Nem gramática, nem autômato – diagramas sintáticos

Professor Eraldo Pereira Marinho

Compiladores 2º semestre 2020

UNESP/IGCE Rio Claro

Forma normal de Backus-Naur estendida

A forma normal de Backus-Naur estendida, em inglês EBNF, é uma forma necessária para evitar que notação gramatical seja misturada com a notação de expressões regulares, o que causa inconsistência quando se representa linguagens regulares.

Comecemos pelo fecho de Kleene de uma forma sentencial α. Para tanto, consideremos $A \rightarrow A\alpha|\beta$. Como sabemos, essa não é uma boa forma para implementação de analisadores sintáticos descendentes. Contudo, por indução, temos $A \Rightarrow^{n+1} \beta\alpha^n$, $\forall\, n \in \mathbb{N}$. Sabemos que α^n é um caso particular de α^*, para qualquer $n \in \mathbb{N}$ – informalmente, teríamos $A \Rightarrow^* \beta\alpha^*$.

Introduzimos a notação $\{\alpha\}$ para denotar o significado de α^*. Assim, $A \rightarrow A\alpha \mid \beta$ é equivalente a $A \rightarrow \beta\{\alpha\}$.

EBNF

Observemos que a forma equivalente a $A \rightarrow A\alpha \mid \beta$, sem recursão esquerda, é $A \rightarrow \beta R, R \rightarrow \alpha R \mid \varepsilon$. É imediato que $R \rightarrow \alpha R \mid \varepsilon$ é equivalente a $R \rightarrow \{\alpha\}$, na EBNF. Assim, $A \rightarrow \beta R, R \rightarrow \alpha R \mid \varepsilon$ é, também, equivalente a $A \rightarrow \beta\{\alpha\}$, uma vez que $A \rightarrow \beta R, R \rightarrow \{\alpha\}$ teria uma produção adicional desnecessária.

Tomemos o caso $A \rightarrow \alpha A \mid \beta$, com recursão direita. É fácil mostrar que $A \Rightarrow^{n+1} \alpha^n \beta$. Neste caso, $A \rightarrow \{\alpha\}\beta$.

EBNF

Agora, consideremos a produção $A \rightarrow \alpha \mid \varepsilon$. Isso quer dizer que a forma sentencial que deriva de α é opcional, uma vez que A, em $\beta A \gamma$, pode ser substituído por $\beta \alpha \gamma$, mas pode ser substituído por $\beta \gamma$. Assim, a condição opcional de α é denotada por $[\alpha]$, ou seja, $[\alpha] \equiv \alpha \mid \varepsilon$. Portanto, $A \rightarrow \alpha \mid \varepsilon$ é equivalente à EBNF $A \rightarrow [\alpha]$. Neste caso, teríamos $\beta A \gamma \Rightarrow \beta [\alpha] \gamma$, deixando claro que α é opcional em $\beta \alpha \gamma$.

EBNF

Sumariamente:

1. $A \to A\alpha \mid \beta$ ou $A \to \beta R,\ R \to \alpha R \mid \varepsilon$ é o mesmo que $A \to \beta\{\alpha\}$
2. $A \to \alpha A \mid \beta$ é o mesmo que $A \to \{\alpha\}\beta$
3. $A \to \alpha \mid \varepsilon$ equivale a $A \to [\alpha]$

No caso de $A \to A\alpha_1 \mid \cdots \mid A\alpha_m \mid \beta_1 \mid \cdots \beta_n$, generalizamos para

$A \to \beta_l\{\alpha_1 \mid \cdots \mid \alpha_m\}$, com $l = 1, \ldots, n$

Como ficaria $A \to \alpha_1 A \mid \cdots \mid \alpha_m A \mid \beta_1 \mid \cdots \beta_n$?

Diagrama sintático

Adotamos a notação Pascal (Wirth 1972) para diagrama sintático.

Para mais bem ilustrar, tomemos, novamente, a gramática simplificada de expressões, na forma LL(1), mas agora incluindo o operador unário de negação $\ominus$:

$$E \rightarrow \ominus TR \mid TR$$

$$R \rightarrow \oplus TR \mid \varepsilon$$

$$T \rightarrow FQ$$

$$Q \rightarrow \otimes FQ \mid \varepsilon$$

$$F \rightarrow (E) \mid \boldsymbol{v} \mid \boldsymbol{c}$$

Diagrama sintático

Passando para a EBNF, a gramática anterior fica reduzida a

$$E \rightarrow [\ominus]T\{\oplus\ T\}$$

$$T \rightarrow F\{\otimes\ F\}$$

$$F \rightarrow (E) \mid \boldsymbol{v} \mid \boldsymbol{c}$$

Essas produções podem ser denotadas conforme os seguintes diagramas sintáticos (próximo slide)

Diagrama sintático

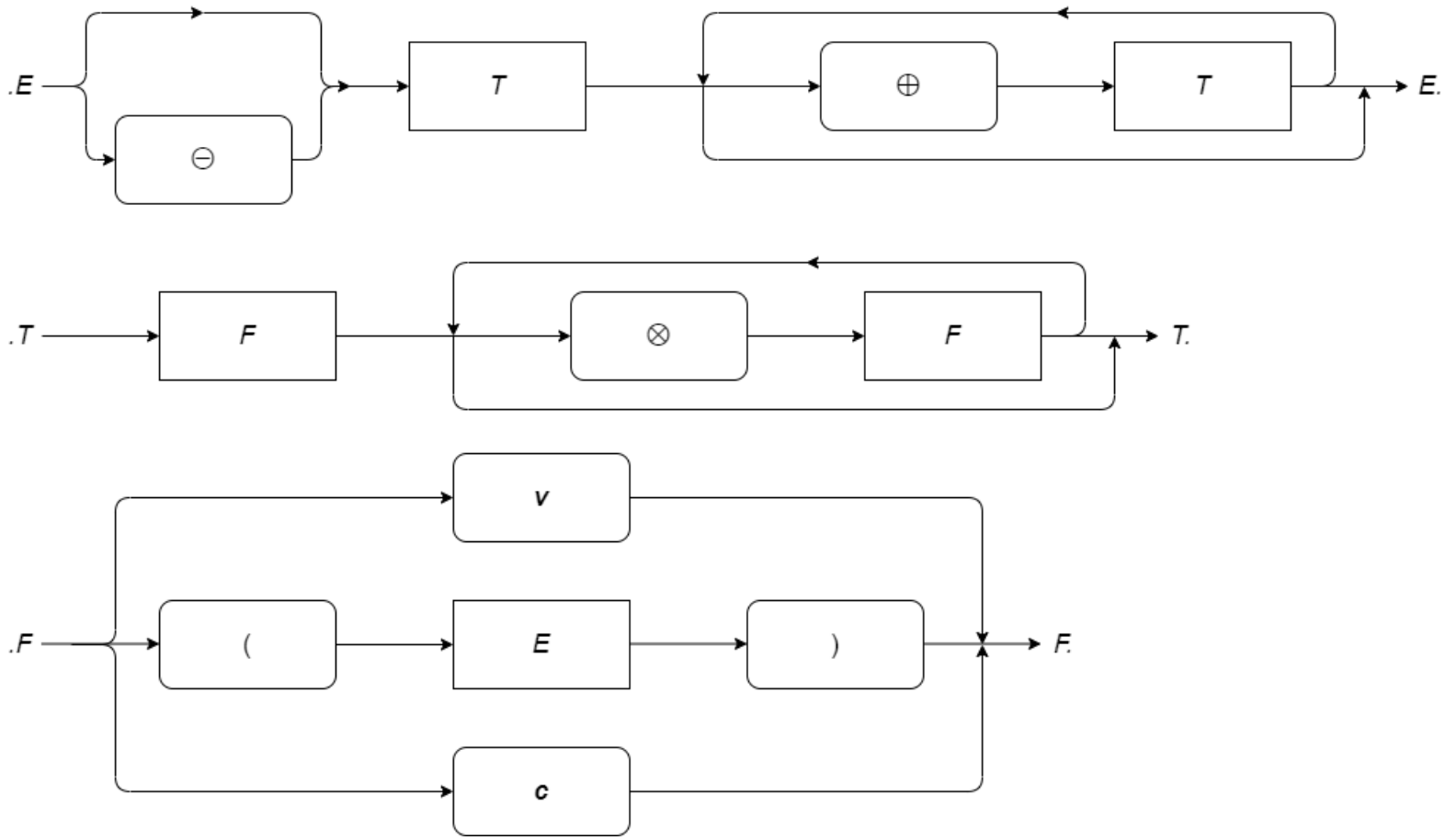
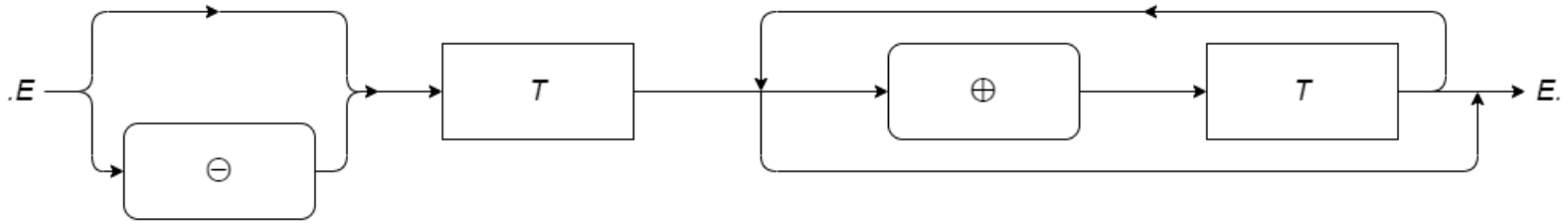
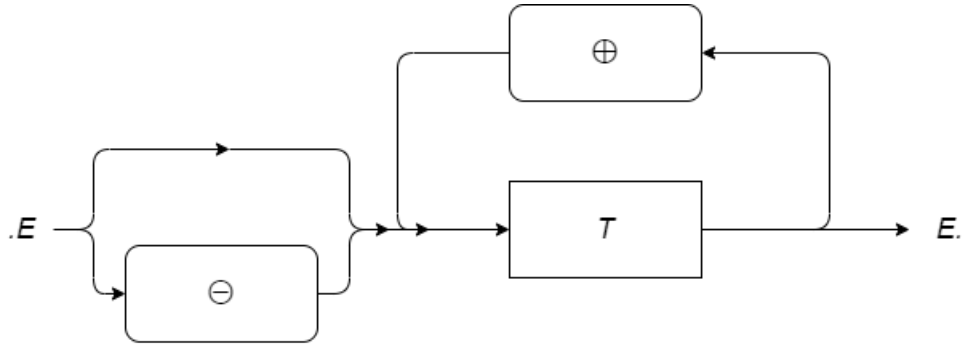


Diagrama sintático

Vamos dar um zoom na produção $E \rightarrow \ominus T\{\oplus T\}$:



Observemos que esse diagrama é topologicamente idêntico a



Deste modo, T é invocado pelo menos uma vez como no diagrama original. O loopback só ocorre se $lookahead$ avistar o operador $\oplus$. Caso contrário, o fluxo deixa o diagrama.

Diagrama sintático

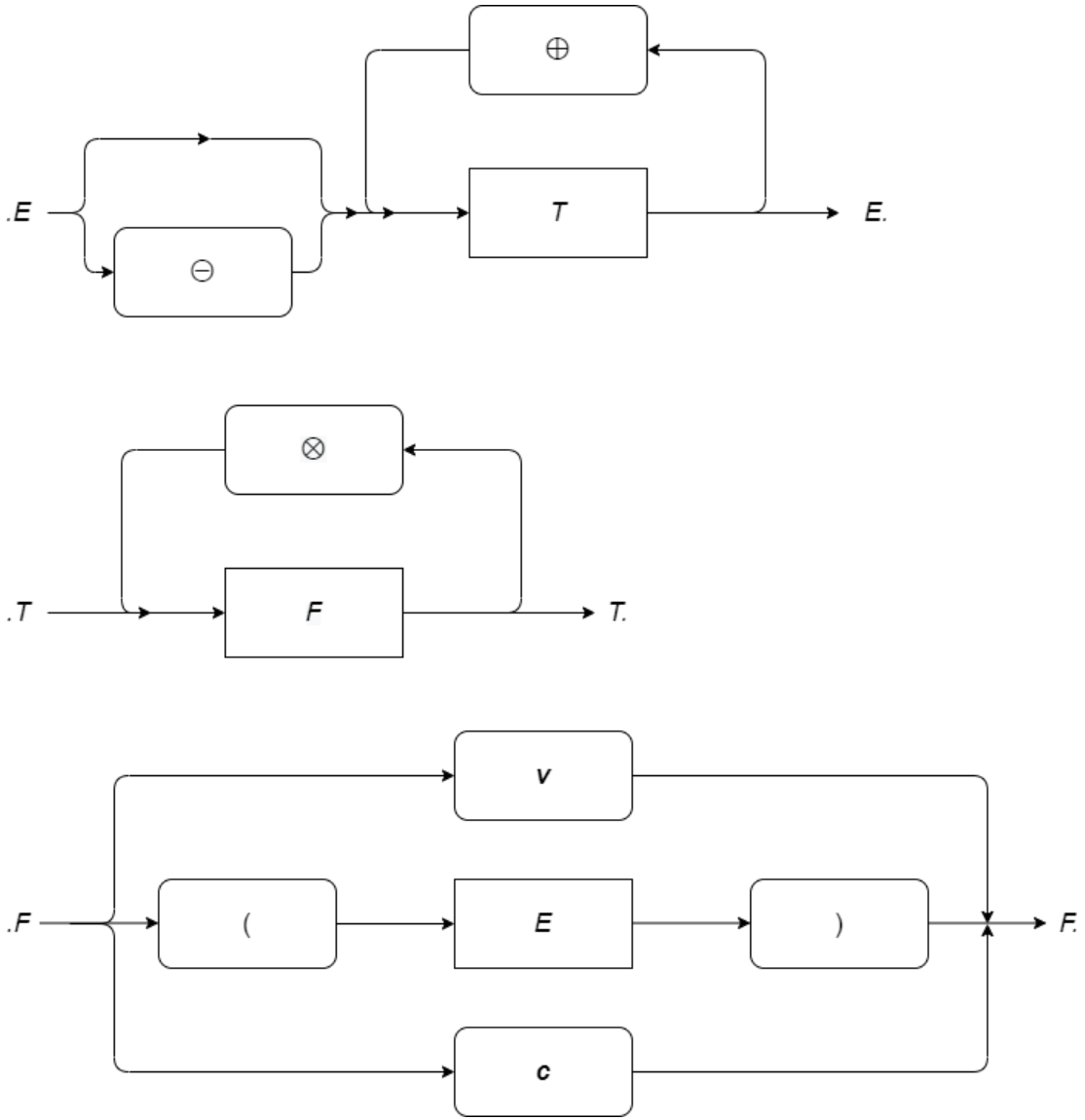
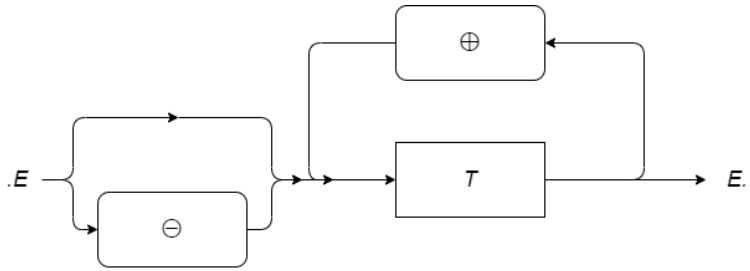


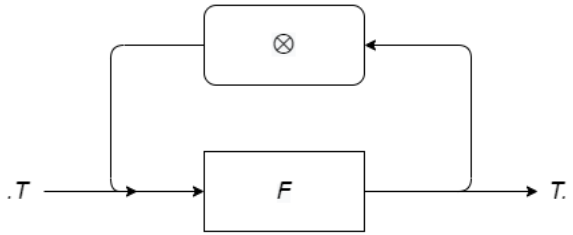
Diagrama sintático – aplicação

Aproveitando essa última representação de um diagrama sintático de expressões, podemos escrever os seguintes módulos:

```
void E(void) {
          if (lookahead == ⊖) {
                    match(⊖);
          }
_T:
          T();
          if (lookahead == ⊕) {
                    match(⊕);
                    goto _T;
          }
}
```




```
void T(void) {
_F:
          F();
          if (lookahead == ⊗) {
                    match(⊗);
                    goto _F;
          }
}
```




```
void F(void) {
          switch (lookahead) {
          case '(': match('('); E(); match(')'); break;
          case c: match(c); break;
          default: match(v);
          }
}
```

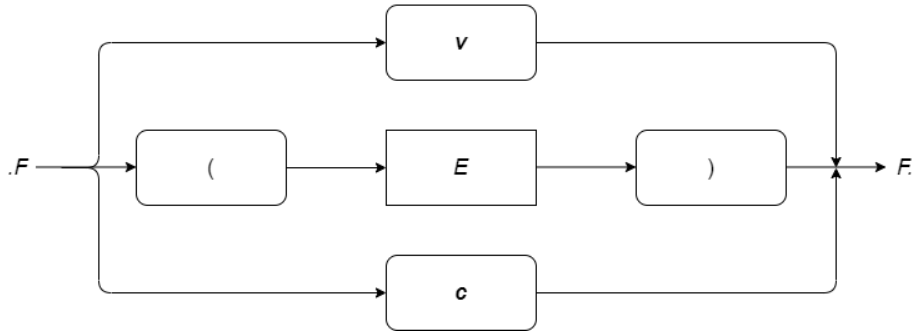
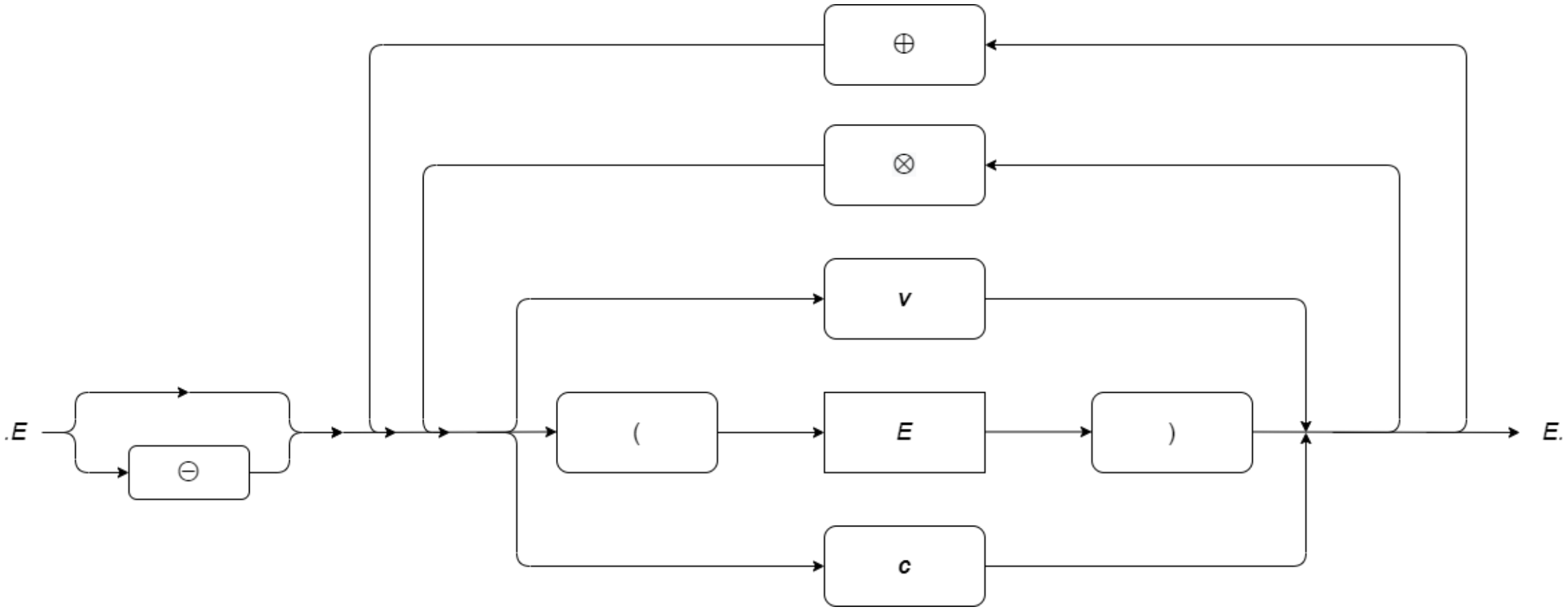


Diagrama sintático – pondo tudo junto



Questões, valendo 2 horas

1. O que é um diagrama sintático e qual sua vantagem no projeto de um analisador sintático?
2. Diagrama sintático pode ser utilizado no desenvolvimento de um analisador léxico?
3. De que modo a expressão regular a^* | b seria representada num diagrama sintático?
4. Podemos afirmar que gramáticas EBNF são facilmente transliteradas para diagramas sintáticos? Justifique.