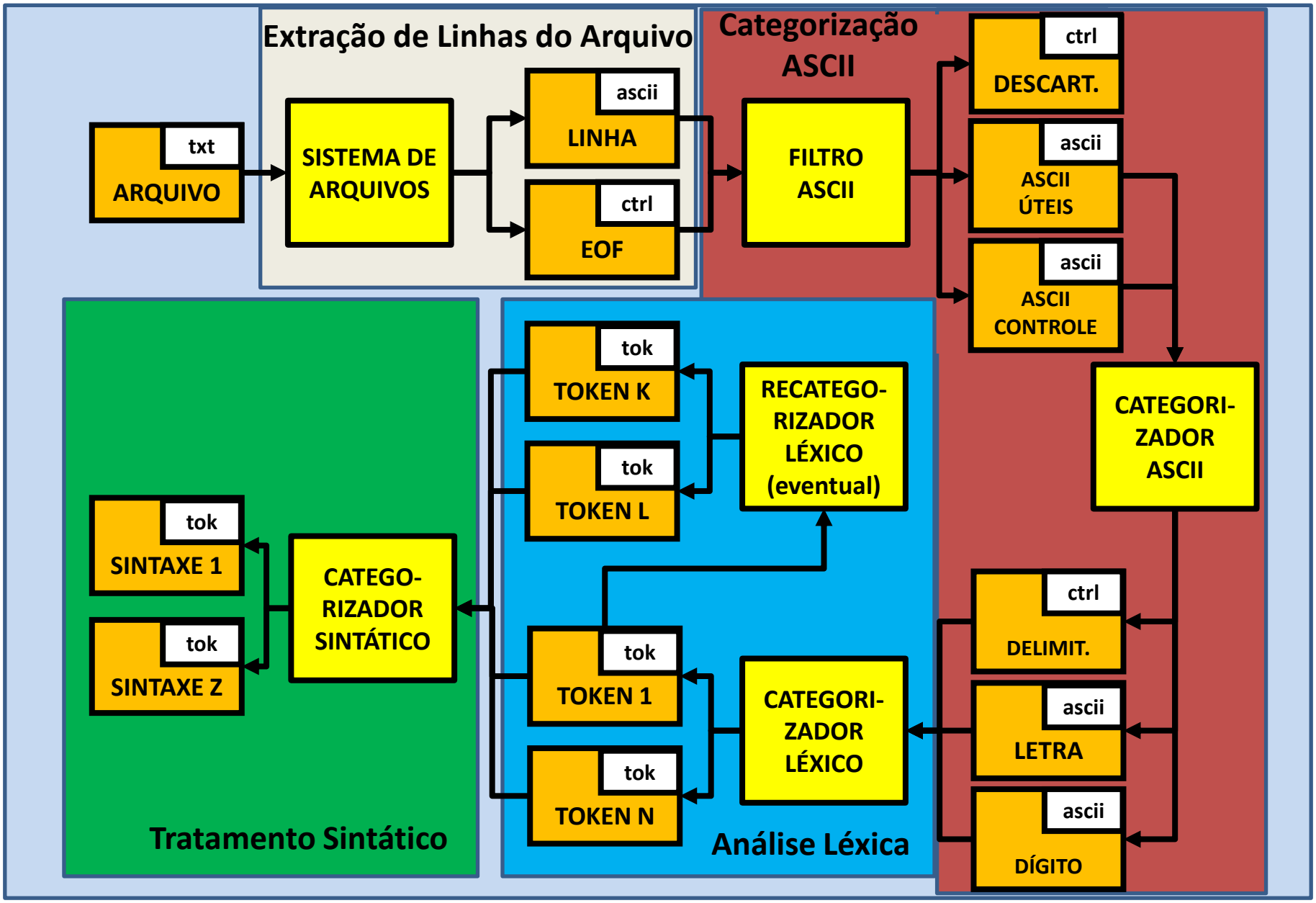
Nível 6 de abstração



Nível 7 de abstração (tratamento sintático)

- Uma vez estando todos os tokens corretamente classificados, é possível finalizar a análise léxica do texto fonte, e passar à fase seguinte da compilação.
- Nessa fase, a sequência de tokens é verificada quanto ao correto posicionamento relativo dos mesmos, usando como referência as regras gramaticais da linguagem.
- As próximas aulas são dedicadas ao estudo de métodos de reconhecimento e de análise sintática, por isso esta atividade será detalhada mais tarde.

Nível 7 de abstração



Próximos níveis de detalhamento

- Até aqui foram tratados os níveis de abstração associados com a análise do texto de entrada, sob um enfoque léxico.
- No processo da compilação, a sequência de tokens até agora considerados deverá ser, a partir deste ponto, verificada quanto à sua adequação sintática, com base em comparações frente ao conjunto de estruturas sintáticas legítimas da linguagem, definidas na gramática que a formaliza. Tema para as próximas aulas.

Para fazer no projeto

- Revise todos os slides desta apresentação tendo em vista o seu projeto. Corrija o que estiver errado, e complete o que estiver incompleto.
- Sempre dentro da técnica de programação guiada por eventos, use instâncias do seu motor de eventos para tornar operacionais os 6 primeiros níveis de abstração apresentados.
- Faça-os funcionar juntos acrescentando-os um de cada vez aos demais que já estiverem funcionando.
- A cada acréscimo, faça os testes necessários para certificar-se de que o programa desejado está funcionando a contento.
- Não deixe também de refazer os testes anteriores, para ter certeza de que algo que já funcionava não tenha deixado de funcionar por causa de elementos recém-introduzidos.
- O nível 7 de abstração (tratamento sintático) será nosso alvo de atenção nas próximas aulas da disciplina.