Tradução dirigida pela sintaxe

Definições dirigidas pela sintaxe

Prof. Edson Alves

Faculdade UnB Gama

Sumário

Definição dirigida pela sintaxe

Definição

Uma definição dirigida pela sintaxe é uma generalização de uma gramática livre de contexto na qual cada símbolo gramatical possui um conjunto de atributos, particionados em dois subconjuntos: atributos herdados e atributos sintetizados.

Atributos

- Se cada nó da árvore gramatical contém um registro para armazenar informações, cada atributo seria um membro deste registro

Atributos

- Se cada nó da árvore gramatical contém um registro para armazenar informações, cada atributo seria um membro deste registro
- Um atributo pode representar qualquer valor associado ao símbolo gramatical (um número, uma cadeia, um endereço de memória, etc)

Atributos

- Se cada nó da árvore gramatical contém um registro para armazenar informações, cada atributo seria um membro deste registro
- Um atributo pode representar qualquer valor associado ao símbolo gramatical (um número, uma cadeia, um endereço de memória, etc)
- O valor para um atributo de uma árvore gramatical é computado a partir de uma regra semântica associada à produção usada naquele nó

Atributos

- Se cada nó da árvore gramatical contém um registro para armazenar informações, cada atributo seria um membro deste registro
- Um atributo pode representar qualquer valor associado ao símbolo gramatical (um número, uma cadeia, um endereço de memória, etc)
- O valor para um atributo de uma árvore gramatical é computado a partir de uma regra semântica associada à produção usada naquele nó
- Um atributo sintetizado é computado a partir dos valores dos atributos dos filhos do nó

Atributos

- Se cada nó da árvore gramatical contém um registro para armazenar informações, cada atributo seria um membro deste registro
- Um atributo pode representar qualquer valor associado ao símbolo gramatical (um número, uma cadeia, um endereço de memória, etc)
- O valor para um atributo de uma árvore gramatical é computado a partir de uma regra semântica associada à produção usada naquele nó
- Um atributo sintetizado é computado a partir dos valores dos atributos dos filhos do nó
- Um atributo herdado é computado a partir dos valores dos atributos dos irmãos e do pai do nó

Regras semânticas

- As regras semânticas estabelecem dependências entre os atributos, as quais são representadas por meio de um grafo

Regras semânticas

- As regras semânticas estabelecem dependências entre os atributos, as quais são representadas por meio de um grafo
- A ordem de avaliação das regras gramaticais é derivada a partir do grafo de dependências

Regras semânticas

- As regras semânticas estabelecem dependências entre os atributos, as quais são representadas por meio de um grafo
- A ordem de avaliação das regras gramaticais é derivada a partir do grafo de dependências
- A avaliação das regras semânticas determina os valores dos atributos para os nós da árvore gramatical para uma dada cadeia de entrada

Regras semânticas

- As regras semânticas estabelecem dependências entre os atributos, as quais são representadas por meio de um grafo
- A ordem de avaliação das regras gramaticais é derivada a partir do grafo de dependências
- A avaliação das regras semânticas determina os valores dos atributos para os nós da árvore gramatical para uma dada cadeia de entrada
- Regras semânticas podem ter efeitos colaterais associados

Regras semânticas

- As regras semânticas estabelecem dependências entre os atributos, as quais são representadas por meio de um grafo
- A ordem de avaliação das regras gramaticais é derivada a partir do grafo de dependências
- A avaliação das regras semânticas determina os valores dos atributos para os nós da árvore gramatical para uma dada cadeia de entrada
- Regras semânticas podem ter efeitos colaterais associados
- Na prática, a árvore semãntica ou o grafo de dependências não precisam ser construídos explicitamente

Regras semânticas

- As regras semânticas estabelecem dependências entre os atributos, as quais são representadas por meio de um grafo
- A ordem de avaliação das regras gramaticais é derivada a partir do grafo de dependências
- A avaliação das regras semânticas determina os valores dos atributos para os nós da árvore gramatical para uma dada cadeia de entrada
- Regras semânticas podem ter efeitos colaterais associados
- Na prática, a árvore semãntica ou o grafo de dependências não precisam ser construídos explicitamente
- Uma árvore gramatical que exibe os valores dos atributos de cada nó é denominada árvore gramatical anotada

Regras semânticas

- As regras semânticas estabelecem dependências entre os atributos, as quais são representadas por meio de um grafo
- A ordem de avaliação das regras gramaticais é derivada a partir do grafo de dependências
- A avaliação das regras semânticas determina os valores dos atributos para os nós da árvore gramatical para uma dada cadeia de entrada
- Regras semânticas podem ter efeitos colaterais associados
- Na prática, a árvore semãntica ou o grafo de dependências não precisam ser construídos explicitamente
- Uma árvore gramatical que exibe os valores dos atributos de cada nó é denominada árvore gramatical anotada
- O processo de computador os valores dos atributos é denominado anotação da árvore

Gramática de atributos

Definição

Seja uma definição dirigida pela sintaxe. Cada produção $A \to \alpha$ estará associada a um conjunto de regras semânticas da forma $b := f(c_1, c_2, \ldots, c_k)$ onde f é uma função, $c_1, c_2, \ldots, c_k$ são atributos pertencentes aos símbolos gramaticais da produção e vale apenas uma das duas alternativas:

(i) b é um atributo sintetizado de A

(ii) b é um atributo herdado, pertencente ao um dos símbolos do lado direito da produção

Em ambos casos, b depende dos atributos $c_1, c_2, \ldots, c_k$.

Se, para qualquer atributo b, a função f não possui efeitos colaterais, então esta definição dirigida pela sintaxe é denominada gramática de atributos.

Exemplo de definição dirigida pela sintaxe

Produção	Regra semântica
$L \rightarrow E$ **n**	IMPRIMIR$(E.val)$
$E \rightarrow E_1$ + T	$E.val := E_1.val + T.val$
$E \rightarrow T$	$E.val := T.val$
$T \rightarrow T_1 \times F$	$T.val := T_1.val \times F.val$
$T \rightarrow F$	$T.val := F.val$
$F \rightarrow (E)$	$F.val := E.val$
$F \rightarrow$ **digito**	$F.val :=$ **digito**$.lexval$

Nesta definição dirigida pela sintaxe para uma calculadora, L representa uma linha, **n** uma quebra de linha e o atributo $lexval$ do terminal **digito** é determinado pelo analisador léxico.

Atributos sintetizados

- Na prática, os atributos sintetizados são usados extensivamente

Atributos sintetizados

- Na prática, os atributos sintetizados são usados extensivamente
- Uma definição dirigida pela sintaxe que utilize exclusivamente atributos sintetizados é denominada uma definição S-atribuída

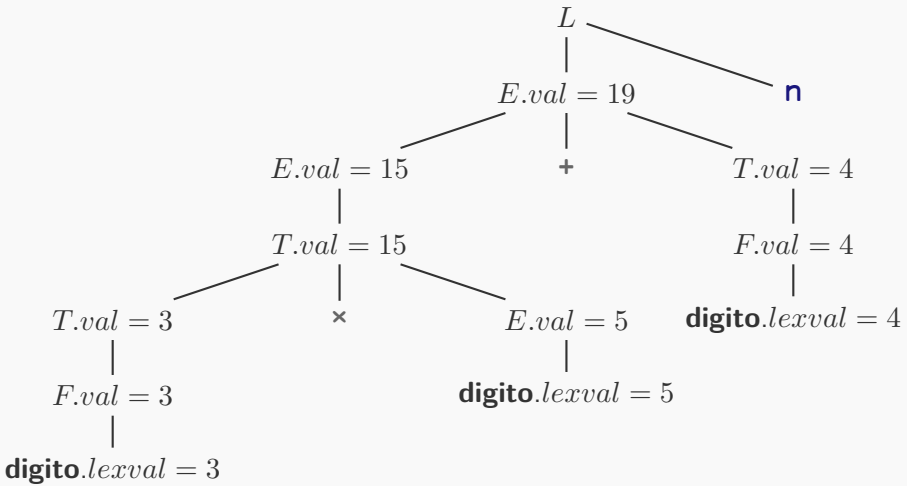
Atributos sintetizados

- Na prática, os atributos sintetizados são usados extensivamente
- Uma definição dirigida pela sintaxe que utilize exclusivamente atributos sintetizados é denominada uma definição S-atribuída
- Em uma árvore gramatical para uma definição S-atribuída, os valores dos atributos dos nós podem ser todos computados avaliando-se as regras gramaticais dos nós, a partir das folhas para a raiz

Atributos sintetizados

- Na prática, os atributos sintetizados são usados extensivamente
- Uma definição dirigida pela sintaxe que utilize exclusivamente atributos sintetizados é denominada uma definição S-atribuída
- Em uma árvore gramatical para uma definição S-atribuída, os valores dos atributos dos nós podem ser todos computados avaliando-se as regras gramaticais dos nós, a partir das folhas para a raiz
- No exemplo anterior, todos os atributos da definição dirigida pela sintaxe são sintetizados

Árvore gramatical anotada para a expressão 3×5+4n



Atributos herdados

- Um atributo de um nó de uma árvore gramatical é dito herdado se o seu valor é definido a partir dos valores dos atributos de seu pai e/ou de seus irmãos

Atributos herdados

- Um atributo de um nó de uma árvore gramatical é dito herdado se o seu valor é definido a partir dos valores dos atributos de seu pai e/ou de seus irmãos
- Tais atributos são úteis para representar relações de dependência de um construção de uma linguagem de programação com o contexto onde ele ocorre

Atributos herdados

- Um atributo de um nó de uma árvore gramatical é dito herdado se o seu valor é definido a partir dos valores dos atributos de seu pai e/ou de seus irmãos
- Tais atributos são úteis para representar relações de dependência de um construção de uma linguagem de programação com o contexto onde ele ocorre
- Por exemplo, com atributos herdados é possível determinar se um identificador aparece do lado esquerdo ou direito de uma atribuição

Atributos herdados

- Um atributo de um nó de uma árvore gramatical é dito herdado se o seu valor é definido a partir dos valores dos atributos de seu pai e/ou de seus irmãos
- Tais atributos são úteis para representar relações de dependência de um construção de uma linguagem de programação com o contexto onde ele ocorre
- Por exemplo, com atributos herdados é possível determinar se um identificador aparece do lado esquerdo ou direito de uma atribuição
- É possível reescrever uma definição dirigida pela sintaxe de modo que sejam usados apenas atributos sintetizados

Atributos herdados

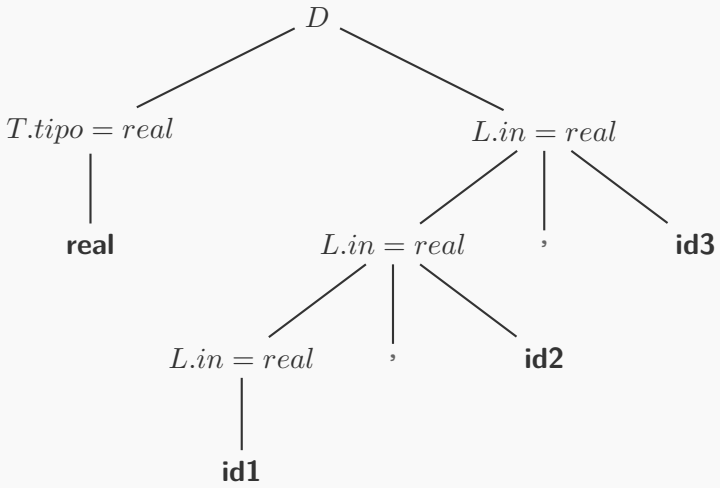
- Um atributo de um nó de uma árvore gramatical é dito herdado se o seu valor é definido a partir dos valores dos atributos de seu pai e/ou de seus irmãos
- Tais atributos são úteis para representar relações de dependência de um construção de uma linguagem de programação com o contexto onde ele ocorre
- Por exemplo, com atributos herdados é possível determinar se um identificador aparece do lado esquerdo ou direito de uma atribuição
- É possível reescrever uma definição dirigida pela sintaxe de modo que sejam usados apenas atributos sintetizados
- Contudo, o uso de atributos herdados permite uma descrição mais natural das relações de dependência

Exemplo de definição dirigida pela sintaxe com atributo herdado

Produção	Regra semântica
$D \rightarrow TL$	$L.in := T.tipo$
$T \rightarrow$ **int**	$T.tipo = inteiro$
$T \rightarrow$ **real**	$T.tipo = real$
$L \rightarrow L_1,$ **id**	$L_1.in := L.in$ INCLUIRTIPO(**id**, $entrada$, $L.in$)
$L \rightarrow$ **id**	INCLUIRTIPO(**id**, $entrada$, $L.in$)

A definição dirigida pela sintaxe acima gera definições D onde a palavra-chave **int** ou **real** precedem uma lista de identificadores. O atributo $L.in$ é herdado, enquanto que o atributo $T.tipo$ é sintetizado.

Árvore gramatical da definição `real id1, id2, id3`



Grafo de dependências

Definição

O grafo que estabelece as relações entre os diferentes atributos de uma definição dirigida pela sintaxe, onde os nós são os atributos e uma aresta (a, b) indica que o atributo a deve ser determinado antes do atributo b, é denominado grafo de dependências.

Na construção de um grafo de dependências, deve ser inserido um nó para cada chamada de procedimento, o que corresponde à introdução de um atributo fictício associado a esta chamada.

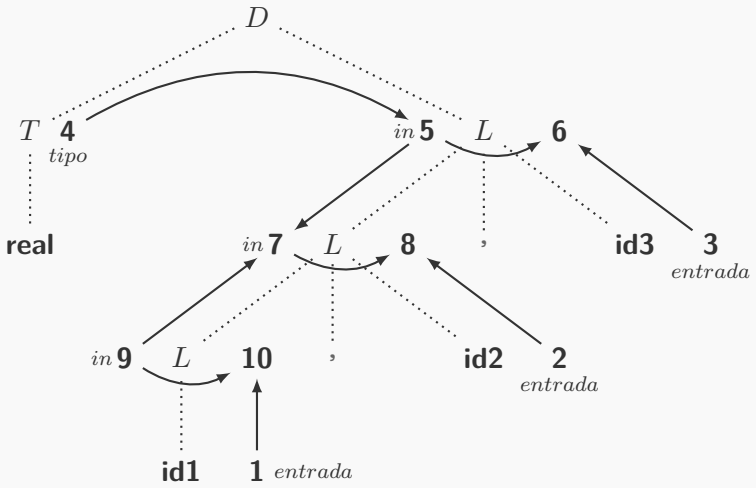
Algoritmo para geração do grafo de dependências

Input: Uma árvore gramatical de uma definição dirigida pela sintaxe
Output: O grafo de dependência

1: **for** cada n da árvore gramatical **do**
2: **for** cada atributo a do símbolo gramatical em n **do**
3: construa um nó no grafo de dependências para a

4: **for** cada n da árvore gramatical **do**
5: **for** cada regra semântica $b := f(c_1, c_2, \ldots, c_k)$ associada a produção usada em n **do**
6: **for** $i = 1, k$ **do**
7: adicione ao grafo uma aresta partindo de c_i para b

Grafo de dependência da definição `real id1, id2, id3`



Ordenação topológica

Definição

Seja $G(V, E)$ um grafo direcionado acíclico. Uma ordenação topológica de G é uma sequência de vértices $v_1, v_2, \ldots, v_N$ tal que, se $(v_i, v_j) \in E$, então v_i antecede v_j na sequência.

Ordenação topológica

Definição

Seja $G(V, E)$ um grafo direcionado acíclico. Uma ordenação topológica de G é uma sequência de vértices $v_1, v_2, \ldots, v_N$ tal que, se $(v_i, v_j) \in E$, então v_i antecede v_j na sequência.

Qualquer ordenação topológica do grafo de dependências fornece uma ordem válida de avaliação das regras gramaticais que definem os valores dos atributos, uma vez que, quando a regra $b := f(c_1, c_2, \ldots, c_k)$ for avaliada, os atributos $c_1, c_2, \ldots, c_k$ já terão seus valores definidos.

Tradução a partir de uma definição dirigida pela sintaxe

Pode-se definir uma tradução a partir de uma definição dirigida pela sintaxe da seguinte forma:

Tradução a partir de uma definição dirigida pela sintaxe

Pode-se definir uma tradução a partir de uma definição dirigida pela sintaxe da seguinte forma:

1. Construa a árvore gramatical da entrada e acordo com a gramática subjacente

Tradução a partir de uma definição dirigida pela sintaxe

Pode-se definir uma tradução a partir de uma definição dirigida pela sintaxe da seguinte forma:

1. Construa a árvore gramatical da entrada e acordo com a gramática subjacente
2. Construa o grafo de dependências

Tradução a partir de uma definição dirigida pela sintaxe

Pode-se definir uma tradução a partir de uma definição dirigida pela sintaxe da seguinte forma:

1. Construa a árvore gramatical da entrada e acordo com a gramática subjacente
2. Construa o grafo de dependências
3. Gere uma ordenação topológica do grafo de dependências

Tradução a partir de uma definição dirigida pela sintaxe

Pode-se definir uma tradução a partir de uma definição dirigida pela sintaxe da seguinte forma:

1. Construa a árvore gramatical da entrada e acordo com a gramática subjacente
2. Construa o grafo de dependências
3. Gere uma ordenação topológica do grafo de dependências
4. Use a ordenação topológica para avaliar as regras semânticas

Tradução a partir de uma definição dirigida pela sintaxe

Pode-se definir uma tradução a partir de uma definição dirigida pela sintaxe da seguinte forma:

1. Construa a árvore gramatical da entrada e acordo com a gramática subjacente
2. Construa o grafo de dependências
3. Gere uma ordenação topológica do grafo de dependências
4. Use a ordenação topológica para avaliar as regras semânticas
5. A avaliação das regras semânticas produzirá a tradução da entrada

Construção de árvores sintáticas para expressões

- Árvores sintáticas para expressões podem ser construídas de forma semelhante à tradução para notação posfixa

Construção de árvores sintáticas para expressões

- Árvores sintáticas para expressões podem ser construídas de forma semelhante à tradução para notação posfixa
- Deve ser construído um nó para cada operação e cada operando

Construção de árvores sintáticas para expressões

- Árvores sintáticas para expressões podem ser construídas de forma semelhante à tradução para notação posfixa
- Deve ser construído um nó para cada operação e cada operando
- Os filhos do nó de um operador ser subárvores que representam as subexpressões que constituem os operandos daquele operador

Construção de árvores sintáticas para expressões

- Árvores sintáticas para expressões podem ser construídas de forma semelhante à tradução para notação posfixa
- Deve ser construído um nó para cada operação e cada operando
- Os filhos do nó de um operador ser subárvores que representam as subexpressões que constituem os operandos daquele operador
- Cada nó pode ser implementado como um registro com vários campos que caracterizam o nó

Construção de árvores sintáticas para expressões

- Árvores sintáticas para expressões podem ser construídas de forma semelhante à tradução para notação posfixa
- Deve ser construído um nó para cada operação e cada operando
- Os filhos do nó de um operador ser subárvores que representam as subexpressões que constituem os operandos daquele operador
- Cada nó pode ser implementado como um registro com vários campos que caracterizam o nó
- O registro de nós que representam operadores devem conter um campo que identifica o operador e os demais campos devem ser ponteiros para os operandos

Construção de árvores sintáticas para expressões

- Árvores sintáticas para expressões podem ser construídas de forma semelhante à tradução para notação posfixa
- Deve ser construído um nó para cada operação e cada operando
- Os filhos do nó de um operador ser subárvores que representam as subexpressões que constituem os operandos daquele operador
- Cada nó pode ser implementado como um registro com vários campos que caracterizam o nó
- O registro de nós que representam operadores devem conter um campo que identifica o operador e os demais campos devem ser ponteiros para os operandos
- As folhas das árvores contém os tokens

Construção de árvores sintáticas para expressões

- Árvores sintáticas para expressões podem ser construídas de forma semelhante à tradução para notação posfixa
- Deve ser construído um nó para cada operação e cada operando
- Os filhos do nó de um operador ser subárvores que representam as subexpressões que constituem os operandos daquele operador
- Cada nó pode ser implementado como um registro com vários campos que caracterizam o nó
- O registro de nós que representam operadores devem conter um campo que identifica o operador e os demais campos devem ser ponteiros para os operandos
- As folhas das árvores contém os tokens
- O registro de uma folha deve identificar o token e também armazenar um ponteiro para a entrada do token na tabela de símbolos

Funções para a criação de nós da árvore sintática de uma expressão

Cada uma das funções abaixo retorna um ponteiro para o nó criado. Assuma que os operadores são todos binários.

Funções para a criação de nós da árvore sintática de uma expressão

Cada uma das funções abaixo retorna um ponteiro para o nó criado. Assuma que os operadores são todos binários.

1. $\textsc{criarNo}(op, L, R)$: cria um nó de operador cujo rótulo é op, L é o ponteiro do operando à esquerda e R o ponteiro do operando à direita

Funções para a criação de nós da árvore sintática de uma expressão

Cada uma das funções abaixo retorna um ponteiro para o nó criado. Assuma que os operadores são todos binários.

1. criarNo(op, L, R): cria um nó de operador cujo rótulo é op, L é o ponteiro do operando à esquerda e R o ponteiro do operando à direita
2. criarFolha(**id**, p): cria um nó para um identificador com rótulo **id**, onde p é o ponteiro para o identificador na tabela de símbolos

Funções para a criação de nós da árvore sintática de uma expressão

Cada uma das funções abaixo retorna um ponteiro para o nó criado. Assuma que os operadores são todos binários.

1. CRIARNO(op, L, R): cria um nó de operador cujo rótulo é op, L é o ponteiro do operando à esquerda e R o ponteiro do operando à direita
2. CRIARFOLHA(**id**, p): cria um nó para um identificador com rótulo **id**, onde p é o ponteiro para o identificador na tabela de símbolos
3. CRIARFOLHA(**num**, val): cria um nó para um número, com rótulo **num**, cujo valor é indicado por val

Criação da árvore sintática da expressão a - 4 + c

Chamadas de funções

Criação da árvore sintática da expressão `a - 4 + c`

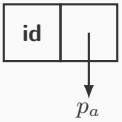
Chamadas de funções

$$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$$

Criação da árvore sintática da expressão a - 4 + c

Chamadas de funções

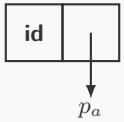
$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$



Criação da árvore sintática da expressão a - 4 + c

Chamadas de funções

$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{num}, 4)$



Criação da árvore sintática da expressão a - 4 + c

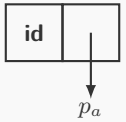
Chamadas de funções

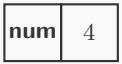
$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{num}, 4)$

Criação da árvore sintática da expressão a - 4 + c

Chamadas de funções

$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{num}, 4)$
$p_3 := \text{CRIARNO}(+, p_1, p_2)$

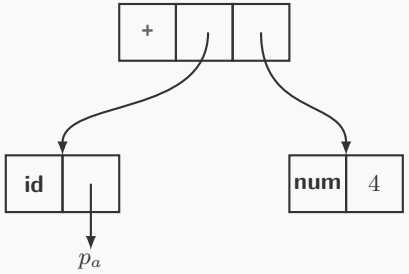




Criação da árvore sintática da expressão `a - 4 + c`

Chamadas de funções

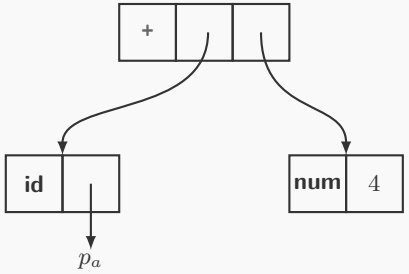
$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{num}, 4)$
$p_3 := \text{CRIARNO}(+, p_1, p_2)$



Criação da árvore sintática da expressão a - 4 + c

Chamadas de funções

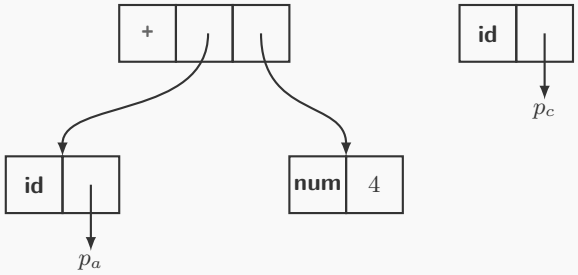
$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{num}, 4)$
$p_3 := \text{CRIARNO}(+, p_1, p_2)$
$p_4 := \text{CRIARFOLHA}(\textbf{id}, p_c)$



Criação da árvore sintática da expressão a - 4 + c

Chamadas de funções

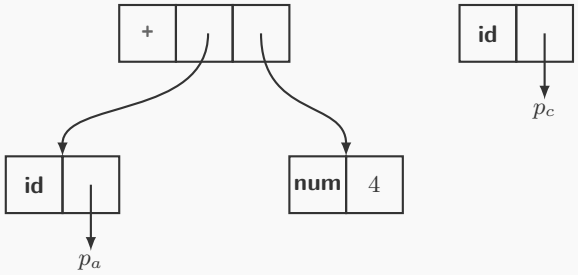
$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{num}, 4)$
$p_3 := \text{CRIARNO}(+, p_1, p_2)$
$p_4 := \text{CRIARFOLHA}(\textbf{id}, p_c)$



Criação da árvore sintática da expressão a - 4 + c

Chamadas de funções

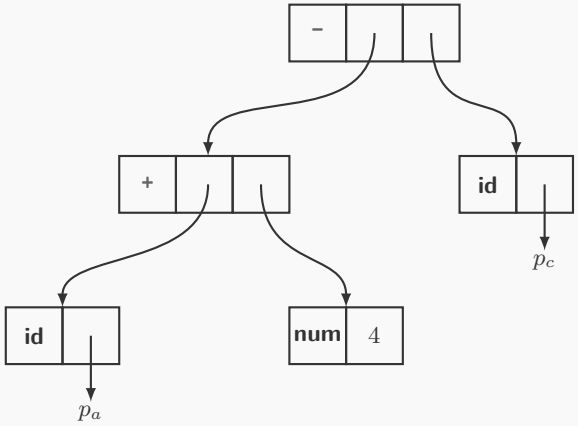
$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{num}, 4)$
$p_3 := \text{CRIARNO}(+, p_1, p_2)$
$p_4 := \text{CRIARFOLHA}(\textbf{id}, p_c)$
$p_5 := \text{CRIARNO}(-, p_3, p_4)$



Criação da árvore sintática da expressão `a - 4 + c`

Chamadas de funções

$$
\begin{aligned}
p_1 &:= \textsc{criarFolha}(\textbf{id}, p_a) \\
p_2 &:= \textsc{criarFolha}(\textbf{num}, 4) \\
p_3 &:= \textsc{criarNo}(+, p_1, p_2) \\
p_4 &:= \textsc{criarFolha}(\textbf{id}, p_c) \\
p_5 &:= \textsc{criarNo}(-, p_3, p_4)
\end{aligned}
$$



Definição dirigida pela sintaxe para a construção de árvores sintáticas

- É possível construir árvores sintáticas para expressões por meio de uma definição S-atribuída

Definição dirigida pela sintaxe para a construção de árvores sintáticas

- É possível construir árvores sintáticas para expressões por meio de uma definição S-atribuída
- As regras semânticas agendam as chamadas das funções de criação de nós que irão construir a árvore

Definição dirigida pela sintaxe para a construção de árvores sintáticas

- É possível construir árvores sintáticas para expressões por meio de uma definição S-atribuída
- As regras semânticas agendam as chamadas das funções de criação de nós que irão construir a árvore
- O atributo sintetizado $nptr$ controla os ponteiros para os nós retornados pelas funções

Definição dirigida pela sintaxe para a construção de árvores sintáticas

- É possível construir árvores sintáticas para expressões por meio de uma definição S-atribuída
- As regras semânticas agendam as chamadas das funções de criação de nós que irão construir a árvore
- O atributo sintetizado $nptr$ controla os ponteiros para os nós retornados pelas funções
- O atributo $entrada$ armazena o endereço de um token na tabela de símbolos e o atributo val o valor de um número

Definição dirigida pela sintaxe para a construção de árvores sintáticas

- É possível construir árvores sintáticas para expressões por meio de uma definição S-atribuída
- As regras semânticas agendam as chamadas das funções de criação de nós que irão construir a árvore
- O atributo sintetizado $nptr$ controla os ponteiros para os nós retornados pelas funções
- O atributo $entrada$ armazena o endereço de um token na tabela de símbolos e o atributo val o valor de um número
- Estes dois atributos devem ser computados na análise léxica

Definição dirigida pela sintaxe para expressões aritméticas de adição e subtração

Produção	Regra semântica
$E \rightarrow E_1$ + T	$E.nptr := \text{CRIARNO}(+, E_1.nptr, T.nptr)$
$E \rightarrow E_1$ - T	$E.nptr := \text{CRIARNO}(-, E_1.nptr, T.nptr)$
$E \rightarrow T$	$E.nptr := T.nptr$
$T \rightarrow (E)$	$T.nptr := E.nptr$
$T \rightarrow$ **id**	$T.nptr := \text{CRIARNO}(\textbf{id}, \textbf{id}.entrada)$
$T \rightarrow$ **num**	$T.nptr := \text{CRIARNO}(\textbf{num}, \textbf{num}.val)$

DAG

- Um grafo direcionado acíclico (*directed acyclic graph – DAG*) é um grafo cujas arestas são direcionadas e que não possui ciclos

DAG

- Um grafo direcionado acíclico (*directed acyclic graph – DAG*) é um grafo cujas arestas são direcionadas e que não possui ciclos
- Um DAG pode ser usado para identificar subexpressões comuns em uma expressão

DAG

- Um grafo direcionado acíclico (*directed acyclic graph – DAG*) é um grafo cujas arestas são direcionadas e que não possui ciclos
- Um DAG pode ser usado para identificar subexpressões comuns em uma expressão
- De forma similar às árvores sintáticas, um nó representa um operador e seus filhos representam os operandos

DAG

- Um grafo direcionado acíclico (*directed acyclic graph – DAG*) é um grafo cujas arestas são direcionadas e que não possui ciclos
- Um DAG pode ser usado para identificar subexpressões comuns em uma expressão
- De forma similar às árvores sintáticas, um nó representa um operador e seus filhos representam os operandos
- Se houver uma ou mais expressões comuns, os nós do DAG podem ter "mais de um pai"

DAG

- Um grafo direcionado acíclico (*directed acyclic graph – DAG*) é um grafo cujas arestas são direcionadas e que não possui ciclos
- Um DAG pode ser usado para identificar subexpressões comuns em uma expressão
- De forma similar às árvores sintáticas, um nó representa um operador e seus filhos representam os operandos
- Se houver uma ou mais expressões comuns, os nós do DAG podem ter "mais de um pai"
- Nas árvores sintáticas, expressões comuns são duplicadas na árvore

Construção do DAG a partir de uma definição S-atribuída

- Uma definição S-atribuída para a construção de árvores sintáticas para expressões aritméticas de adições e subtrações pode se adaptada para a construção do DAG

Construção do DAG a partir de uma definição S-atribuída

- Uma definição S-atribuída para a construção de árvores sintáticas para expressões aritméticas de adições e subtrações pode se adaptada para a construção do DAG
- De fato, basta modificar o comportamento das funções CRIARNO( ) e CRIARFOLHA( )

Construção do DAG a partir de uma definição S-atribuída

- Uma definição S-atribuída para a construção de árvores sintáticas para expressões aritméticas de adições e subtrações pode se adaptada para a construção do DAG
- De fato, basta modificar o comportamento das funções CRIARNO( ) e CRIARFOLHA( )
- Ao invés de criar um novo nó a cada chamada, estas funções devem verificar se os parâmatros passados já não foram usados para construir um nó

Construção do DAG a partir de uma definição S-atribuída

- Uma definição S-atribuída para a construção de árvores sintáticas para expressões aritméticas de adições e subtrações pode se adaptada para a construção do DAG
- De fato, basta modificar o comportamento das funções CRIARNO( ) e CRIARFOLHA( )
- Ao invés de criar um novo nó a cada chamada, estas funções devem verificar se os parâmatros passados já não foram usados para construir um nó
- Em caso afirmativo, as funções devem retornar o ponteiro usado anteriormente na criação do nó

Construção do DAG a partir de uma definição S-atribuída

- Uma definição S-atribuída para a construção de árvores sintáticas para expressões aritméticas de adições e subtrações pode se adaptada para a construção do DAG
- De fato, basta modificar o comportamento das funções CRIARNO( ) e CRIARFOLHA( )
- Ao invés de criar um novo nó a cada chamada, estas funções devem verificar se os parâmatros passados já não foram usados para construir um nó
- Em caso afirmativo, as funções devem retornar o ponteiro usado anteriormente na criação do nó
- Caso contrário, deve ser criado um novo nó e o ponteiro criado deve ser armazenado em uma tabela, associado aos parâmetros usados, para consulta posterior

Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$

Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$

a

Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{id}, p_a)$

a

Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_3 := \text{CRIARFOLHA}(\textbf{id}, p_b)$

a

Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_3 := \text{CRIARFOLHA}(\textbf{id}, p_b)$

a

b

Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_3 := \text{CRIARFOLHA}(\textbf{id}, p_b)$
$p_4 := \text{CRIARFOLHA}(\textbf{id}, p_c)$

a

b

Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_3 := \text{CRIARFOLHA}(\textbf{id}, p_b)$
$p_4 := \text{CRIARFOLHA}(\textbf{id}, p_c)$

a

b c

Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_3 := \text{CRIARFOLHA}(\textbf{id}, p_b)$
$p_4 := \text{CRIARFOLHA}(\textbf{id}, p_c)$
$p_5 := \text{CRIARNO}(-, p_3, p_4)$

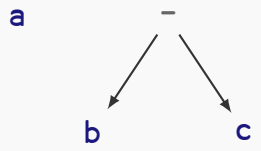
a

b c

Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

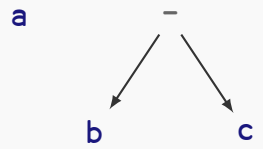
$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_3 := \text{CRIARFOLHA}(\textbf{id}, p_b)$
$p_4 := \text{CRIARFOLHA}(\textbf{id}, p_c)$
$p_5 := \text{CRIARNO}(-, p_3, p_4)$



Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

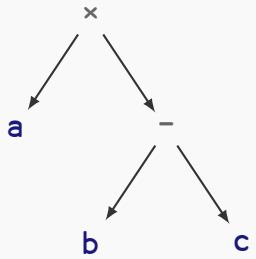
$$
\begin{aligned}
p_1 &:= \text{CRIARFOLHA}(\textbf{id}, p_a) \\
p_2 &:= \text{CRIARFOLHA}(\textbf{id}, p_a) \\
p_3 &:= \text{CRIARFOLHA}(\textbf{id}, p_b) \\
p_4 &:= \text{CRIARFOLHA}(\textbf{id}, p_c) \\
p_5 &:= \text{CRIARNO}(-, p_3, p_4) \\
p_6 &:= \text{CRIARNO}(\times, p_2, p_5)
\end{aligned}
$$



Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

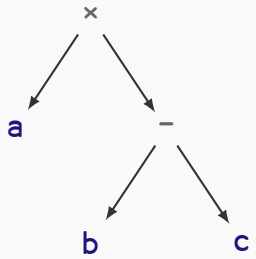
$$
\begin{aligned}
p_1 &:= \text{CRIARFOLHA}(\textbf{id}, p_a) \\
p_2 &:= \text{CRIARFOLHA}(\textbf{id}, p_a) \\
p_3 &:= \text{CRIARFOLHA}(\textbf{id}, p_b) \\
p_4 &:= \text{CRIARFOLHA}(\textbf{id}, p_c) \\
p_5 &:= \text{CRIARNO}(-, p_3, p_4) \\
p_6 &:= \text{CRIARNO}(\times, p_2, p_5)
\end{aligned}
$$



Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

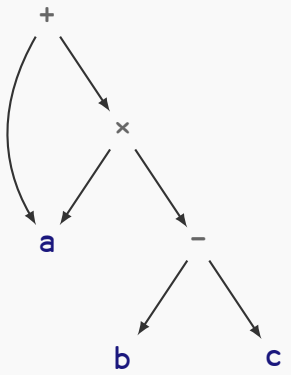
$$
\begin{aligned}
p_1 &:= \text{CRIARFOLHA}(\textbf{id}, p_a) \\
p_2 &:= \text{CRIARFOLHA}(\textbf{id}, p_a) \\
p_3 &:= \text{CRIARFOLHA}(\textbf{id}, p_b) \\
p_4 &:= \text{CRIARFOLHA}(\textbf{id}, p_c) \\
p_5 &:= \text{CRIARNO}(-, p_3, p_4) \\
p_6 &:= \text{CRIARNO}(\times, p_2, p_5) \\
p_7 &:= \text{CRIARNO}(+, p_1, p_6)
\end{aligned}
$$



Criação do DAG para a expressão `a + a × (b − c) + (b − c) × d`

Chamadas de funções

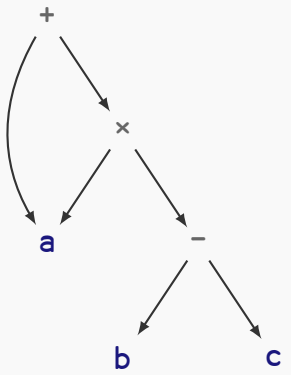
$$
\begin{aligned}
p_1 &:= \text{CRIARFOLHA}(\textbf{id}, p_a) \\
p_2 &:= \text{CRIARFOLHA}(\textbf{id}, p_a) \\
p_3 &:= \text{CRIARFOLHA}(\textbf{id}, p_b) \\
p_4 &:= \text{CRIARFOLHA}(\textbf{id}, p_c) \\
p_5 &:= \text{CRIARNO}(-, p_3, p_4) \\
p_6 &:= \text{CRIARNO}(\times, p_2, p_5) \\
p_7 &:= \text{CRIARNO}(+, p_1, p_6)
\end{aligned}
$$



Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

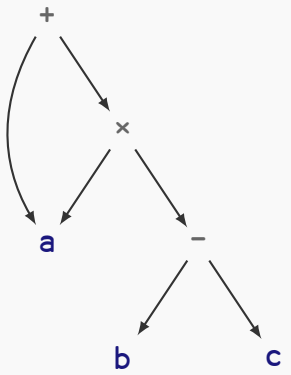
$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_3 := \text{CRIARFOLHA}(\textbf{id}, p_b)$
$p_4 := \text{CRIARFOLHA}(\textbf{id}, p_c)$
$p_5 := \text{CRIARNO}(-, p_3, p_4)$
$p_6 := \text{CRIARNO}(\times, p_2, p_5)$
$p_7 := \text{CRIARNO}(+, p_1, p_6)$
$p_8 := \text{CRIARFOLHA}(\textbf{id}, p_b)$



Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

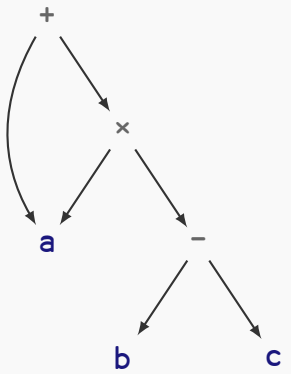
$$
\begin{aligned}
p_1 &:= \textsc{criarFolha}(\textbf{id}, p_a)\\
p_2 &:= \textsc{criarFolha}(\textbf{id}, p_a)\\
p_3 &:= \textsc{criarFolha}(\textbf{id}, p_b)\\
p_4 &:= \textsc{criarFolha}(\textbf{id}, p_c)\\
p_5 &:= \textsc{criarNo}(-, p_3, p_4)\\
p_6 &:= \textsc{criarNo}(\times, p_2, p_5)\\
p_7 &:= \textsc{criarNo}(+, p_1, p_6)\\
p_8 &:= \textsc{criarFolha}(\textbf{id}, p_b)\\
p_9 &:= \textsc{criarFolha}(\textbf{id}, p_c)
\end{aligned}
$$



Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

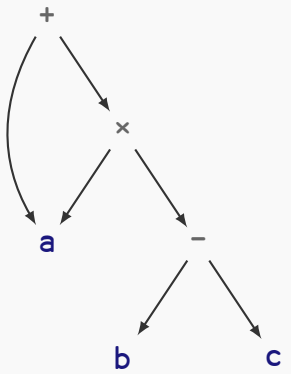
$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_3 := \text{CRIARFOLHA}(\textbf{id}, p_b)$
$p_4 := \text{CRIARFOLHA}(\textbf{id}, p_c)$
$p_5 := \text{CRIARNO}(-, p_3, p_4)$
$p_6 := \text{CRIARNO}(\times, p_2, p_5)$
$p_7 := \text{CRIARNO}(+, p_1, p_6)$
$p_8 := \text{CRIARFOLHA}(\textbf{id}, p_b)$
$p_9 := \text{CRIARFOLHA}(\textbf{id}, p_c)$
$p_{10} := \text{CRIARNO}(-, p_8, p_9)$



Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

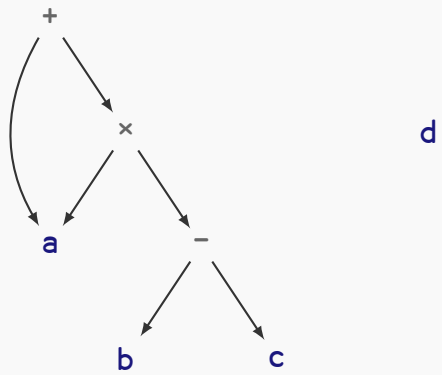
$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_3 := \text{CRIARFOLHA}(\textbf{id}, p_b)$
$p_4 := \text{CRIARFOLHA}(\textbf{id}, p_c)$
$p_5 := \text{CRIARNO}(-, p_3, p_4)$
$p_6 := \text{CRIARNO}(\times, p_2, p_5)$
$p_7 := \text{CRIARNO}(+, p_1, p_6)$
$p_8 := \text{CRIARFOLHA}(\textbf{id}, p_b)$
$p_9 := \text{CRIARFOLHA}(\textbf{id}, p_c)$
$p_{10} := \text{CRIARNO}(-, p_8, p_9)$
$p_{11} := \text{CRIARFOLHA}(\textbf{id}, p_d)$



Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

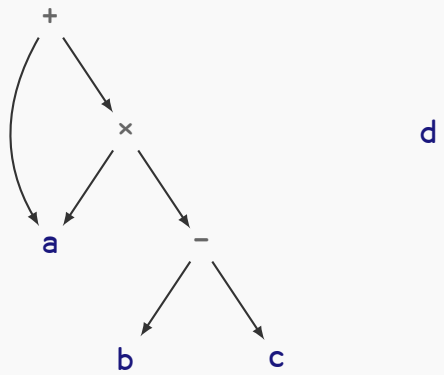
$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_3 := \text{CRIARFOLHA}(\textbf{id}, p_b)$
$p_4 := \text{CRIARFOLHA}(\textbf{id}, p_c)$
$p_5 := \text{CRIARNO}(-, p_3, p_4)$
$p_6 := \text{CRIARNO}(\times, p_2, p_5)$
$p_7 := \text{CRIARNO}(+, p_1, p_6)$
$p_8 := \text{CRIARFOLHA}(\textbf{id}, p_b)$
$p_9 := \text{CRIARFOLHA}(\textbf{id}, p_c)$
$p_{10} := \text{CRIARNO}(-, p_8, p_9)$
$p_{11} := \text{CRIARFOLHA}(\textbf{id}, p_d)$



Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

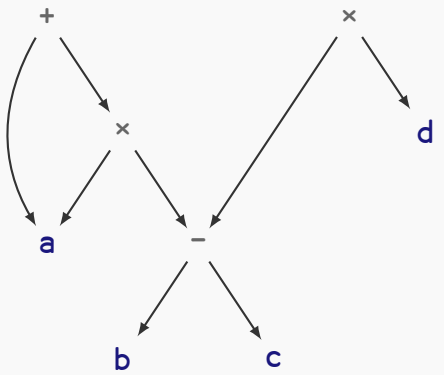
$$
\begin{aligned}
p_1 &:= \text{CRIARFOLHA}(\textbf{id}, p_a)\\
p_2 &:= \text{CRIARFOLHA}(\textbf{id}, p_a)\\
p_3 &:= \text{CRIARFOLHA}(\textbf{id}, p_b)\\
p_4 &:= \text{CRIARFOLHA}(\textbf{id}, p_c)\\
p_5 &:= \text{CRIARNO}(-, p_3, p_4)\\
p_6 &:= \text{CRIARNO}(\times, p_2, p_5)\\
p_7 &:= \text{CRIARNO}(+, p_1, p_6)\\
p_8 &:= \text{CRIARFOLHA}(\textbf{id}, p_b)\\
p_9 &:= \text{CRIARFOLHA}(\textbf{id}, p_c)\\
p_{10} &:= \text{CRIARNO}(-, p_8, p_9)\\
p_{11} &:= \text{CRIARFOLHA}(\textbf{id}, p_d)\\
p_{12} &:= \text{CRIARNO}(\times, p_{10}, p_{11})
\end{aligned}
$$



Criação do DAG para a expressão `a + a × (b - c) + (b - c) × d`

Chamadas de funções

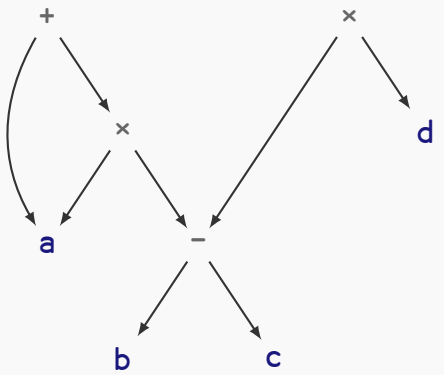
$$
\begin{aligned}
p_1 &:= \text{CRIARFOLHA}(\textbf{id}, p_a)\\
p_2 &:= \text{CRIARFOLHA}(\textbf{id}, p_a)\\
p_3 &:= \text{CRIARFOLHA}(\textbf{id}, p_b)\\
p_4 &:= \text{CRIARFOLHA}(\textbf{id}, p_c)\\
p_5 &:= \text{CRIARNO}(\texttt{-}, p_3, p_4)\\
p_6 &:= \text{CRIARNO}(\times, p_2, p_5)\\
p_7 &:= \text{CRIARNO}(\texttt{+}, p_1, p_6)\\
p_8 &:= \text{CRIARFOLHA}(\textbf{id}, p_b)\\
p_9 &:= \text{CRIARFOLHA}(\textbf{id}, p_c)\\
p_{10} &:= \text{CRIARNO}(\texttt{-}, p_8, p_9)\\
p_{11} &:= \text{CRIARFOLHA}(\textbf{id}, p_d)\\
p_{12} &:= \text{CRIARNO}(\times, p_{10}, p_{11})
\end{aligned}
$$



Criação do DAG para a expressão `a + a × (b − c) + (b − c) × d`

Chamadas de funções

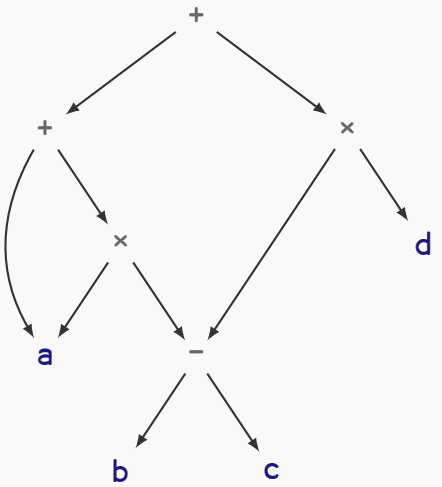
$$
\begin{aligned}
p_1 &:= \text{CRIARFOLHA}(\textbf{id}, p_a)\\
p_2 &:= \text{CRIARFOLHA}(\textbf{id}, p_a)\\
p_3 &:= \text{CRIARFOLHA}(\textbf{id}, p_b)\\
p_4 &:= \text{CRIARFOLHA}(\textbf{id}, p_c)\\
p_5 &:= \text{CRIARNO}(-, p_3, p_4)\\
p_6 &:= \text{CRIARNO}(\times, p_2, p_5)\\
p_7 &:= \text{CRIARNO}(+, p_1, p_6)\\
p_8 &:= \text{CRIARFOLHA}(\textbf{id}, p_b)\\
p_9 &:= \text{CRIARFOLHA}(\textbf{id}, p_c)\\
p_{10} &:= \text{CRIARNO}(-, p_8, p_9)\\
p_{11} &:= \text{CRIARFOLHA}(\textbf{id}, p_d)\\
p_{12} &:= \text{CRIARNO}(\times, p_{10}, p_{11})\\
p_{13} &:= \text{CRIARNO}(+, p_7, p_{12})
\end{aligned}
$$



Criação do DAG para a expressão a + a × (b − c) + (b − c) × d

Chamadas de funções

$p_1 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_2 := \text{CRIARFOLHA}(\textbf{id}, p_a)$
$p_3 := \text{CRIARFOLHA}(\textbf{id}, p_b)$
$p_4 := \text{CRIARFOLHA}(\textbf{id}, p_c)$
$p_5 := \text{CRIARNO}(-, p_3, p_4)$
$p_6 := \text{CRIARNO}(\times, p_2, p_5)$
$p_7 := \text{CRIARNO}(+, p_1, p_6)$
$p_8 := \text{CRIARFOLHA}(\textbf{id}, p_b)$
$p_9 := \text{CRIARFOLHA}(\textbf{id}, p_c)$
$p_{10} := \text{CRIARNO}(-, p_8, p_9)$
$p_{11} := \text{CRIARFOLHA}(\textbf{id}, p_d)$
$p_{12} := \text{CRIARNO}(\times, p_{10}, p_{11})$
$p_{13} := \text{CRIARNO}(+, p_7, p_{12})$



Referências

1. **AHO**, Alfred V, **SETHI**, Ravi, **ULLMAN**, Jeffrey D. *Compiladores: Princípios, Técnicas e Ferramentas*, LTC Editora, 1995.