Unidade VI:

Árvore *Trie*

Prof. Max do Val Machado

Instituto de Ciências Exatas e Informática
Curso de Ciência da Computação

Introdução

- As árvores *trie* são estruturas de dados para a procura rápida de padrões

- Elas são usadas em aplicações de pré-processamento do texto

- O nome *trie* é derivado da palavra *retrieval* (recuperação)

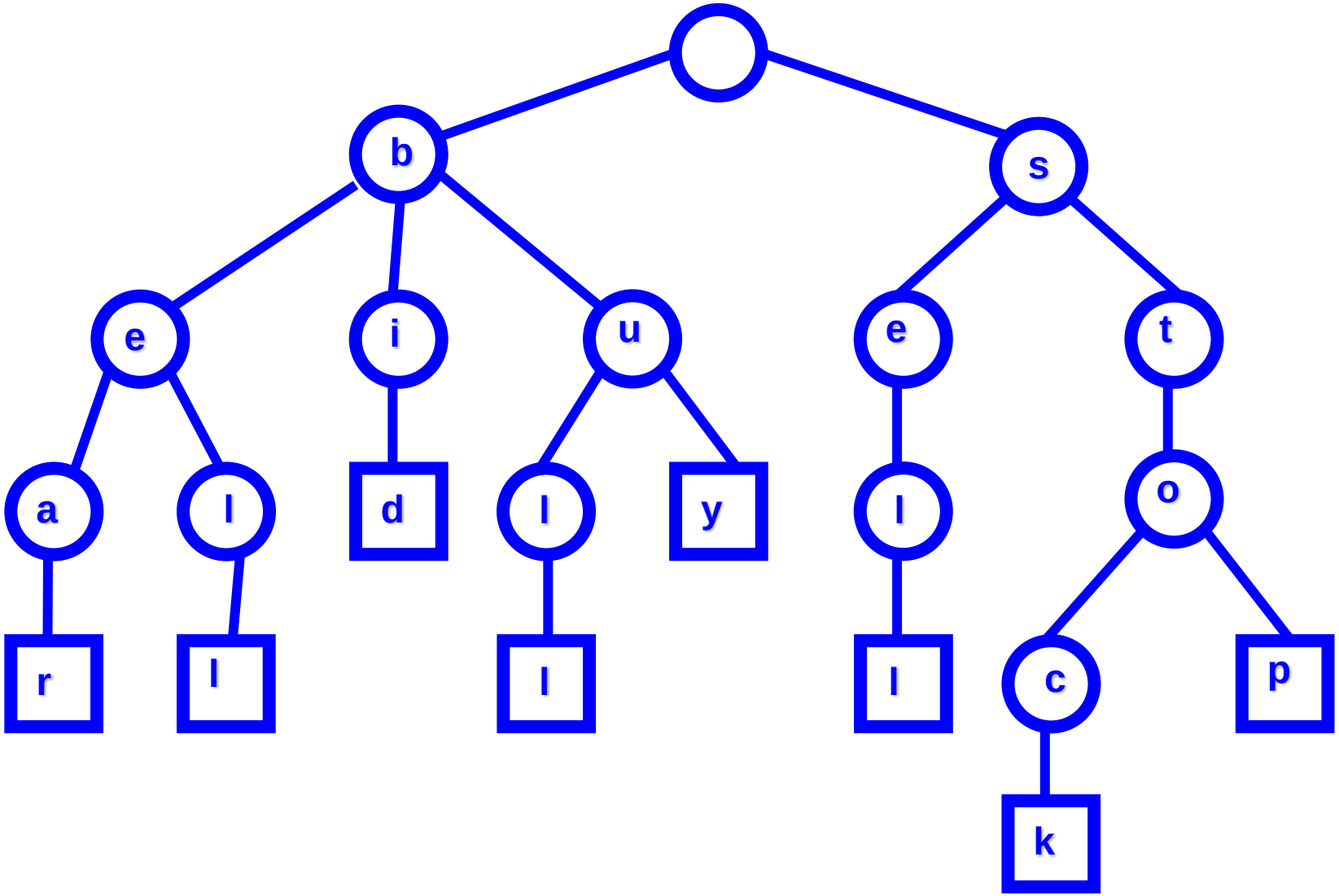
Exemplos de Aplicações

- Índices
- Armazenamento de Palavras (dicionários)
- Procura de uma sequência de DNA em uma base de genomas

Definição

- Tem-se uma coleção de S cadeias de caracteres utilizando o mesmo alfabeto e as operações primárias suportadas são:
  - Procura de padrões
  - Procura de prefixos: Recebe-se uma cadeia X e retornam-se todas as cadeias que têm X como prefixo

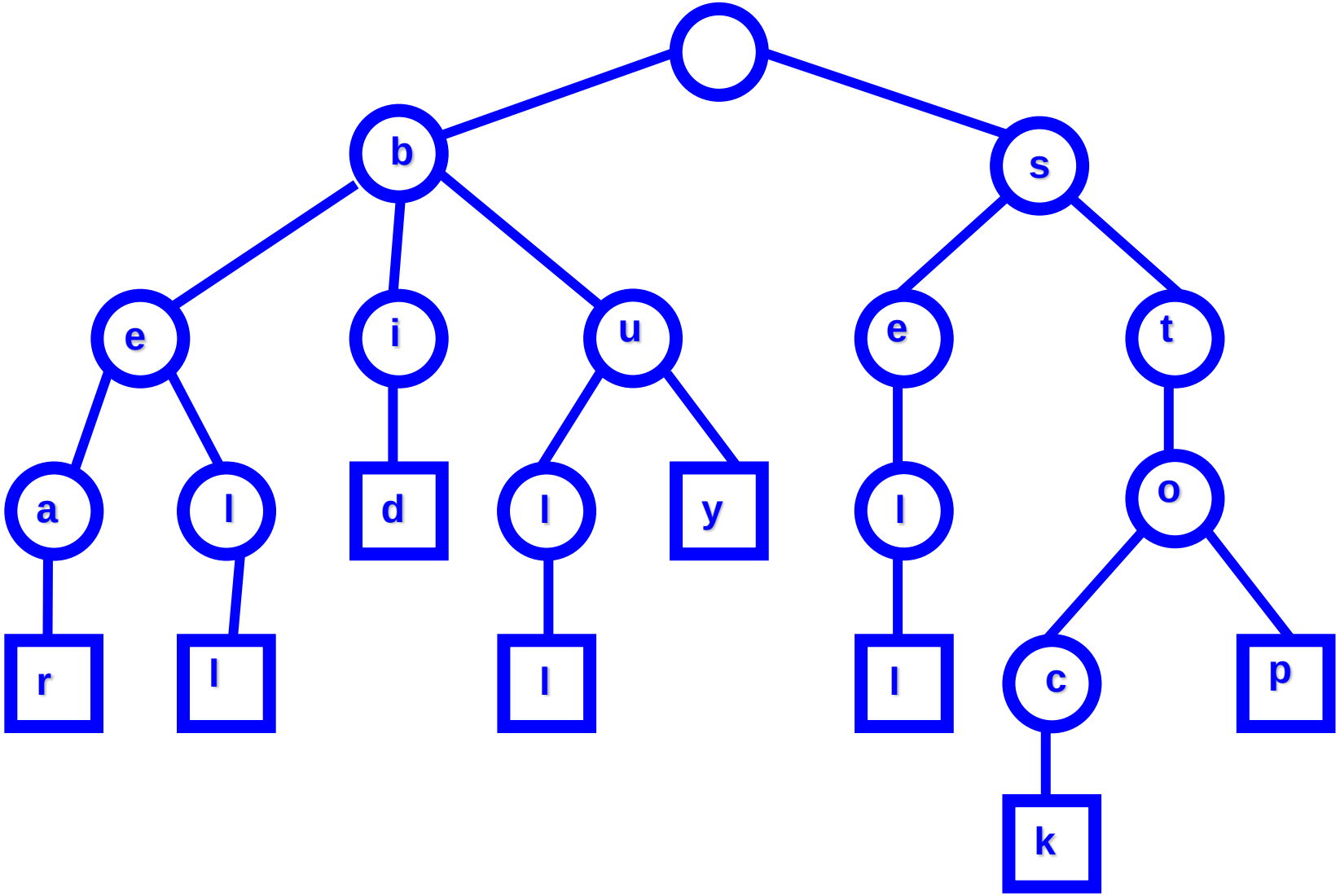
Exemplo



bear - urso
bell - sino
bid - oferta
bull - touro
buy - compra
sell - vende
stock - ação
stop - parar

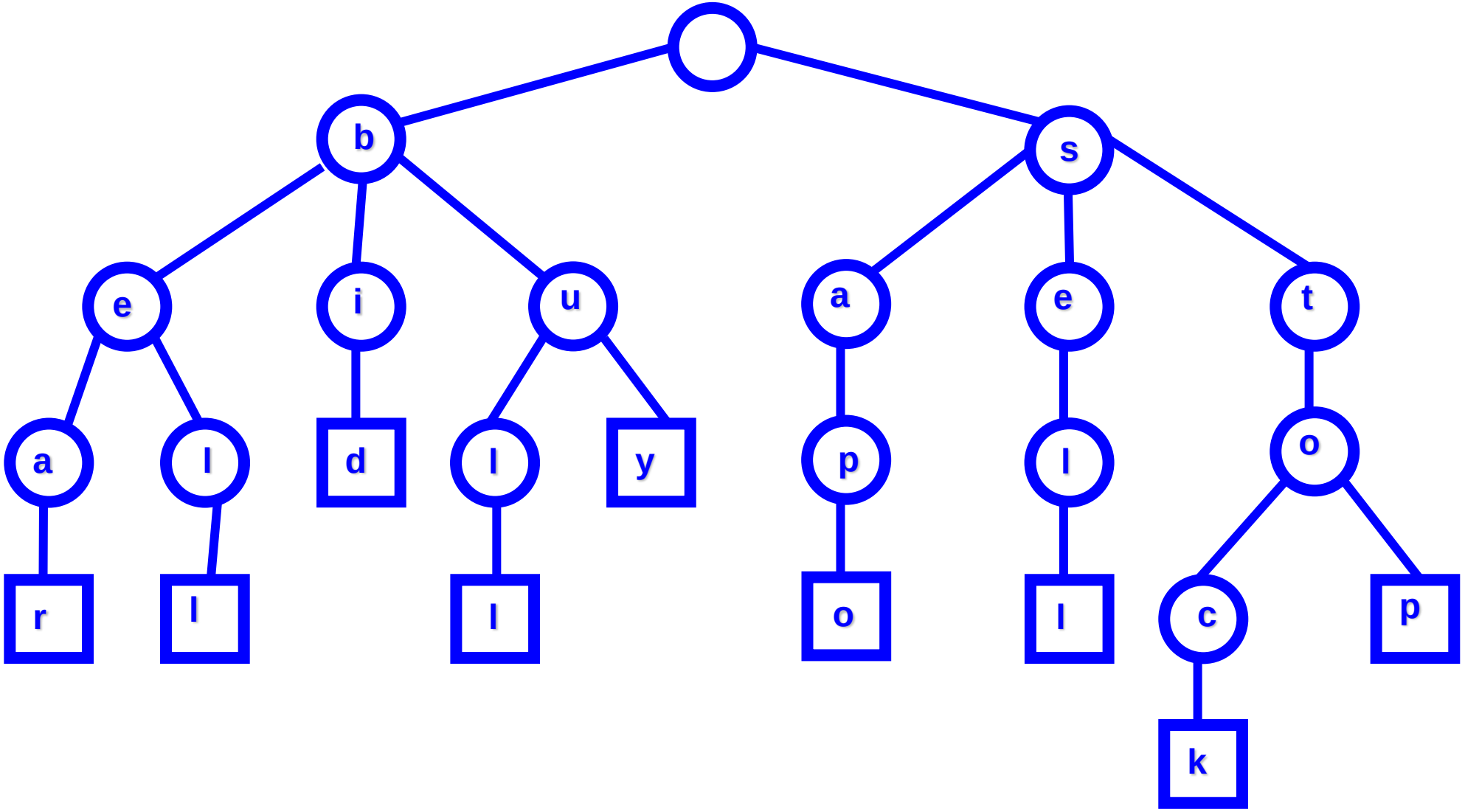
Exercício

- Insira as palavras sapo e sapato na árvore abaixo



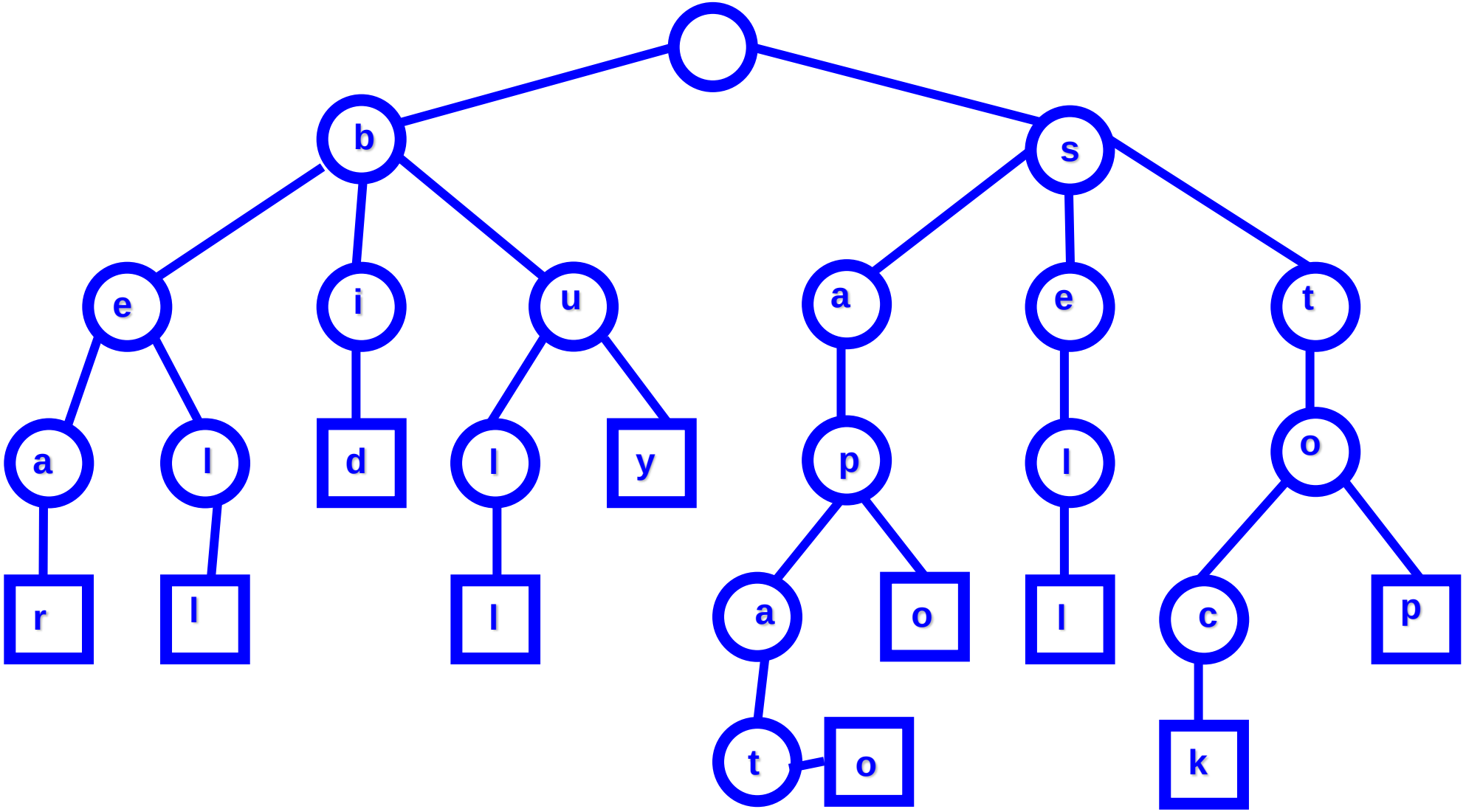
Exercício

- Insira as palavras **sapo** e sapato na árvore abaixo



Exercício

- Insira as palavras sapo e **sapato** na árvore abaixo



Propriedades

- Nenhuma cadeia de S é prefixo de outra cadeia
- Cada nó (exceto a raiz) é rotulado com um caractere do Σ
- A árvore tem s folhas, um para cada cadeia de S
- A concatenação dos rótulos em um caminho da raiz até uma folha, resulta na cadeia de S associada a essa folha

Propriedades

- Em geral, a *trie* é uma árvore múltipla (1...d filhos)
- Se o Σ tem tamanho d igual a 2, a *trie* será uma árvore binária
- Cada nó interno tem no máximo d filhos
- A altura da árvore é igual ao tamanho da maior cadeia em S

Propriedades

- O número de nós é O(n) sendo n o comprimento total de S

- O pior caso para o número de nós acontece quando não existe qualquer prefixo comum entre as cadeias, fazendo com que todos os nós internos (exceto a raiz) tenham um filho

Pesquisar por uma Cadeia de Caracteres

- A partir da raiz, verificamos caractere-a-caractere se existe um caminho na árvore correspondendo à cadeia desejada (por definição, um caminho sempre termina em uma folha)

- Se cada nó tiver uma tabela hash perfeita para endereçar seus filhos, o tempo de pesquisa é O(m) onde m é o tamanho da cadeia a ser procurada

Inserção de uma Cadeia de Caracteres

- Caminhamos na *trie* casando cada caractere da nova cadeia
- Quando não existe um nó para um caractere, criamos o nó e repetimos esse passo para os demais caracteres da cadeia
- Lembrando que nenhuma cadeia é prefixo de outra
- O tempo de inserção é O(m) e a construção total da árvore é O(n), onde n = |S|

Trie Patricia

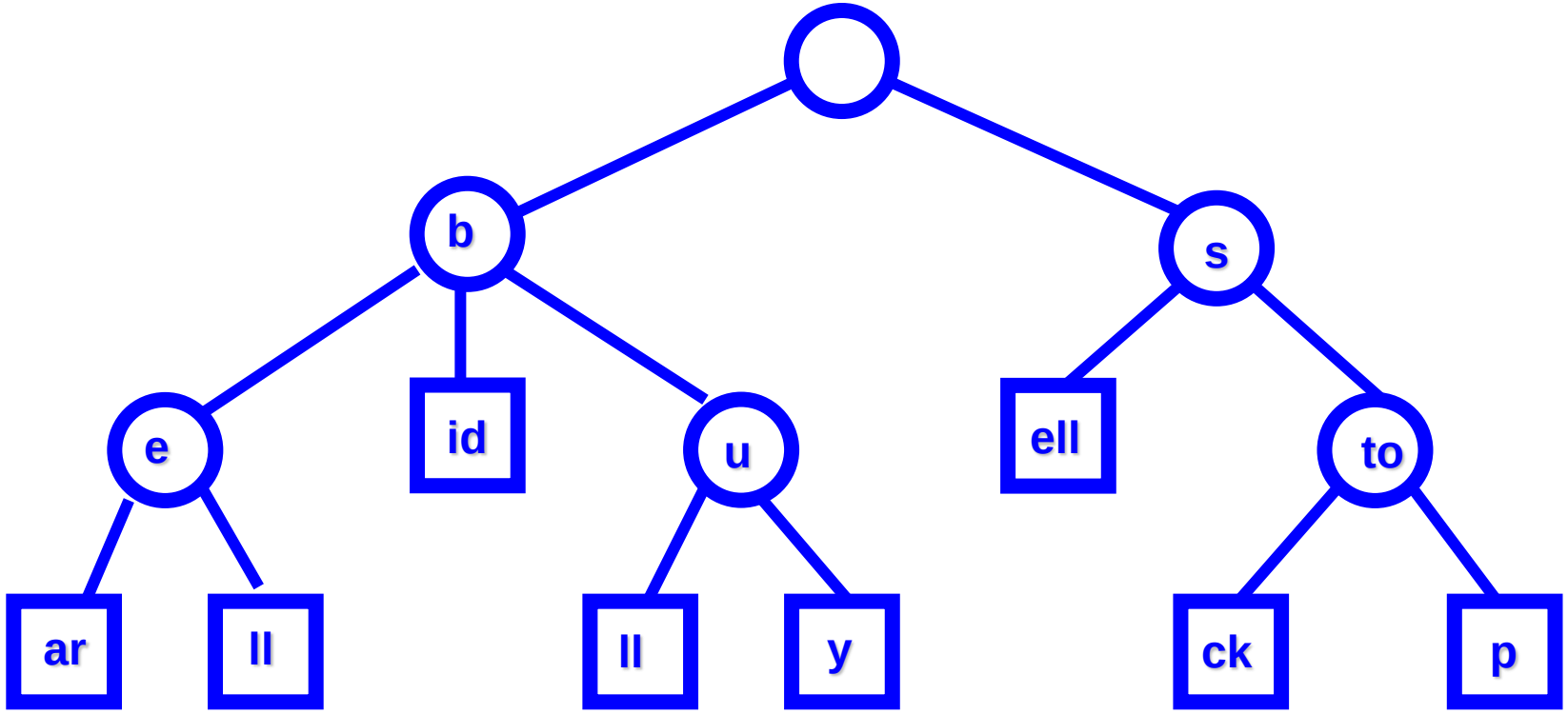
- Significa ***P****ractical* ***A****lgorithm* ***t****o* ***R****etrieve* ***I****nformation* ***C****oded* ***i****n* ***A****lphanumeric*

- Elimina os nós redundantes fazendo com que todos os nós (exceto a raiz) tenham pelo menos dois filhos

  - Na *trie-padrão*, a existência de nós com apenas um filho representa ineficiência em termos de espaço

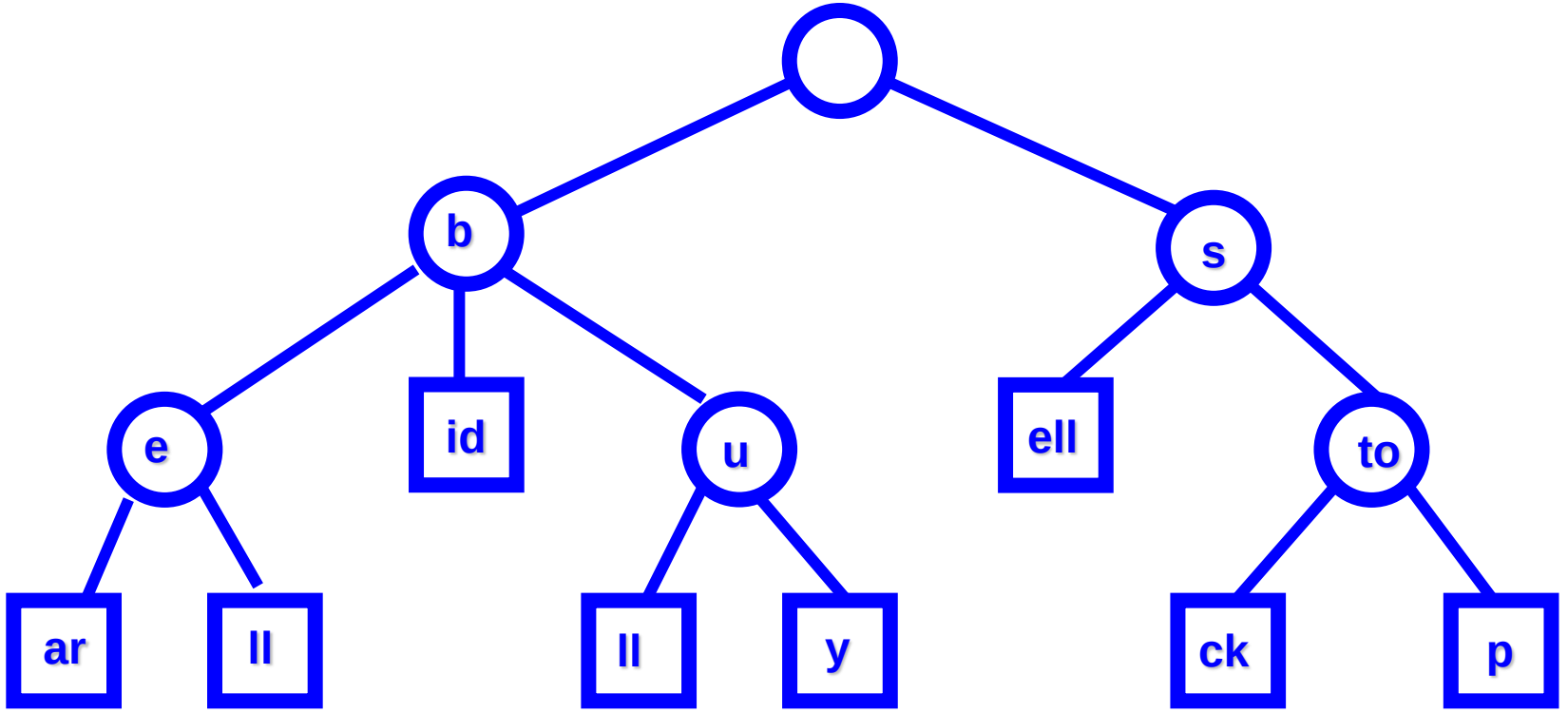
Exemplo de *Trie Patricia*

bear - urso
bell - sino
bid - oferta
bull - touro
buy - compra
sell - vende
stock - ação
stop - parar



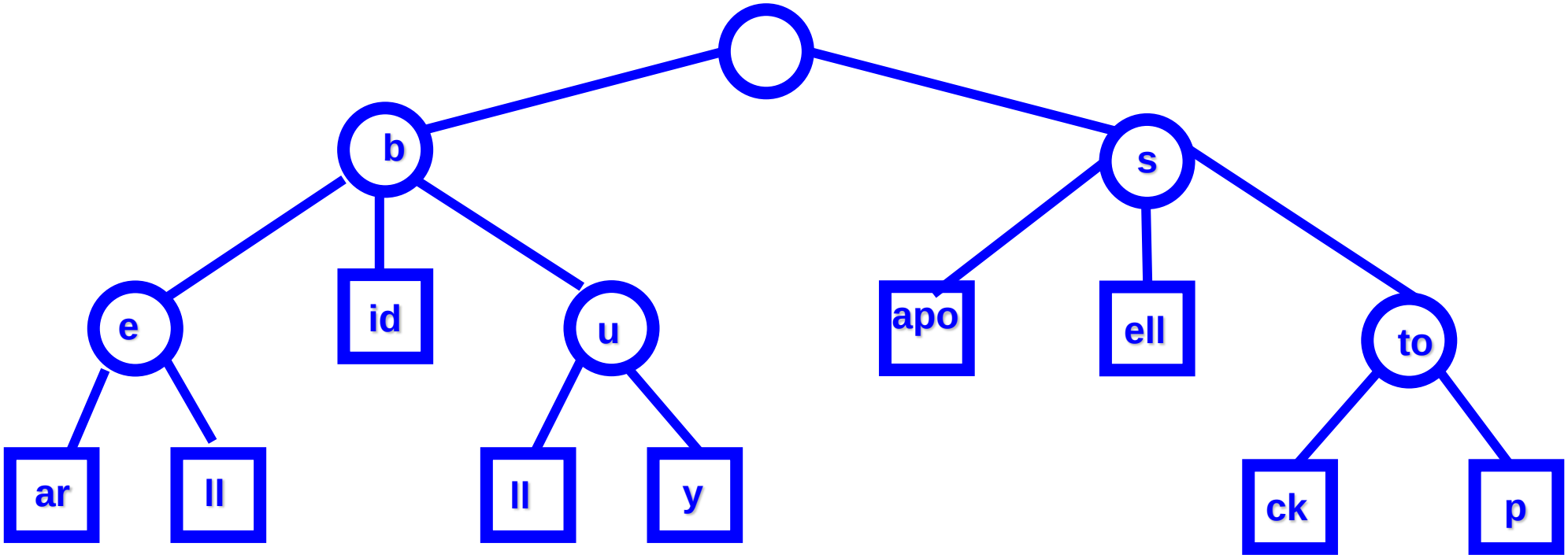
Exercício

- Insira as palavras sapo e sapato na árvore abaixo



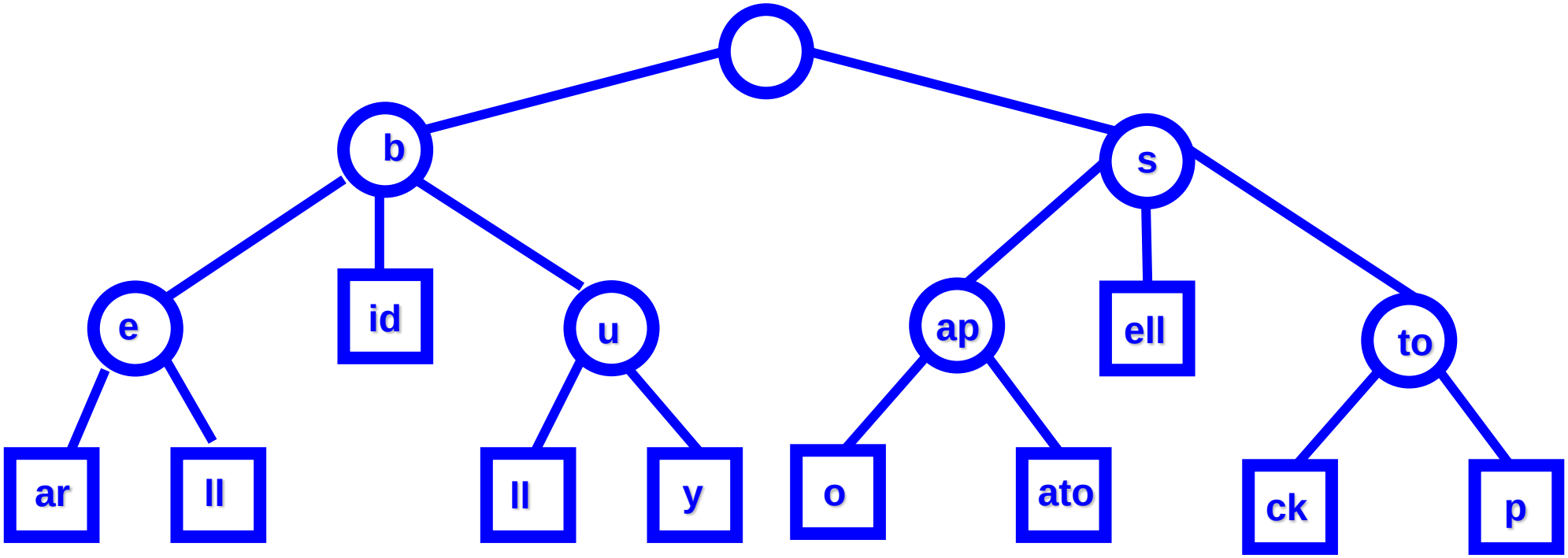
Exercício

- Insira as palavras **sapo** e sapato na árvore abaixo



Exercício

- Insira as palavras sapo e **sapato** na árvore abaixo



Propriedades das *Trie Patricia*

- Os nós são rotulados por *substrings* das cadeias de caracteres pertencentes a S

- O número de nós é proporcional ao número de cadeias existentes em S e não ao comprimento das mesmas
  - Todo nó interno tem entre 2 e d filhos
  - O(s) nós e |s| folhas, onde s é o número de cadeias

Estrutura de Dados das *Trie Patricia*

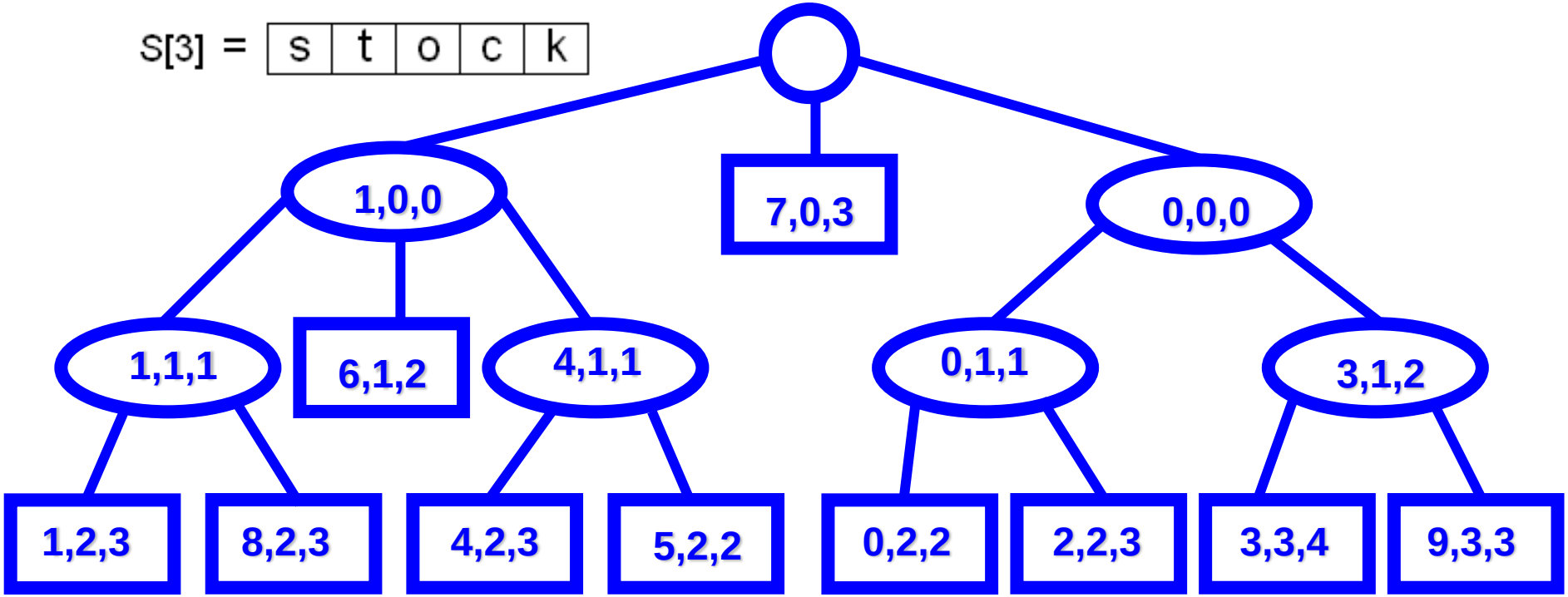
- Cada nó armazena uma tripla de inteiros (i, j, k), indicando o rótulo do nó de tal forma que S[i][j...k], onde:
  - A coleção de cadeias S será S[0], S[1],... S[s-1]
  - j e k representam, respectivamente, a primeira e última (inclusive) posições da cadeira S[i] que correspondem ao rótulo corrente

Exemplo da Estrutura de Dados da *Trie Patricia*

	0	1	2	3	4
S[0] =	s	e	e		
S[1] =	b	e	a	r	
S[2] =	s	e	l	l	
S[3] =	s	t	o	c	k

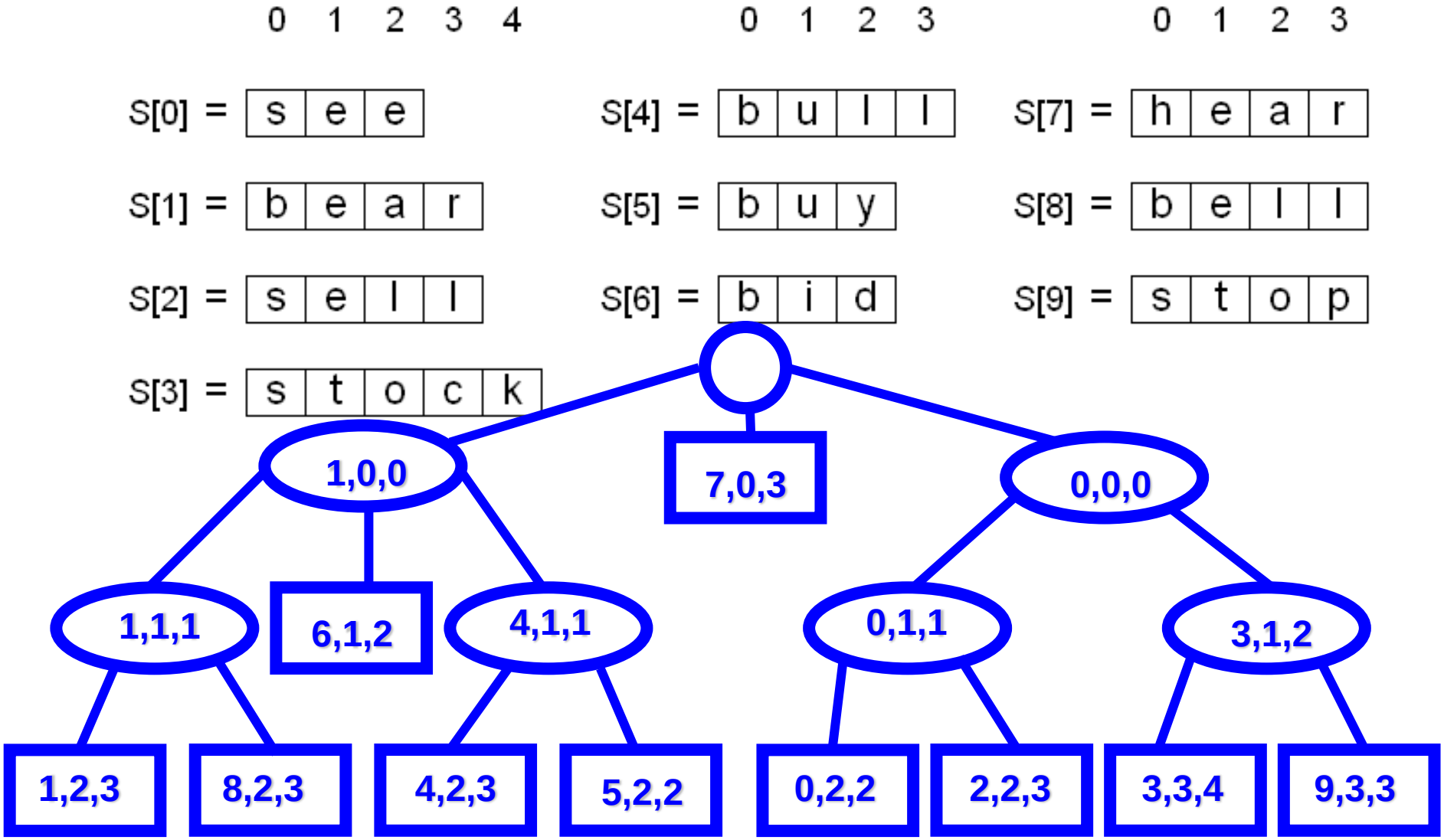
	0	1	2	3
S[4] =	b	u	l	l
S[5] =	b	u	y	
S[6] =	b	i	d	

	0	1	2	3
S[7] =	h	e	a	r
S[8] =	b	e	l	l
S[9] =	s	t	o	p



Exercício

- Insira as palavras sapo e sapato na árvore abaixo



Análise da *Trie Patricia*

- Complexidade de espaço: O(s) (na trie-padrão, O(n))

- Na prática, a *trie-patricia* tem ganhos em termos de espaço mesmo considerando o armazenando da coleção S