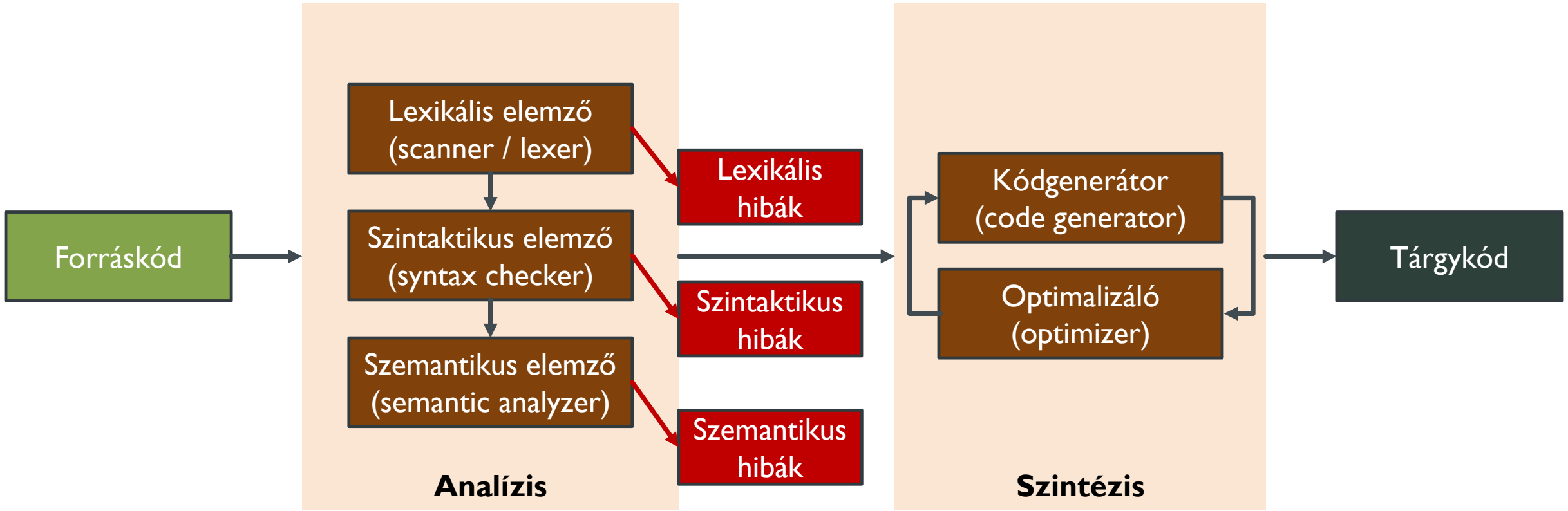
SZINTAKTIKUS ELEMZÉS

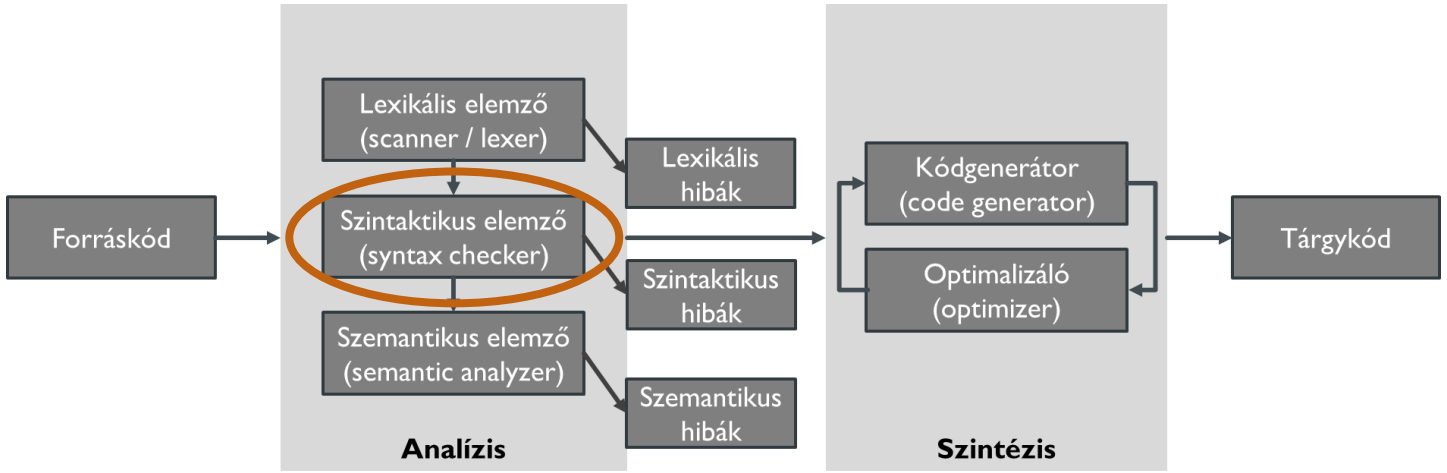
FORMÁLIS NYELVEK ÉS FORDÍTÓPROGRAMOK ALAPJAI

Dévai Gergely
ELTE

A FORDÍTÓPROGRAMOK LOGIKAI FELÉPÍTÉSE



SZINTAKTIKUS ELEMZŐ



- *Feladat:*
  A forrásszöveg szerkezetének felderítése, formai ellenőrzése
- *Bemenet:*
  Lexikális elemek (tokenek) sorozata
- *Kimenet:*
  Szintaxisfa + szintaktikus hibák
- *Eszközök:*
  Környezetfüggetlen nyelvtanok, veremautomaták

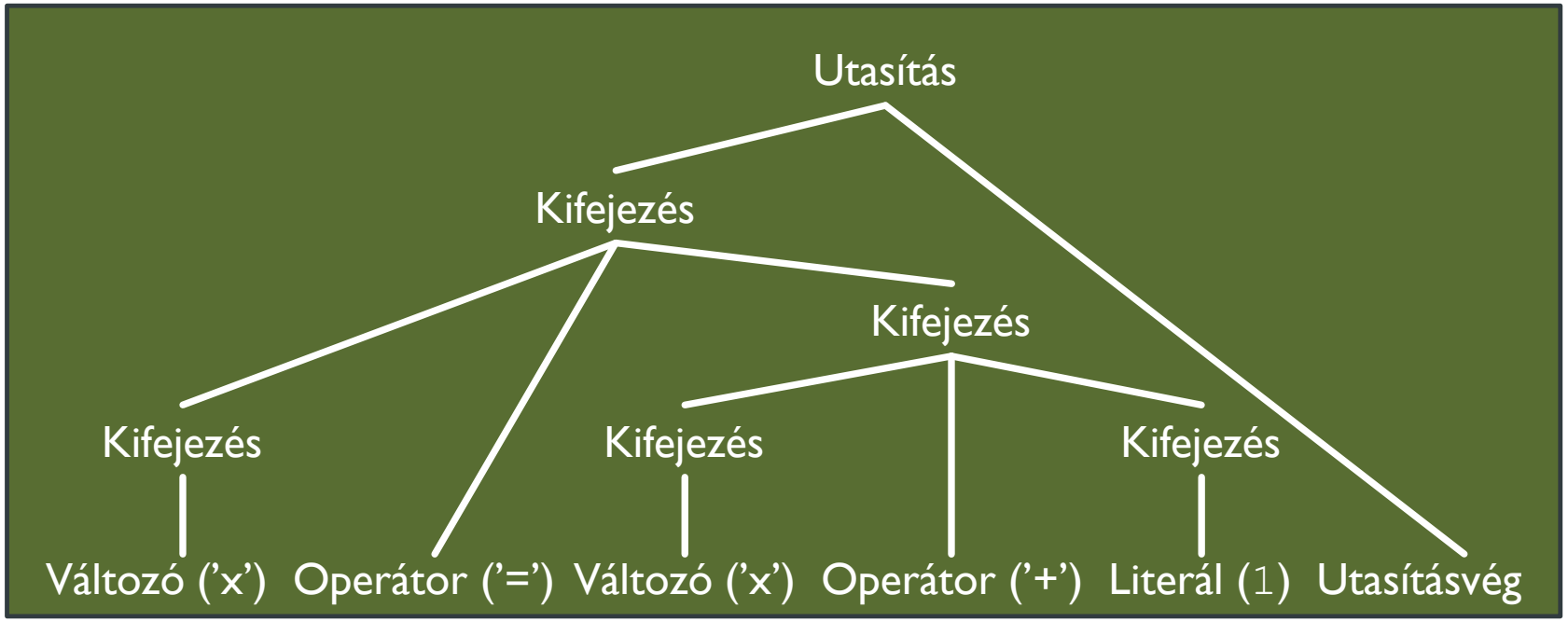
SZINTAKTIKUS ELEMZŐ - PÉLDA

Utasítás →
Kifejezés Utasításvég

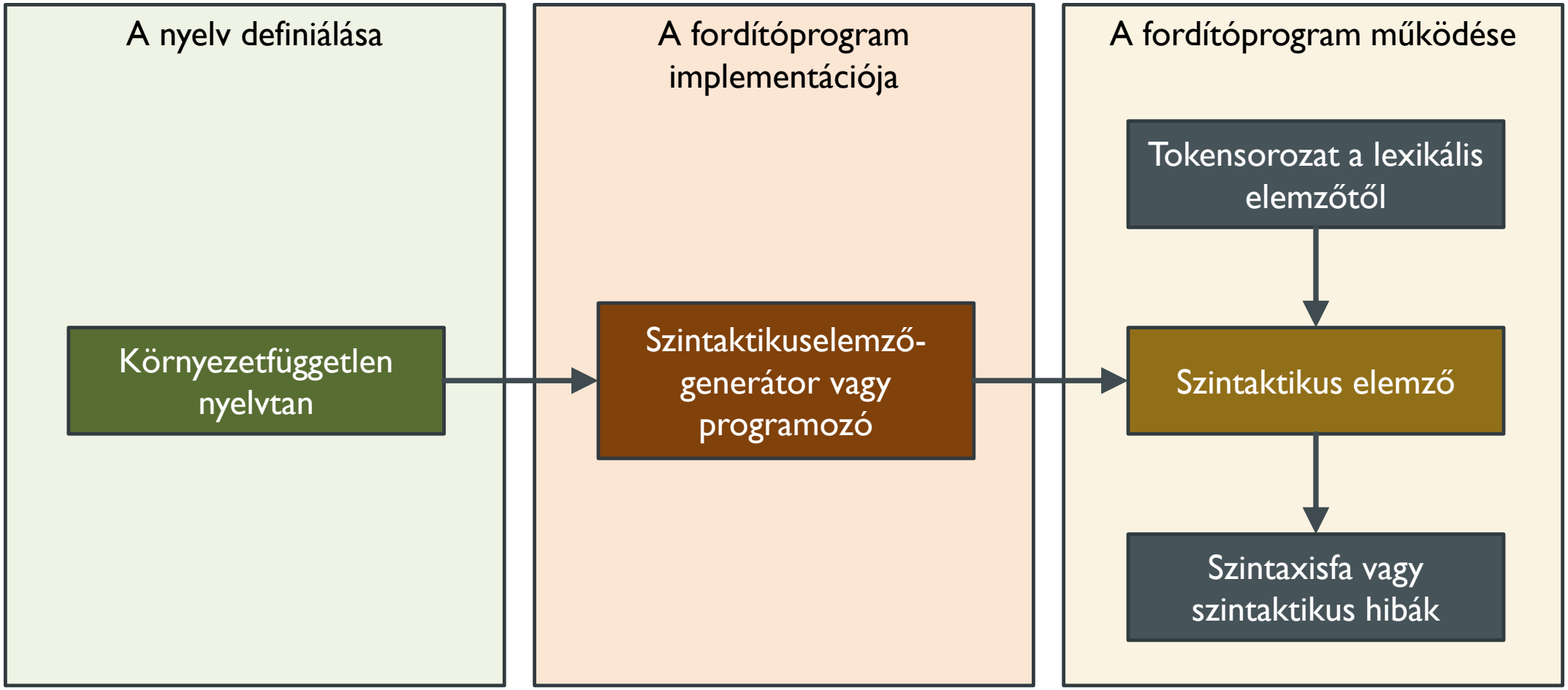
Kifejezés →
Változó | Literál |
Kifejezés Operátor Kifejezés

! A lexikális elemek (tokenek) a szintaktikus elemzés nyelvtanának terminális szimbólumai.

Változó (’x’), Operátor (’=’), Változó (’x’), Operátor (’+’), Literál (1), Utasításvég



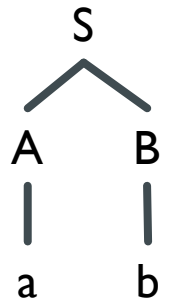
SZINTAKTIKUS ELEMZŐ LÉTREHOZÁSA



MIT VÁRUNK EL A NYELVTANTÓL?

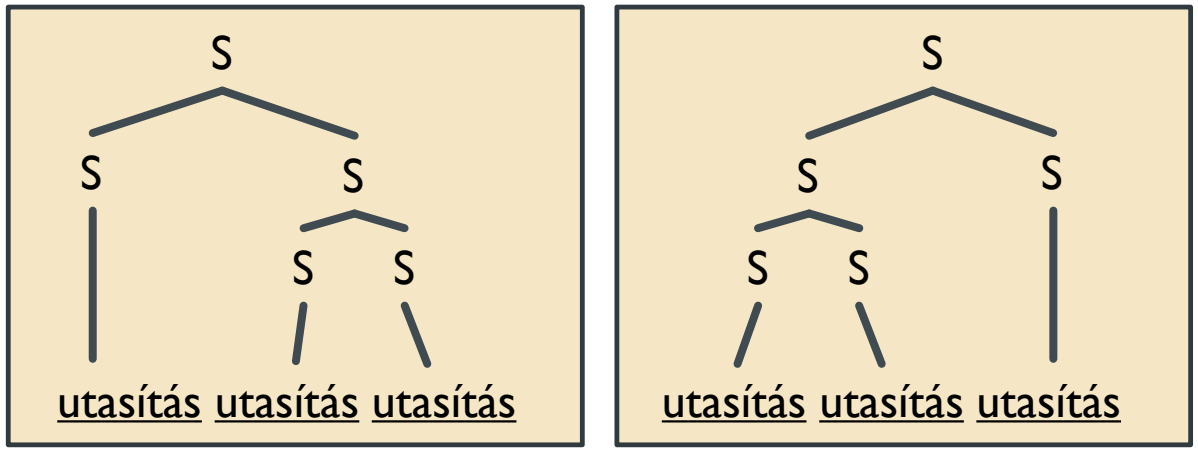
- **Redukáltság**: Nincsenek „felesleges” nemterminálisok
  - Mindegyik nemterminálishoz adható olyan levezetés, amiben szerepel, és nem üres terminális sorozatot vezetünk le belőle.
- **Ciklusmentesség**: Nincs $A \rightarrow^{+} A$ levezetés
  - Példa ciklusos (tehát nem jó) nyelvtanra:
    $S \rightarrow A$
    $A \rightarrow a \mid B$
    $B \rightarrow A$
- **Egyértelműség**: Minden szóhoz pontosan egy szintaxisfa tartozik.
  - Több *levezetés* lehet, de csak egy *szintaxisfa*!

$S \Rightarrow AB \Rightarrow aB \Rightarrow ab$
$S \Rightarrow AB \Rightarrow Ab \Rightarrow ab$



PÉLDA NEM EGYÉRTELMŰ NYELVTANRA

S → utasítás | SS

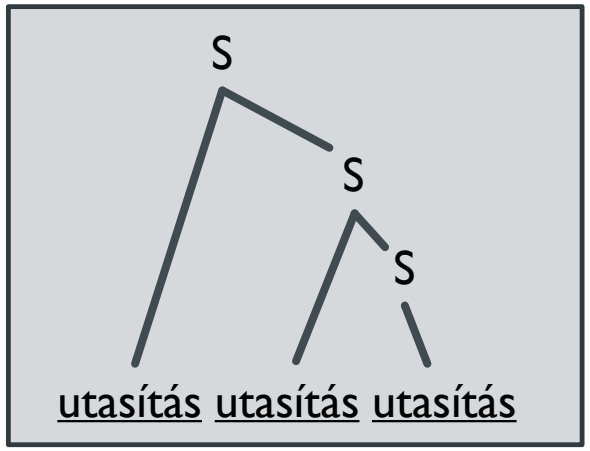


Nem egyértelmű

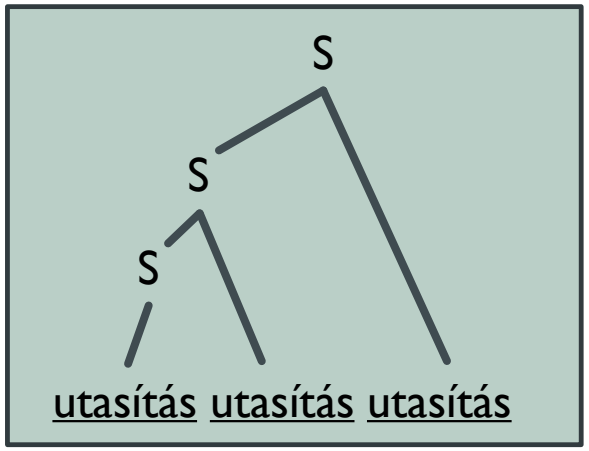
Ez a nemegyértelműség feloldható a nyelvtan átalakításával:

S → utasítás S | utasítás

S → S utasítás | utasítás



Egyértelmű

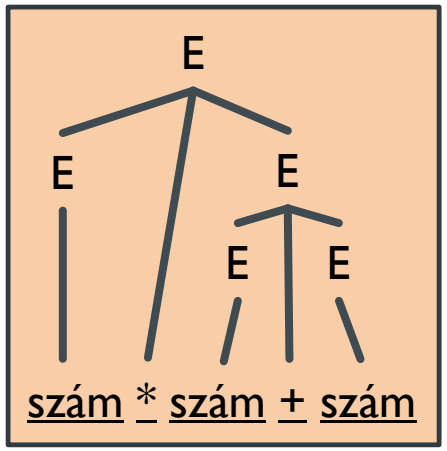


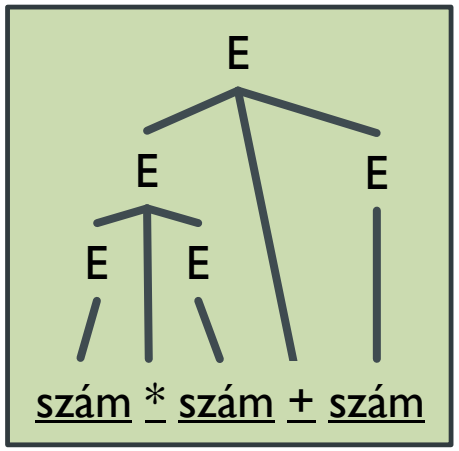
Egyértelmű

MÁSIK PÉLDA NEM EGYÉRTELMŰ NYELVTANRA

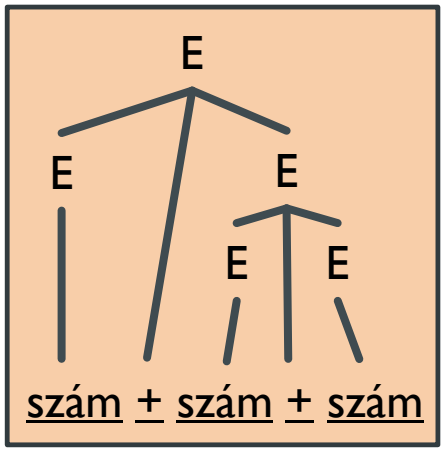
E → szám | E + E | E * E

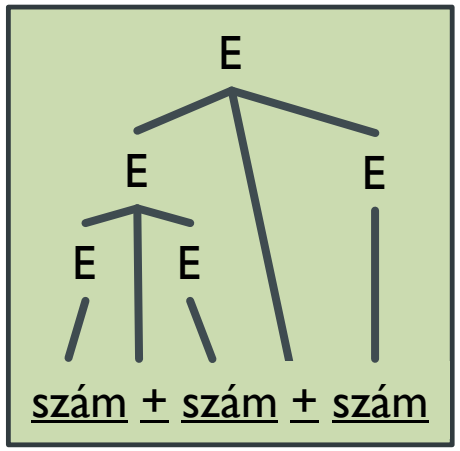
Nem egyértelmű





Nem egyértelmű





Ez a nemegyértelműség feloldható:

- Az operátorok precedenciájának és asszociativitásának megadásával
- A nyelvtan átalakításával

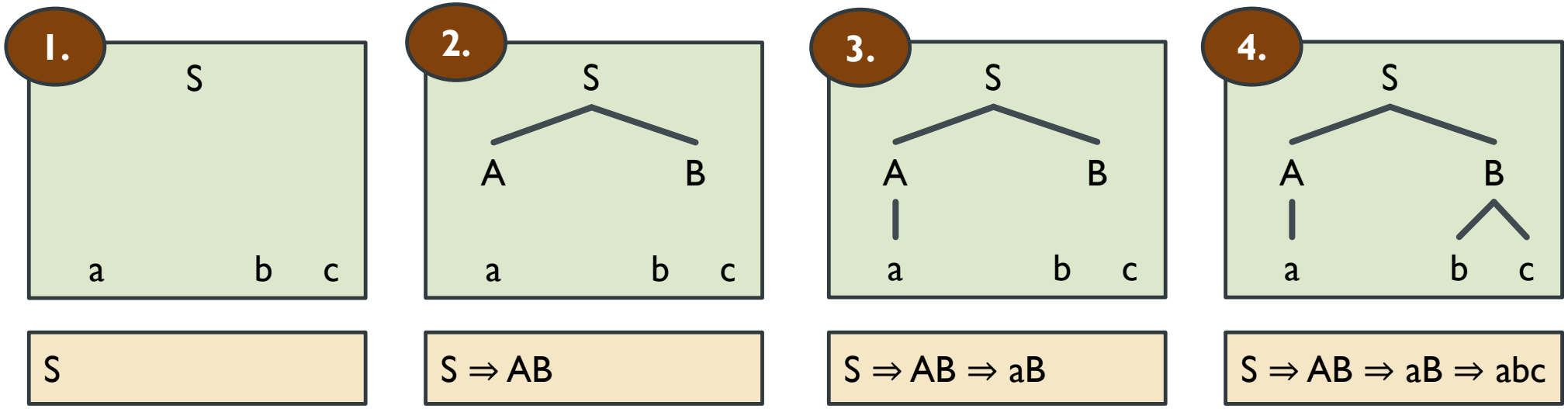
- A * magasabb precedenciájú, mint a **+**.
- Mindkettő balasszociatív.

E → F | E + F
F → szám | F * szám

A SZINTAKTIKUS ELEMZÉS STRATÉGIÁI: **FELÜLRŐL LEFELÉ ELEMZÉS**

Nyelvtan:
S → AB
A → a
B → bc

Input: abc



S

S ⇒ AB

S ⇒ AB ⇒ aB

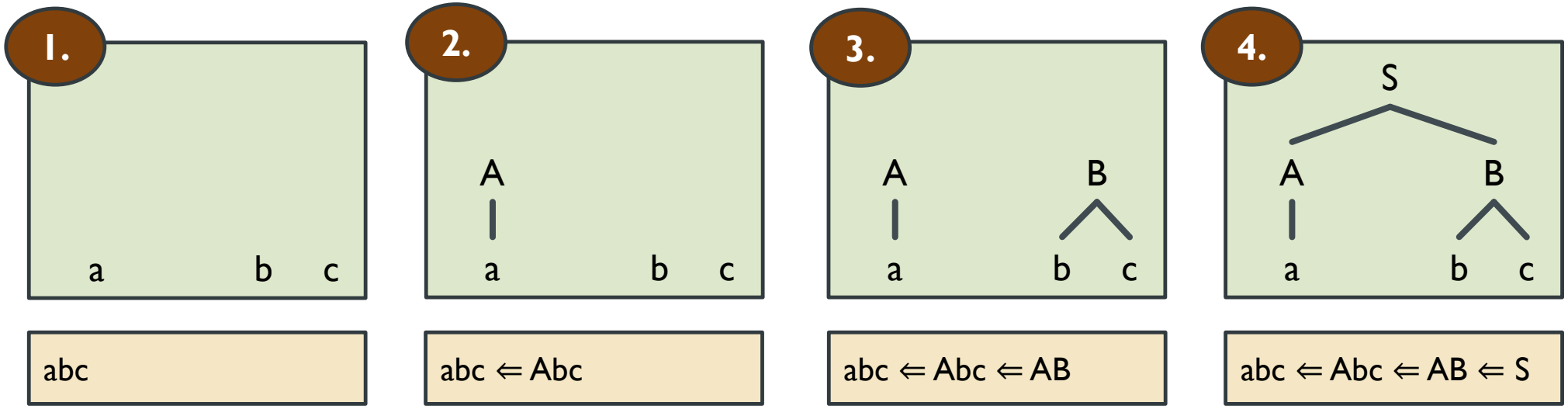
S ⇒ AB ⇒ aB ⇒ abc

- A startszimbólumból indulva a terminálisok felé építjük a fát
- Az input feldolgozása balról jobbra történik
- Legbaloldalibb levezetést állít elő
  - A *legbaloldalibb levezetés* több terminális esetén a legbaloldalibbat helyettesíti: S ⇒ AB ⇒ aB ⇒ abc

A SZINTAKTIKUS ELEMZÉS STRATÉGIÁI: **ALULRÓL FELFELÉ ELEMZÉS**

Nyelvtan:
S → AB
A → a
B → bc

Input: abc



- A terminálisoktól indulva a startszimbólum felé építjük a fát
- Az input feldolgozása balról jobbra történik itt is
- Legjobboldalibb levezetés inverzét állítja elő
  - A *legjobboldalibb levezetés* több terminális esetén a legjobboldalibbat helyettesíti: S ⇒ AB ⇒ Abc ⇒ abc

FELÜLRŐL LEFELÉ ELEMZÉS

LL ELEMZÉSEK

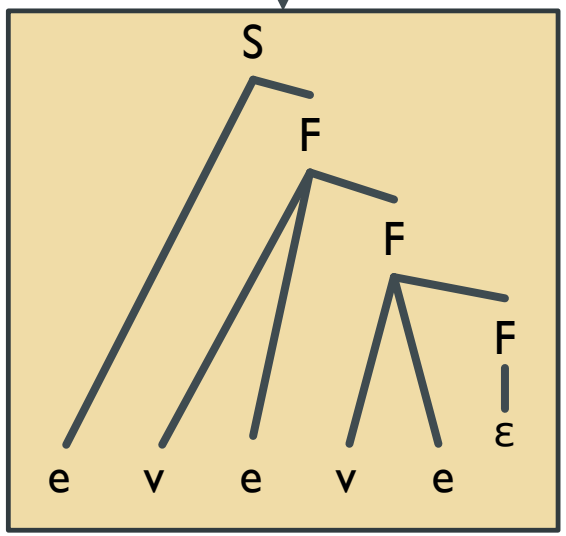
PÉLDA NYELVTAN:VESSZŐVEL ELVÁLASZTOTT LISTA

apple, banana, pear

Lexikális elemző:
[a-z]+ → e (elem)
“,” → v (vessző)
[\t\n] →

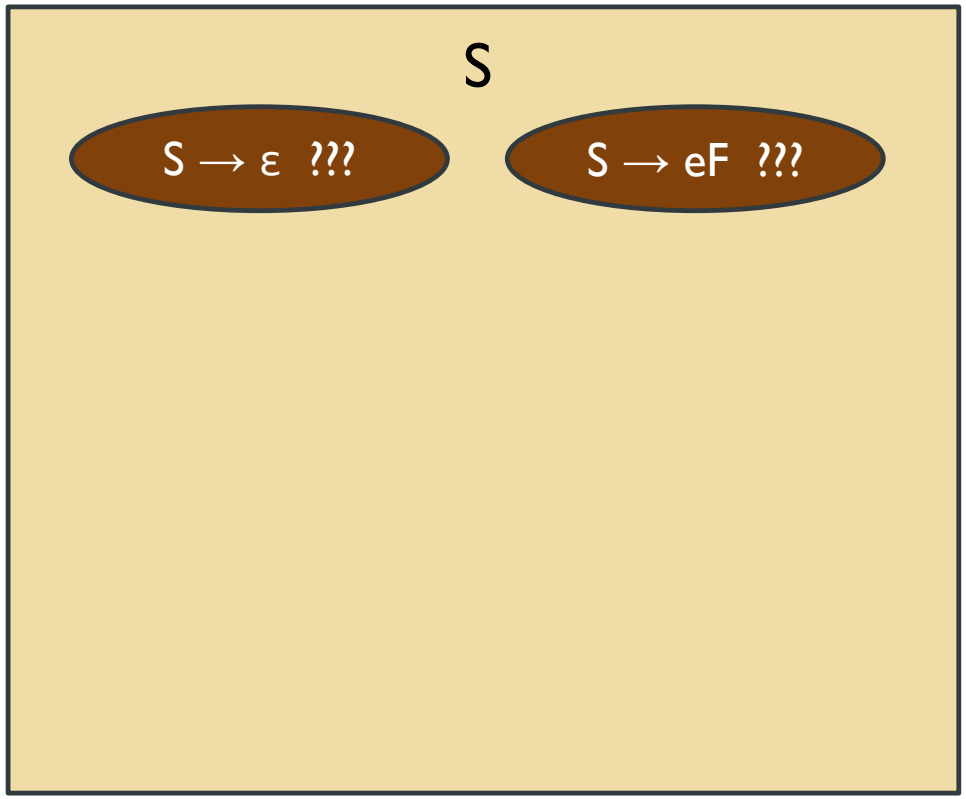
eveve

Szintaktikus elemző:
S → ε | eF
F → ε | veF



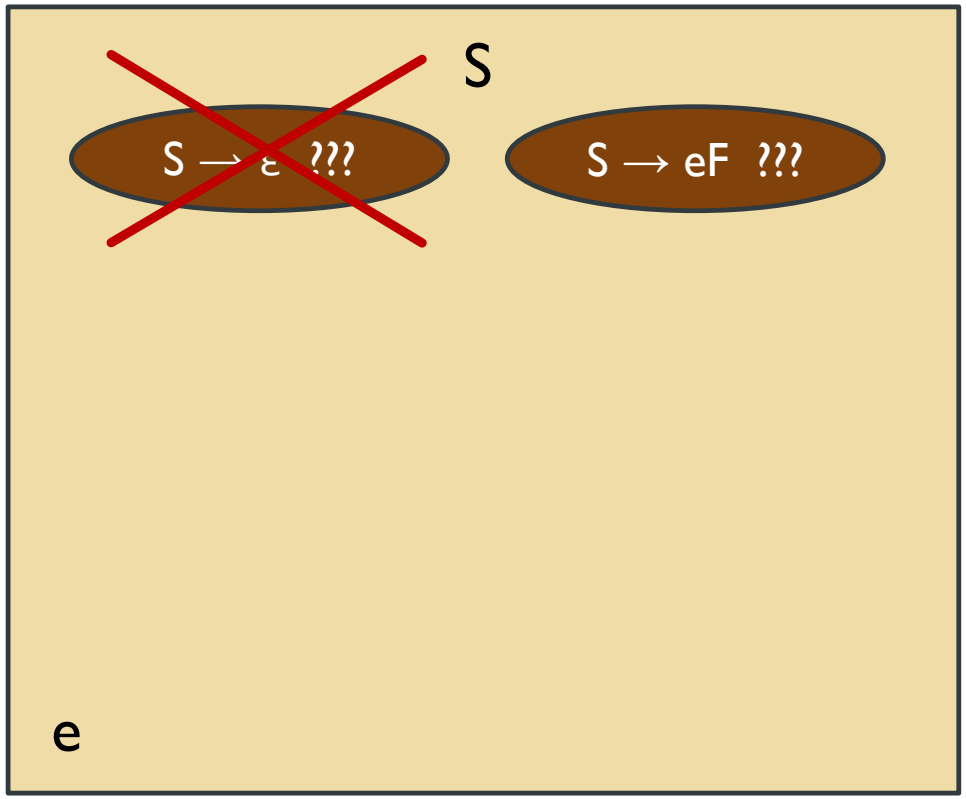
MELYIK SZABÁLYT KELL ALKALMAZNI?

$S \rightarrow \varepsilon \mid eF$
$F \rightarrow \varepsilon \mid veF$



MELYIK SZABÁLYT KELL ALKALMAZNI?

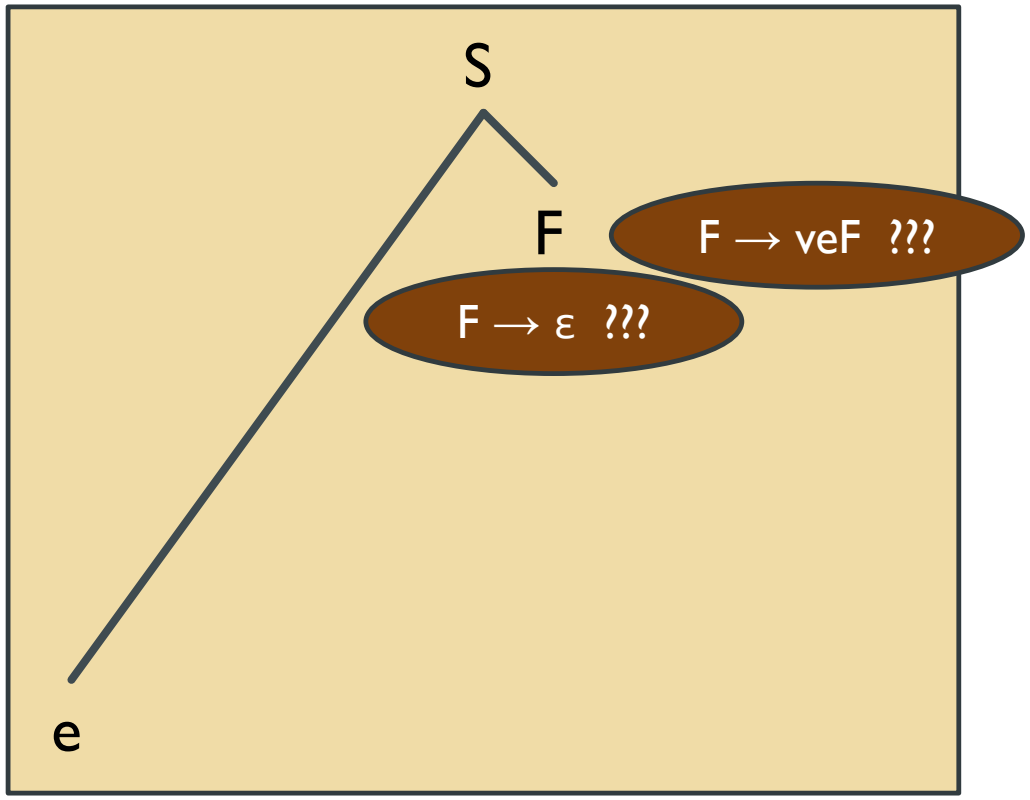
$S \rightarrow \varepsilon \mid eF$
$F \rightarrow \varepsilon \mid veF$



Előreolvasás:
Nézzük meg a következő néhány tokent, és válasszunk szabályt azok alapján!

MELYIK SZABÁLYT KELL ALKALMAZNI?

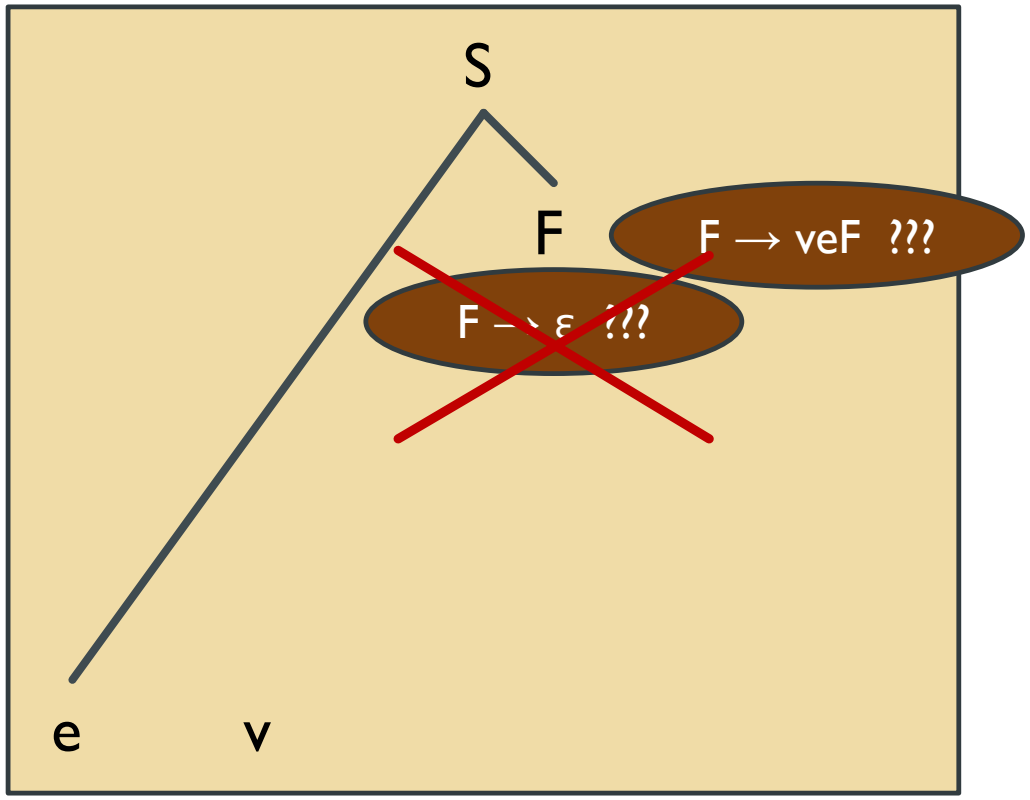
$S \rightarrow \varepsilon \mid eF$
$F \rightarrow \varepsilon \mid veF$



Előreolvasás:
Nézzük meg a következő néhány tokent, és válasszunk szabályt azok alapján!

MELYIK SZABÁLYT KELL ALKALMAZNI?

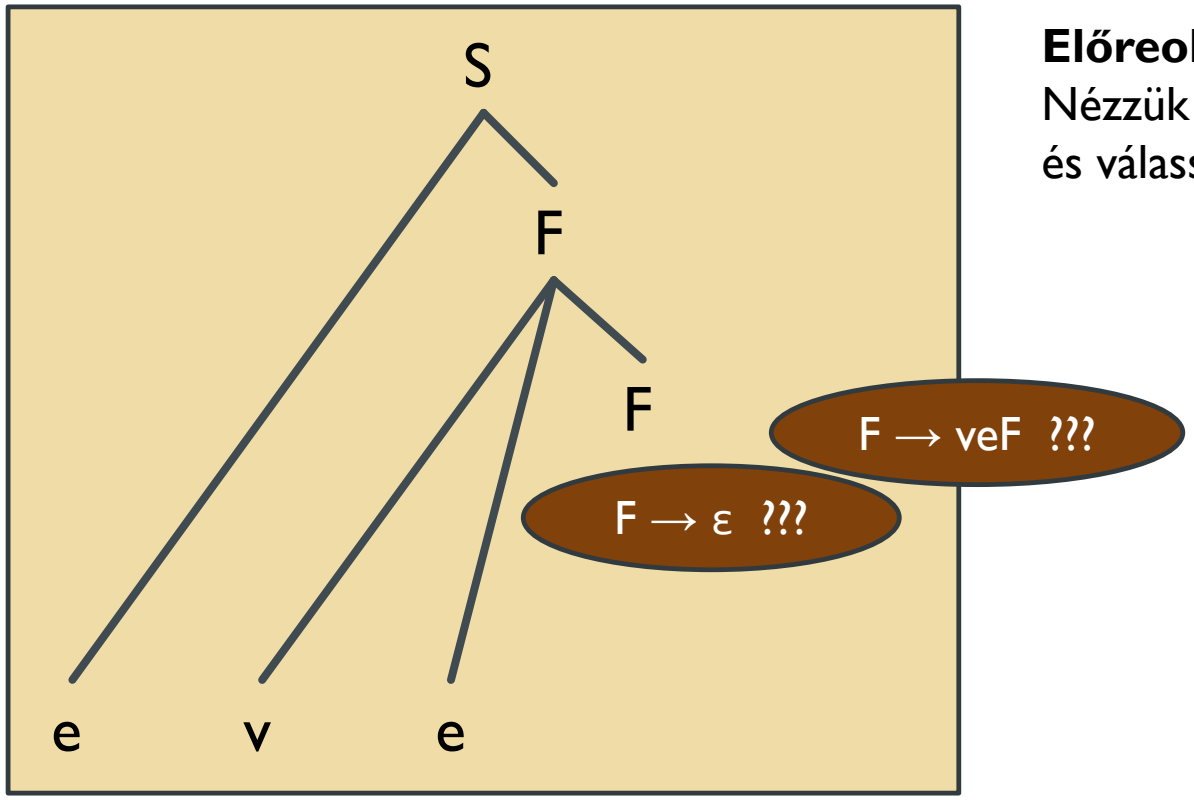
$S \rightarrow \varepsilon \mid eF$
$F \rightarrow \varepsilon \mid veF$



Előreolvasás:
Nézzük meg a következő néhány tokent, és válasszunk szabályt azok alapján!

MELYIK SZABÁLYT KELL ALKALMAZNI?

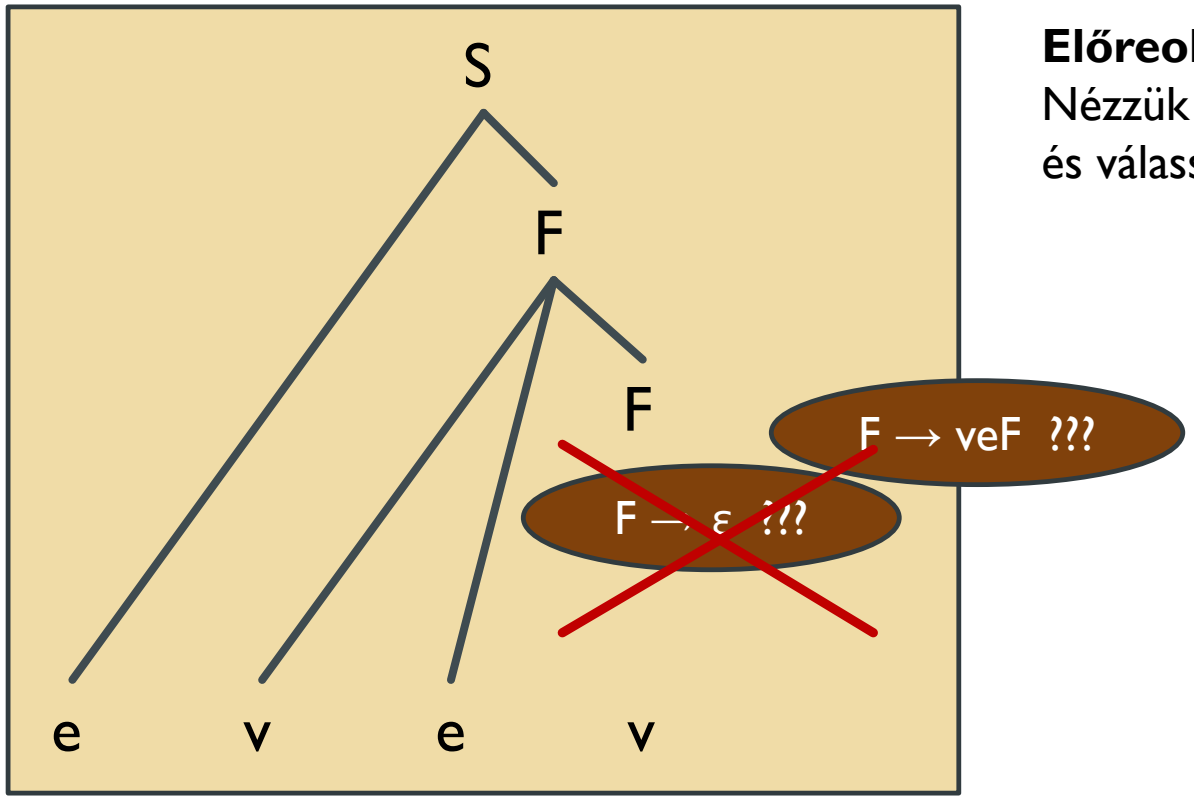
$S \rightarrow \varepsilon \mid eF$
$F \rightarrow \varepsilon \mid veF$



Előreolvasás:
Nézzük meg a következő néhány tokent, és válasszunk szabályt azok alapján!

MELYIK SZABÁLYT KELL ALKALMAZNI?

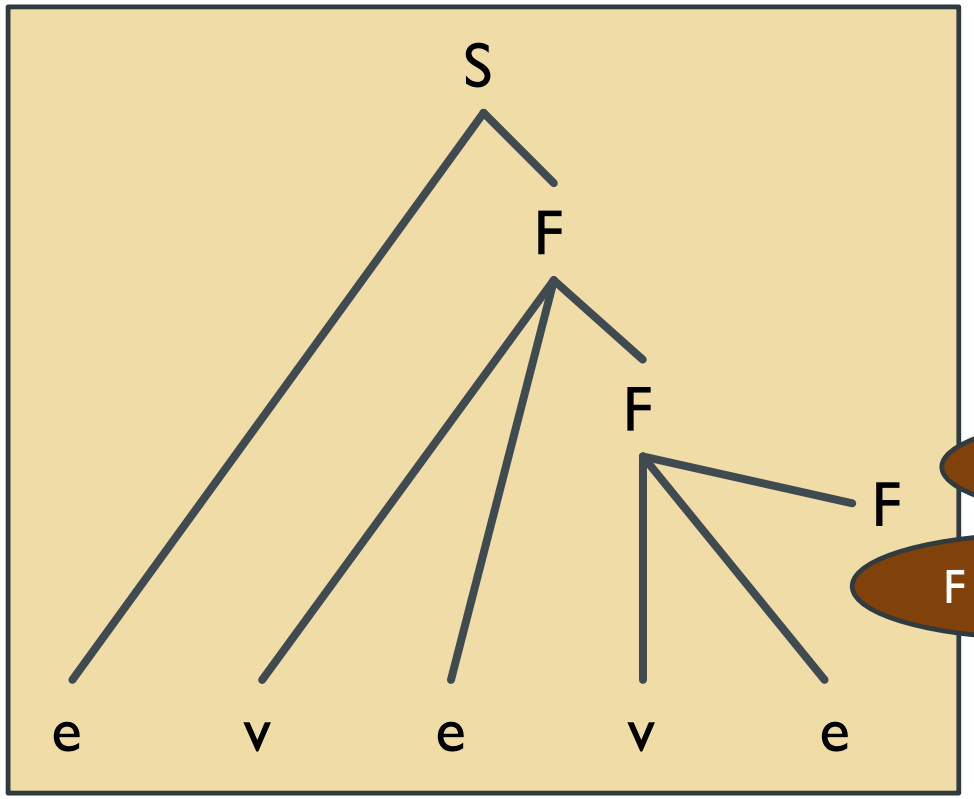
$S \rightarrow \varepsilon \mid eF$
$F \rightarrow \varepsilon \mid veF$



Előreolvasás:
Nézzük meg a következő néhány tokent, és válasszunk szabályt azok alapján!

MELYIK SZABÁLYT KELL ALKALMAZNI?

$S \rightarrow \varepsilon \mid eF$
$F \rightarrow \varepsilon \mid veF$



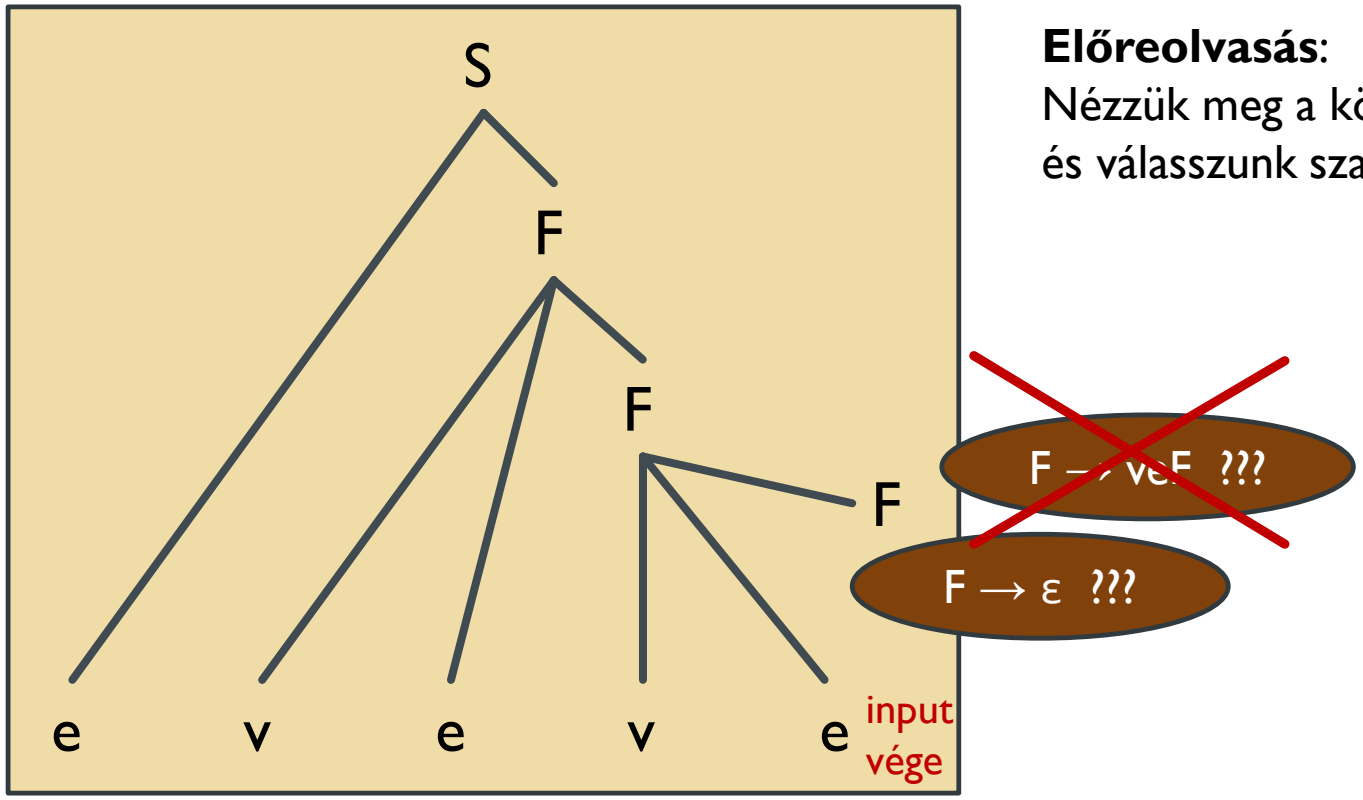
$F \rightarrow veF$???

$F \rightarrow \varepsilon$???

Előreolvasás:
Nézzük meg a következő néhány tokent, és válasszunk szabályt azok alapján!

MELYIK SZABÁLYT KELL ALKALMAZNI?

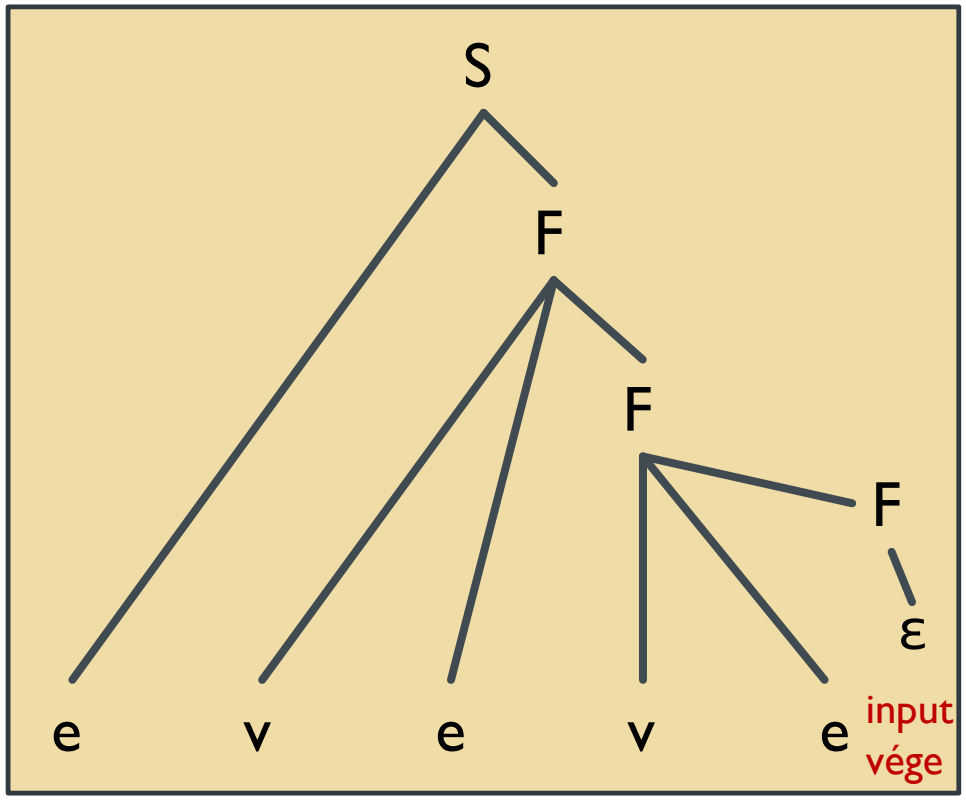
$S \rightarrow \varepsilon \mid eF$
$F \rightarrow \varepsilon \mid veF$



Előreolvasás:
Nézzük meg a következő néhány tokent, és válasszunk szabályt azok alapján!

MELYIK SZABÁLYT KELL ALKALMAZNI?

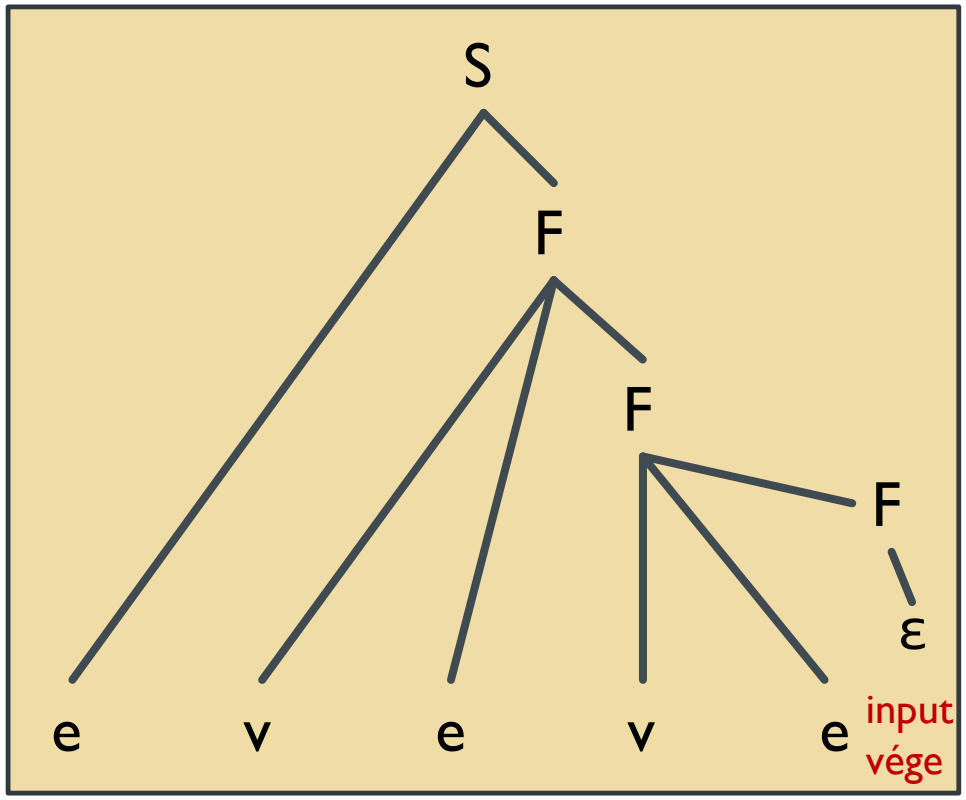
$S \rightarrow \varepsilon \mid eF$
$F \rightarrow \varepsilon \mid veF$



Előreolvasás:
Nézzük meg a következő néhány tokent, és válasszunk szabályt azok alapján!

HÁNY TOKENT KELL ELŐREOLVASNI?

$S \rightarrow \varepsilon \mid eF$
$F \rightarrow \varepsilon \mid veF$



Előreolvasás:
Nézzük meg a következő néhány tokent, és válasszunk szabályt azok alapján!

Hány tokent kell előreolvasni?

- Ez a nyelvtan tulajdonságaitól függ.
- Ennél a nyelvtannál elég egyet.

LL(K) NYELVTANOK

Definíció: LL(k) nyelvtan

Egy környezetfüggetlen nyelvtan *LL(k)* tulajdonságú valamely $k \in \mathbb{N}$ számra, ha felülről lefelé elemzés esetén a legbaloldalibb feldolgozatlan nemterminálishoz egyértelműen meghatározható a rá alkalmazandó nyelvtani szabály legfeljebb *k* token előreolvasásával.

- LL jelentése: Left to right, using Leftmost derivation (Balról jobbra legbaloldalibb levezetéssel)
- Az LL(1) elemzéssel foglalkozunk...

- Van olyan nyelvtan, ami semmilyen $k \in \mathbb{N}$-re sem LL(k), pl.:
  S → A | B
  A → a | aA
  B → ab | aBb

REKURZÍV LESZÁLLÁS

- A rekurzív leszállás az LL(1) elemzés egy implementációja.
- A nyelvtani szabályokat közvetlenül átírjuk függvényekké egy tetszőleges programozási nyelvben.
- Valójában ez az elemző is veremautomata: a függvényhívásokat kezelő futási idejű verem az automata verme.
- Ha a nyelvtan rekurzív, akkor rekurzív vagy kölcsönösen rekurzív függvényeket kapunk, innen a módszer neve.

REKURZÍV LESZÁLLÁS IMPLEMENTÁCIÓJA

$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {



}
```

```
void F() {



}
```

- Minden nemterminálishoz egy-egy függvény.

REKURZÍV LESZÁLLÁS IMPLEMENTÁCIÓJA

$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // S → ε
        } else if(next == e) {
                // S → eF

        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // F → ε
        } else if(next == v) {

                // F → veF

        } else {
                error();
        }
}
```

- Minden nemterminálishoz egy-egy függvény.
- Minden szabályalternatívához egy-egy elágazás-ág. Hibakezelés a különben ágban.

REKURZÍV LESZÁLLÁS IMPLEMENTÁCIÓJA

$S \rightarrow \varepsilon \mid eF$ $\quad$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

- Minden nemterminálishoz egy-egy függvény.
- Minden szabályalternatívához egy-egy elágazás-ág. Hibakezelés a különben ágban.
- Az ágak belsejében a szabály jobboldalának minden szimbólumához egy-egy utasítás:
  - Terminálisokhoz: accept eljáráshívás
  - Nemterminálisokhoz: a hozzájuk tartozó eljárás hívása

REKURZÍV LESZÁLLÁS IMPLEMENTÁCIÓJA

$S \rightarrow \varepsilon \mid eF$ $\quad$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

- Minden nemterminálishoz egy-egy függvény.
- Minden szabályalternatívához egy-egy elágazás-ág. Hibakezelés a különben ágban.
- Az ágak belsejében a szabály jobboldalának minden szimbólumához egy-egy utasítás:
  - Terminálisokhoz: accept eljáráshívás
  - Nemterminálisokhoz: a hozzájuk tartozó eljárás hívása

```
void accept(token t) {
        if(next == t) {
                next = lexer.next();
        } else {
                error();
        }
}
```

Accept eljárás:

- Ha a várt token következik, akkor új token kérése a lexikális elemzőtől
- Egyébként hiba

REKURZÍV LESZÁLLÁS IMPLEMENTÁCIÓJA: HOGYAN HATÁROZZUK MEG AZ ELÁGAZÁSOK FELTÉTELEIT?

$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

- Minden nemterminálishoz egy-egy függvény.
- Minden szabályalternatívához egy-egy elágazás-ág. Hibakezelés a különben ágban.
- Az ágak belsejében a szabály jobboldalának minden szimbólumához egy-egy utasítás:
  - Terminálisokhoz: accept eljáráshívás
  - Nemterminálisokhoz: a hozzájuk tartozó eljárás hívása

```
void accept(token t) {
        if(next == t) {
                next = lexer.next();
        } else {
                error();
        }
}
```

Accept eljárás:

- Ha a várt token következik, akkor új token kérése a lexikális elemzőtől
- Egyébként hiba

AZ ELÁGAZÁSOK FELTÉTELEINEK MEGHATÁROZÁSA: FIRST HALMAZ

Definíció: $FIRST_1$ halmaz

Adott nyelvtan esetén egy α szimbólumsorozatra a $FIRST_1(\alpha)$ halmaz azokat a **terminálisokat** tartalmazza, amelyek az α -ból levezethető szimbólumsorozatok elején állnak. Ha α -ból levezethető az üres szó (ε), akkor ε is eleme a halmaznak.

S → ε | eF
F → ε | veF

$FIRST_1(\varepsilon) = \{\varepsilon\}$
$FIRST_1(eF) = \{e\}$
$FIRST_1(veF) = \{v\}$
$FIRST_1(F) = \{v, \varepsilon\}$

ε
eF
veF
$F \Rightarrow \varepsilon$ $F \Rightarrow vef$

A FIRST halmaz általánosan *n* hosszú eredménysorozatokra is definiálható: $FIRST_n(\alpha)$

AZ ELÁGAZÁSOK FELTÉTELEINEK MEGHATÁROZÁSA: FOLLOW HALMAZ

Definíció: FOLLOW$_1$ halmaz

Adott nyelvtan esetén egy α szimbólumsorozatra a $FOLLOW_1(\alpha)$ halmaz azokat a **terminálisokat** tartalmazza, amelyek az α után állhatnak a kezdőszimbólumból induló levezetésekben. Ha α után nem áll semmi, akkor # (szöveg vége jel) eleme a halmaznak.

S → ε | eF
F → ε | veF

FOLLOW$_1$(S) = {#}
FOLLOW$_1$(F) = {#}
FOLLOW$_1$(e) = {#, v}

S
S ⇒ eF ⇒ veF ⇒ …
S ⇒ eF ⇒ e
S ⇒ eF ⇒ eveF

A FOLLOW halmaz általánosan *n* hosszú eredménysorozatokra is definiálható: FIRST$_n$(α)

AZ ELÁGAZÁSOK FELTÉTELEINEK MEGHATÁROZÁSA

- Az $A \rightarrow \alpha$ szabályhoz meghatározzuk a $FIRST_1(\alpha)$ halmazt.
- Ha ebben van ε, akkor kivesszük ε-t, és helyette hozzávesszük a halmazhoz $FOLLOW_1(A)$ elemeit.
- Az így kapott halmaz elemeiből (pl. $x_1, x_2, ..., x_n$) képezzük az elágazás feltételét:
  next == x_1 || next == x_2 || ... || next == x_n

$S \rightarrow \varepsilon$
$FIRST_1(\varepsilon) = \varepsilon$
$FOLLOW_1(S) = \{\#\}$
Eredmény: {#}

$S \rightarrow eF$
$FIRST_1(eF) = e$
Eredmény: {e}

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

$F \rightarrow \varepsilon$
$FIRST_1(\varepsilon) = \varepsilon$
$FOLLOW_1(F) = \{\#\}$
Eredmény: {#}

$F \rightarrow veF$
$FIRST_1(veF) = v$
Eredmény: {v}

LL(1) TULAJDONSÁG ELLENŐRZÉSE

- Az $A \rightarrow \alpha$ szabályhoz meghatározzuk a $FIRST_1(\alpha)$ halmazt.
- Ha ebben van ε, akkor kivesszük ε-t, és helyette hozzávesszük a halmazhoz $FOLLOW_1(A)$ elemeit.
- Az így kapott halmaz elemeiből (pl. $x_1, x_2, ..., x_n$) képezzük az elágazás feltételét:
  next == x_1 || next == x_2 || ... || next == x_n

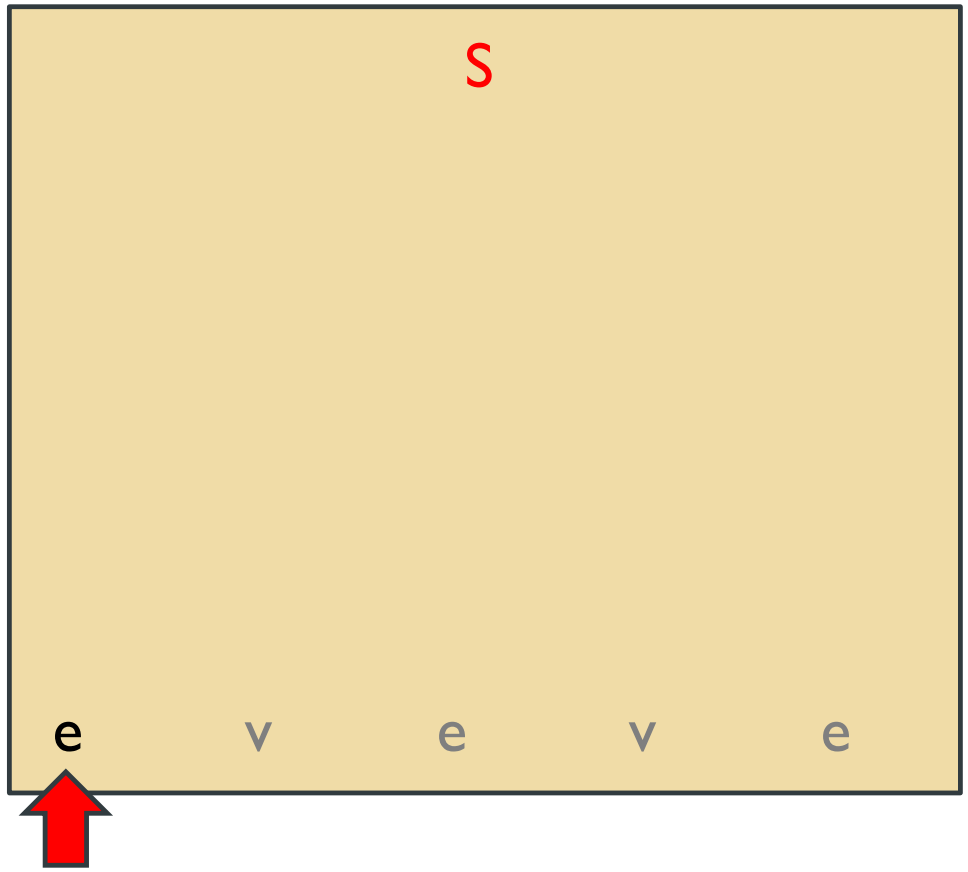
Tétel: LL(1) tulajdonság ellenőrzése

Egy környezetfüggetlen nyelvtan pontosan akkor LL(1) tulajdonságú, ha bármely két $A \rightarrow \alpha, A \rightarrow \beta$ (azonos baloldalú!) szabályokhoz a fenti módon megtározott halmazok diszjunktak.

S → ε	{#}	F → ε	{#}	
S → eF	{e}	F → veF	{v}	LL(1)
{#} ∩ {e} = ∅ ✓		{#} ∩ {v} = ∅ ✓		

S → ε	{#}	N → e	{e}	Nem LL(1)
S → N	{e}	N → Nve	{e}	
{#} ∩ {e} = ∅ ✓		{e} ∩ {e} ≠ ∅ ✗		

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

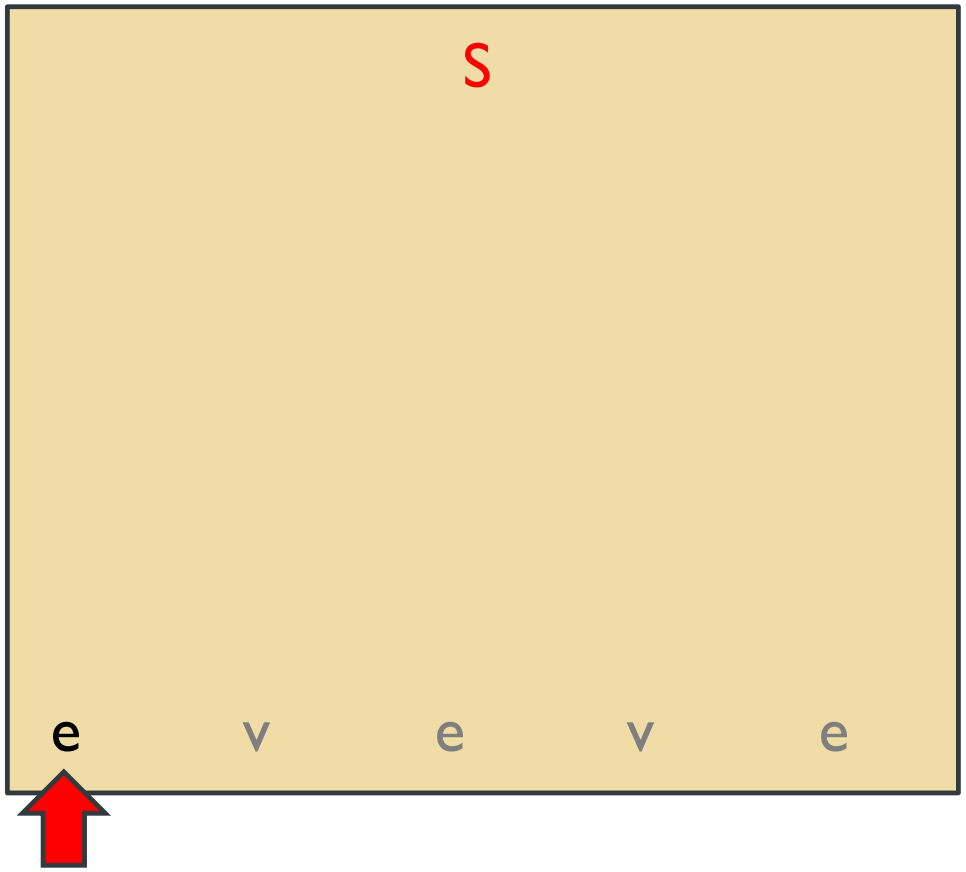


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

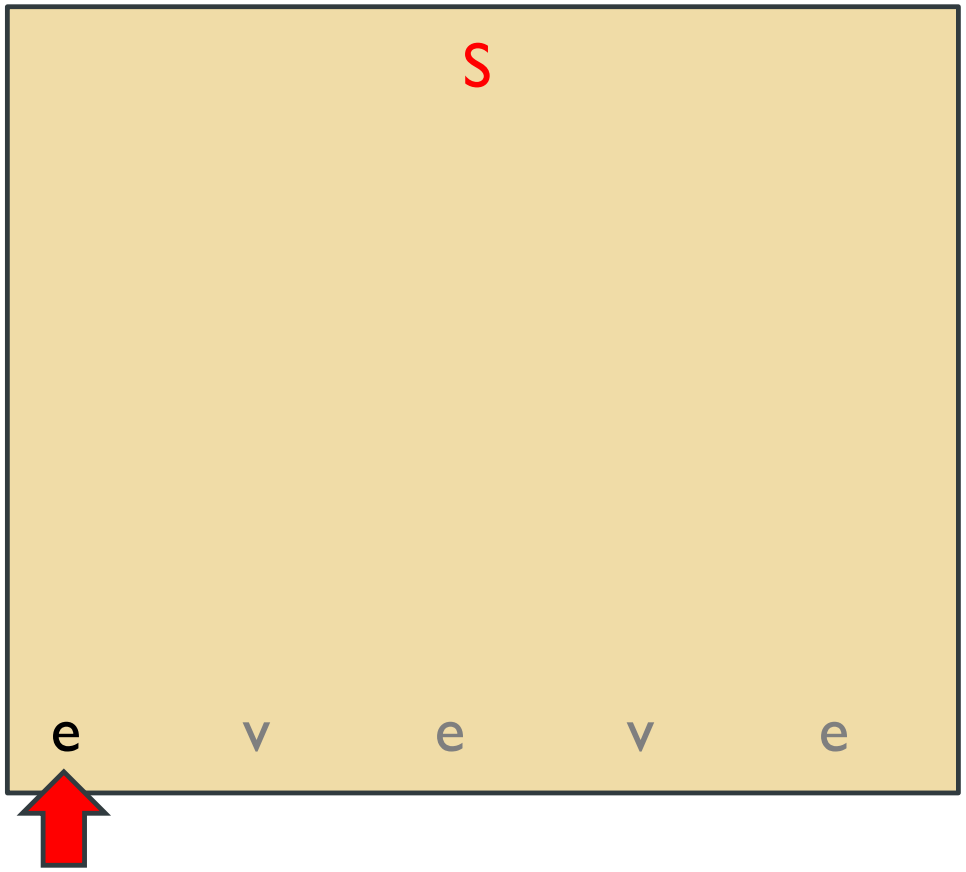


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

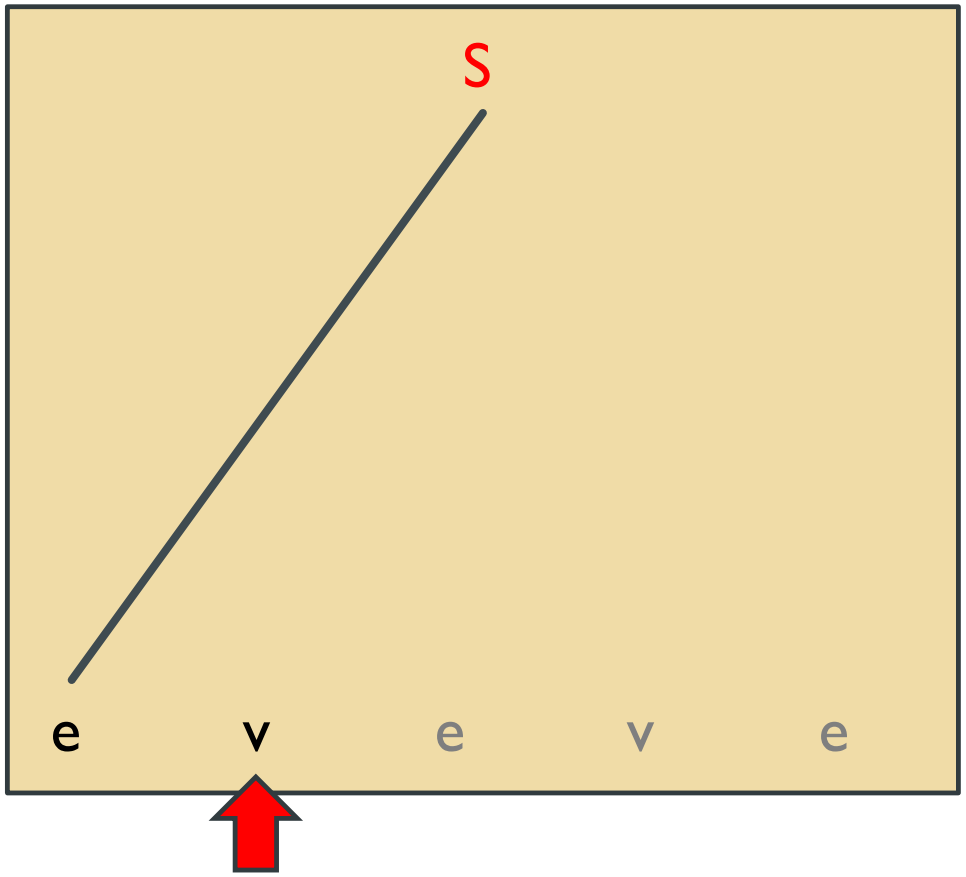


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

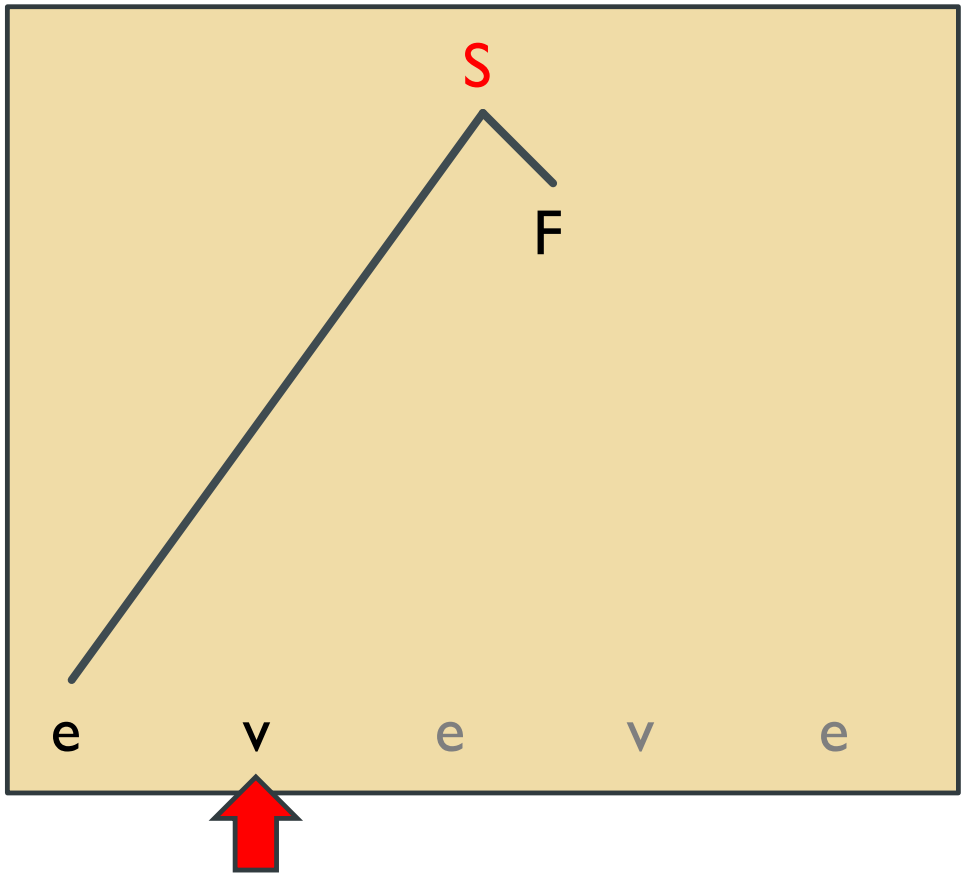


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

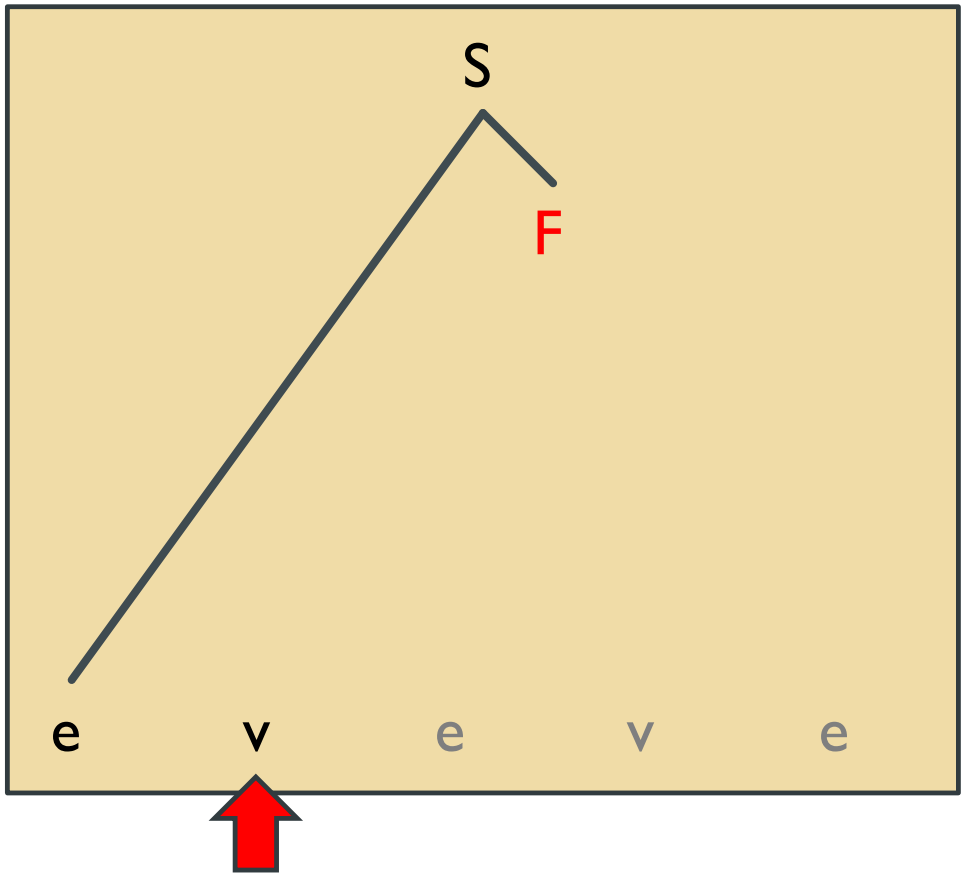


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

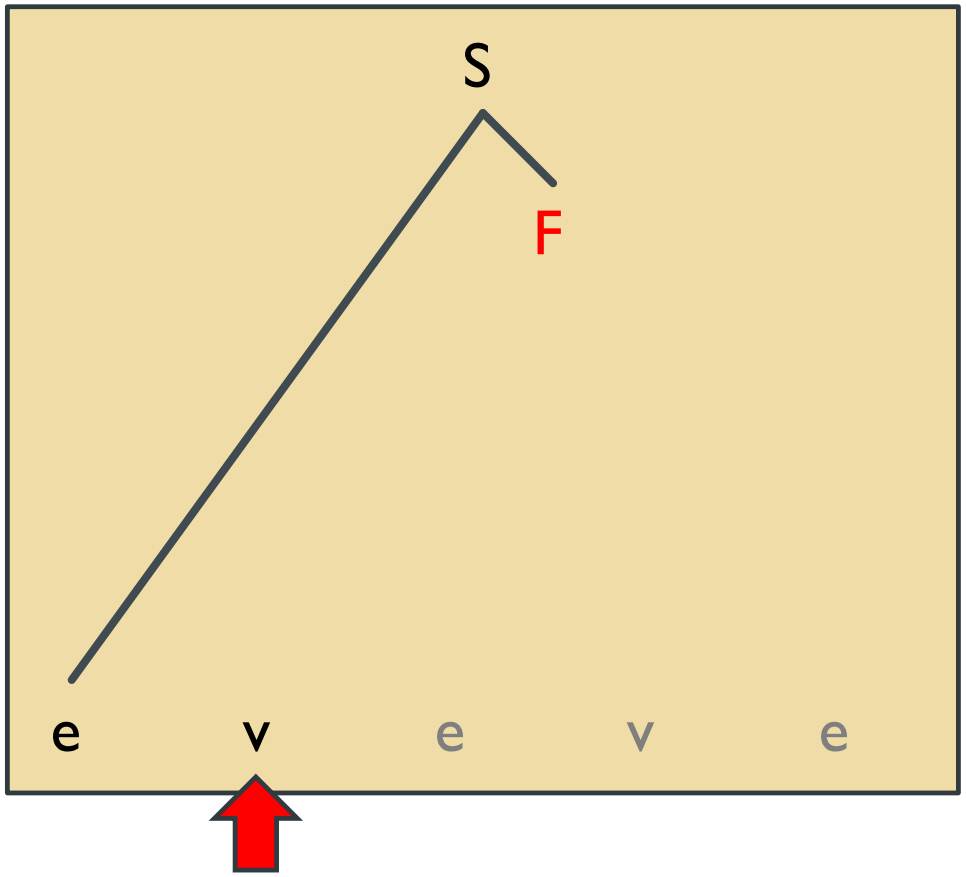


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

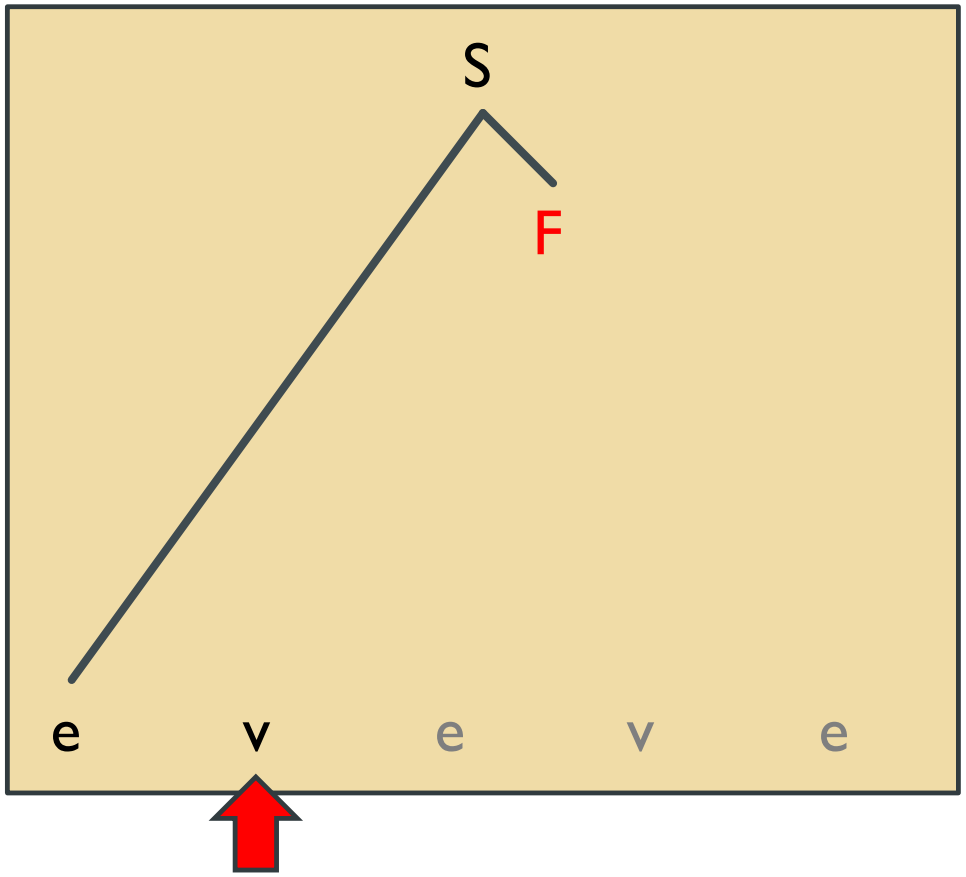


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

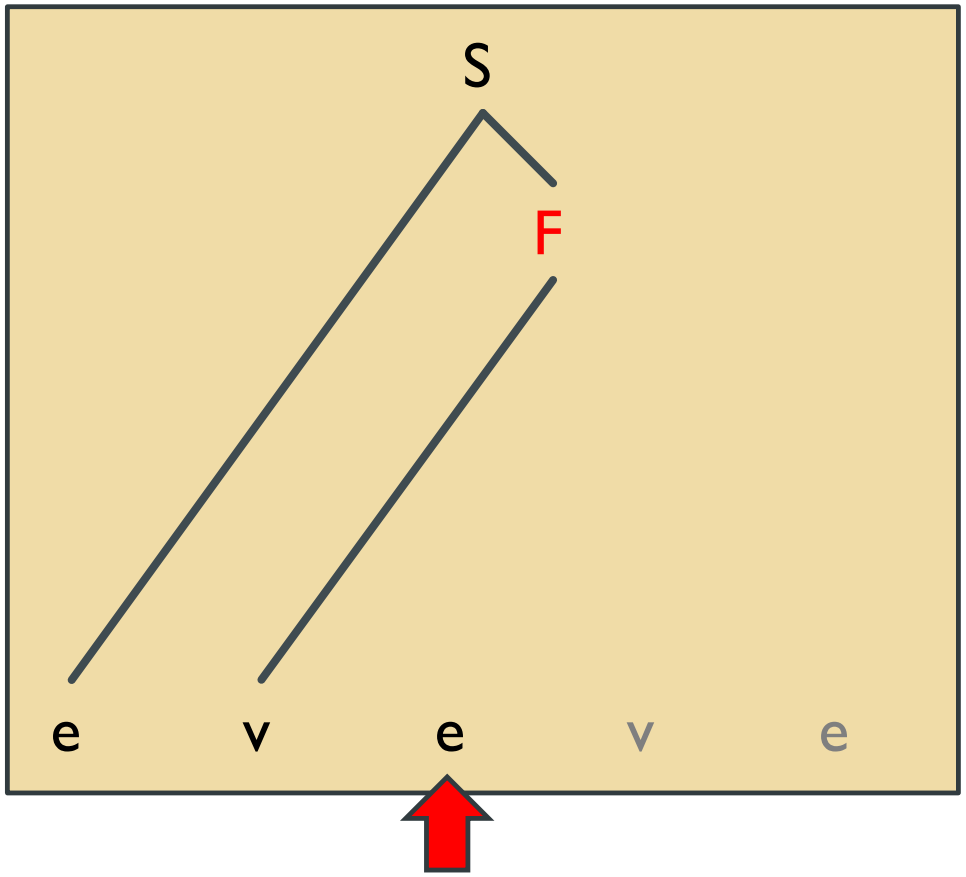


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

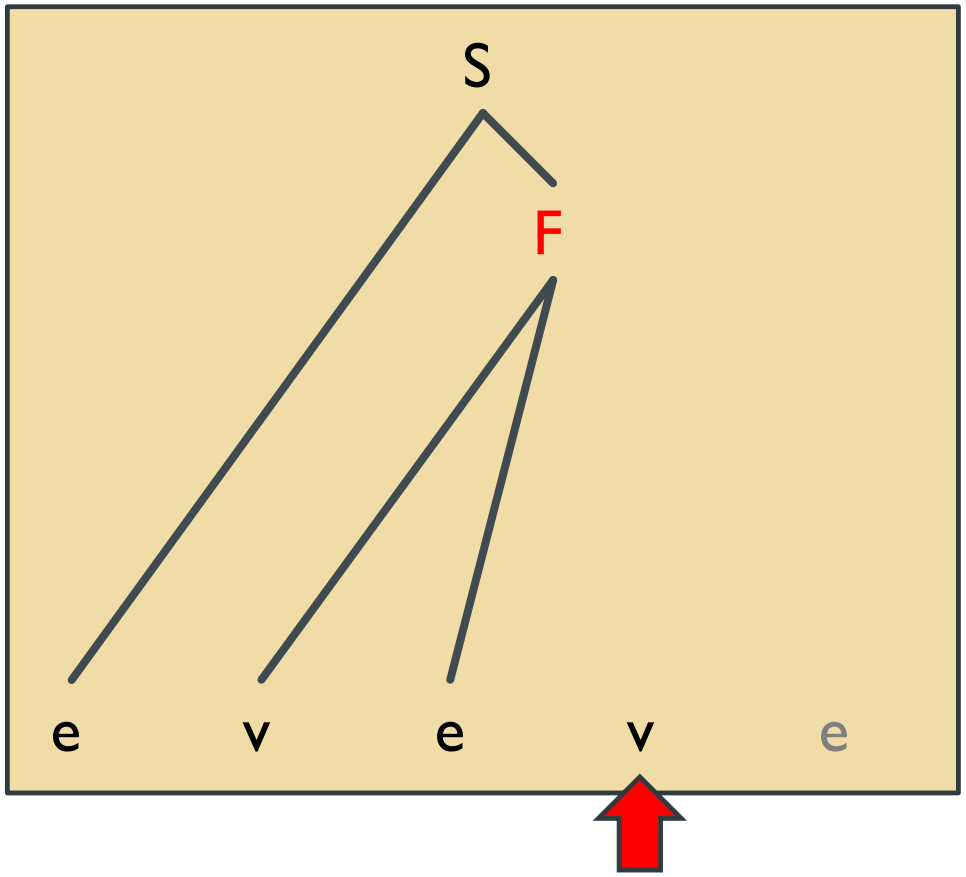


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

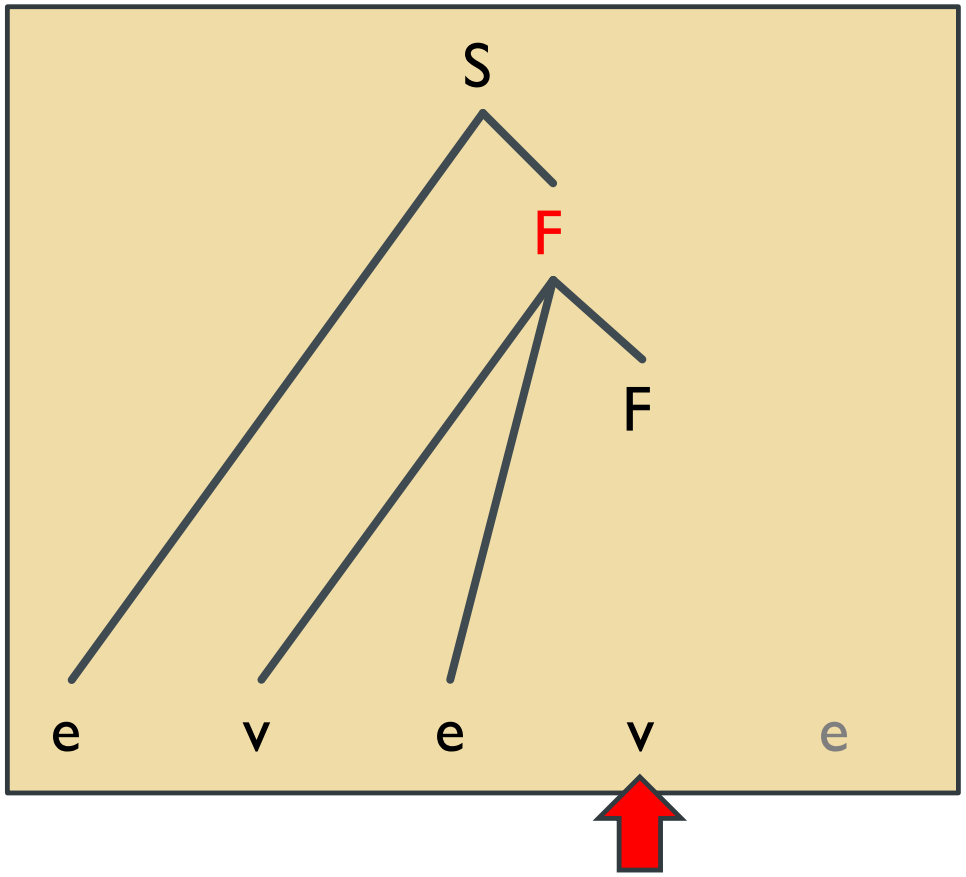


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

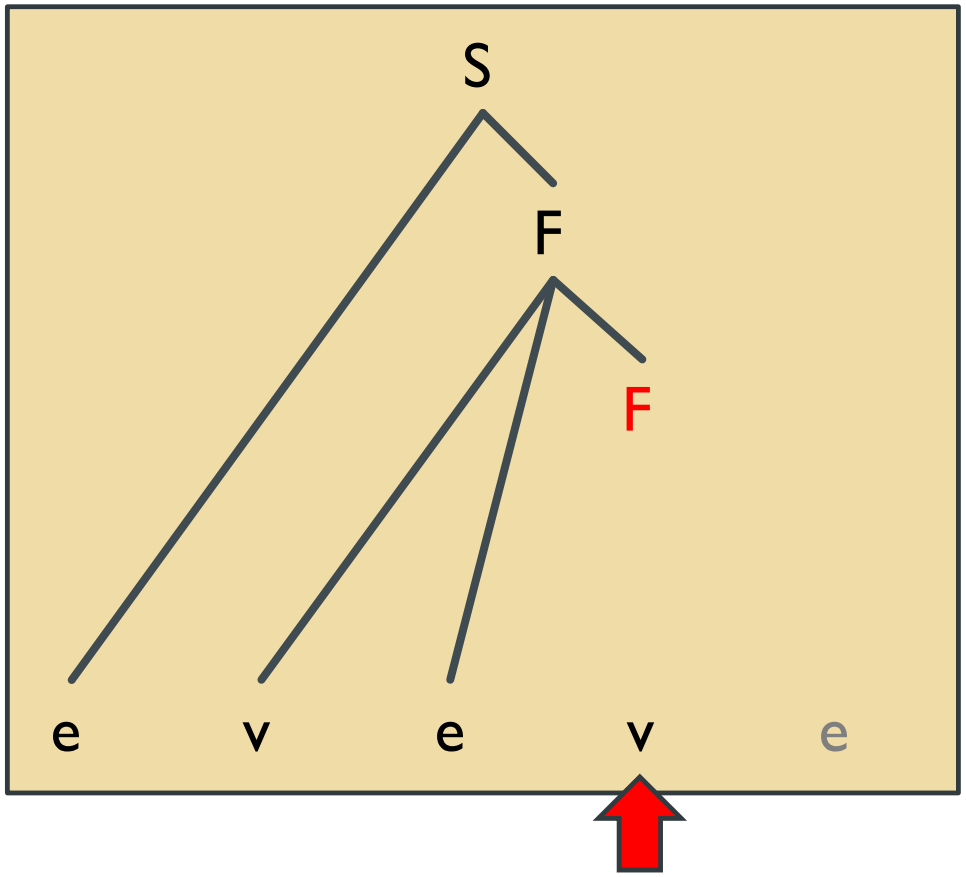


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

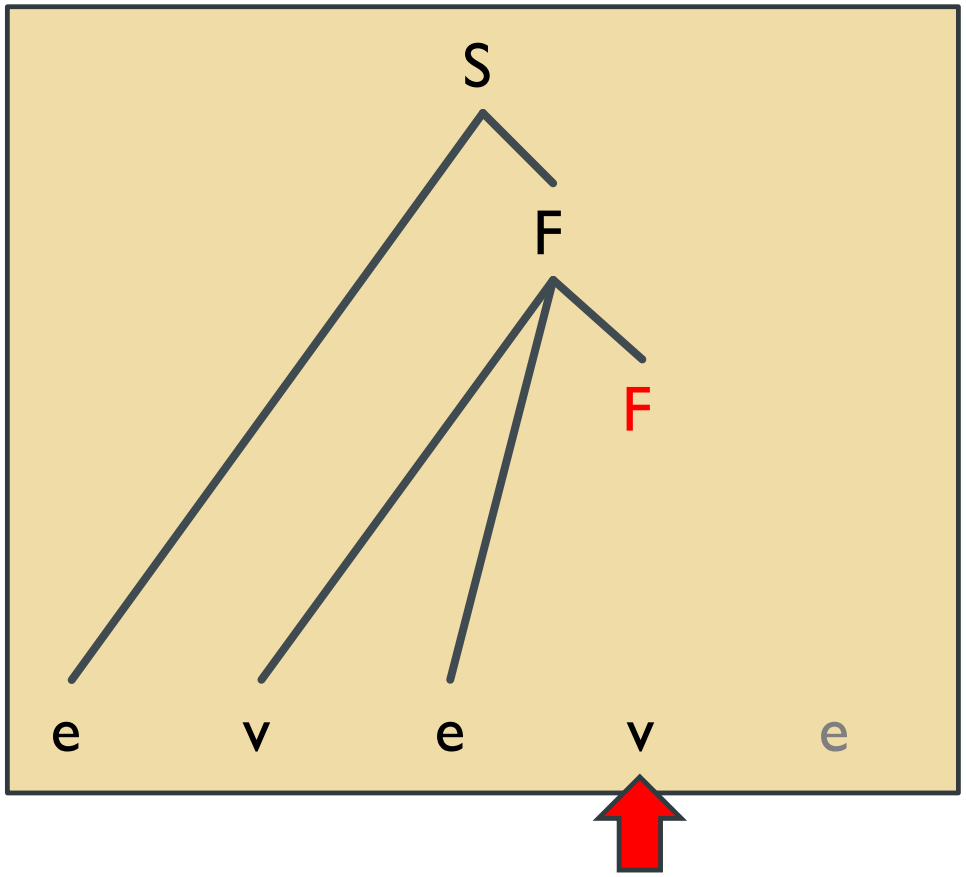


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

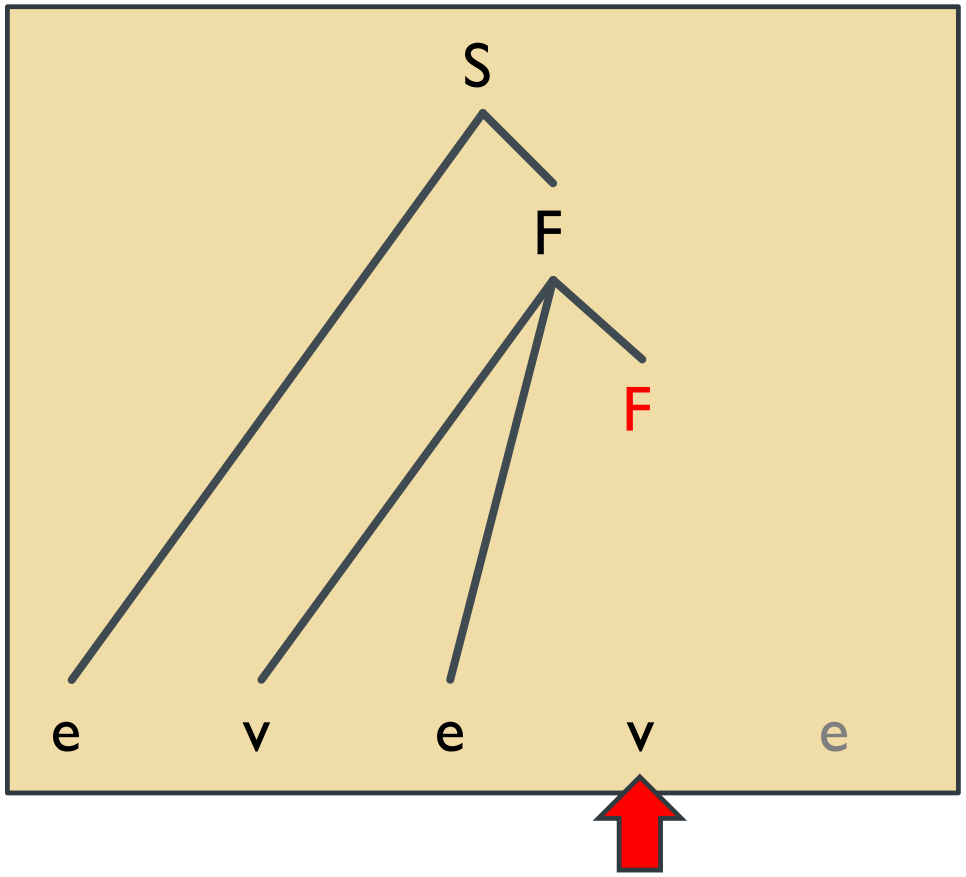


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

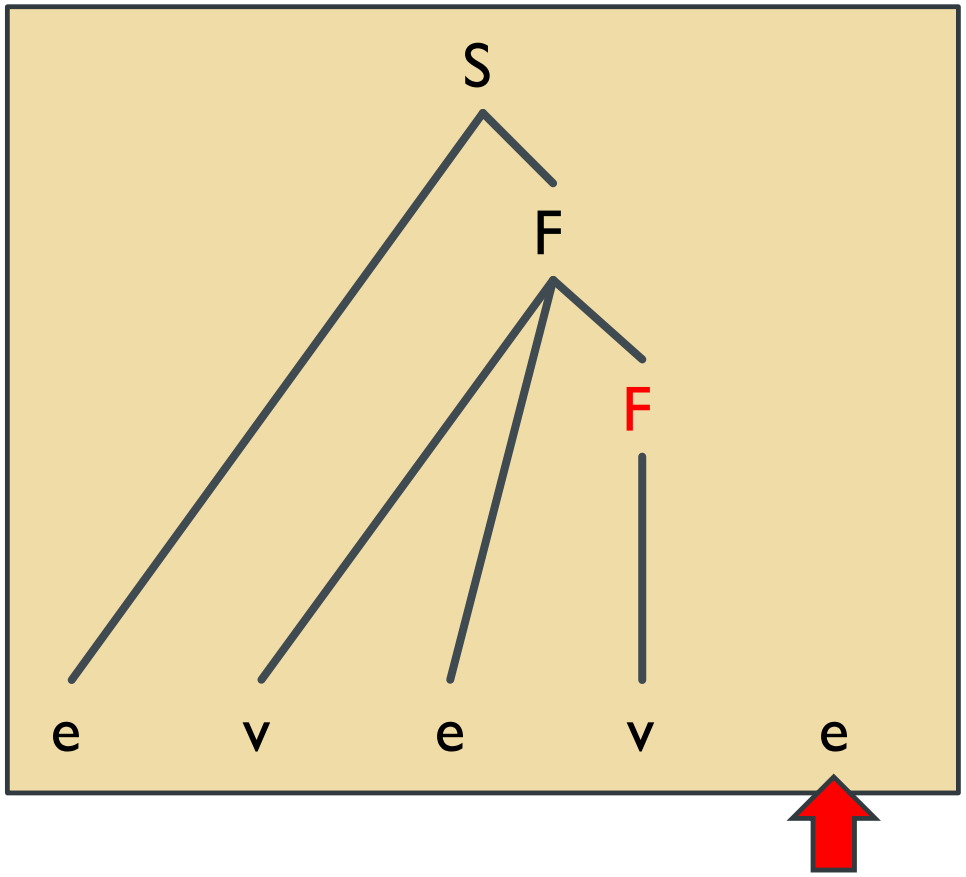


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

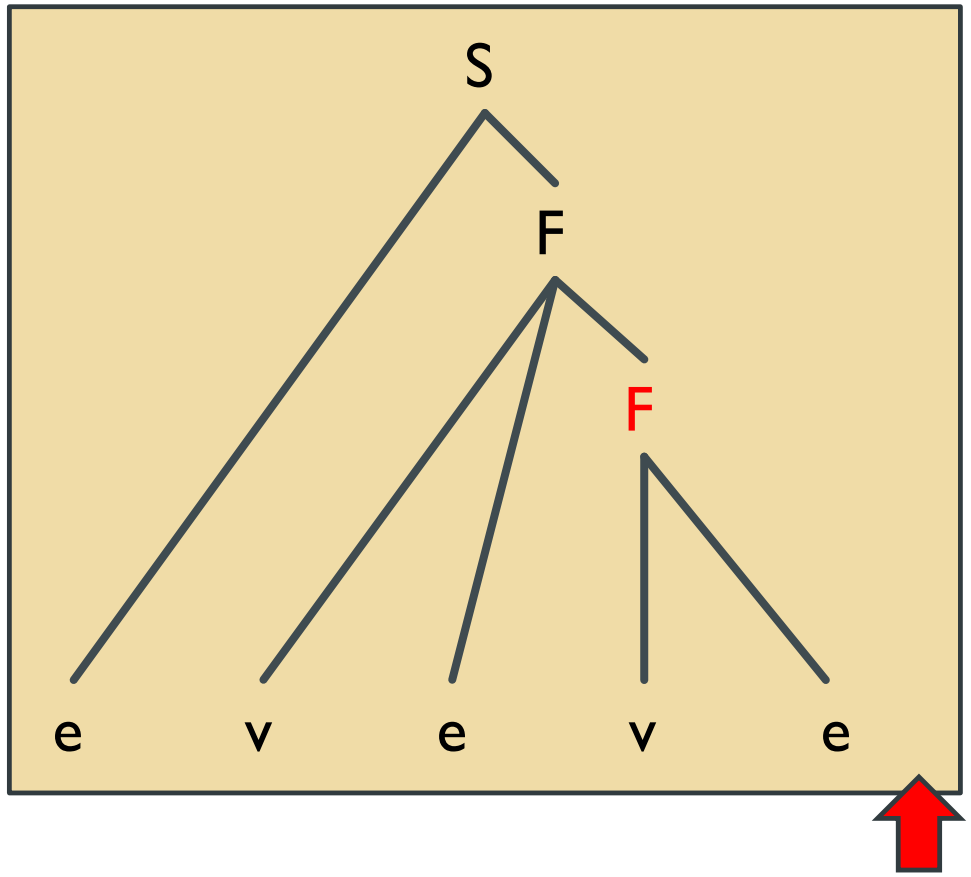


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

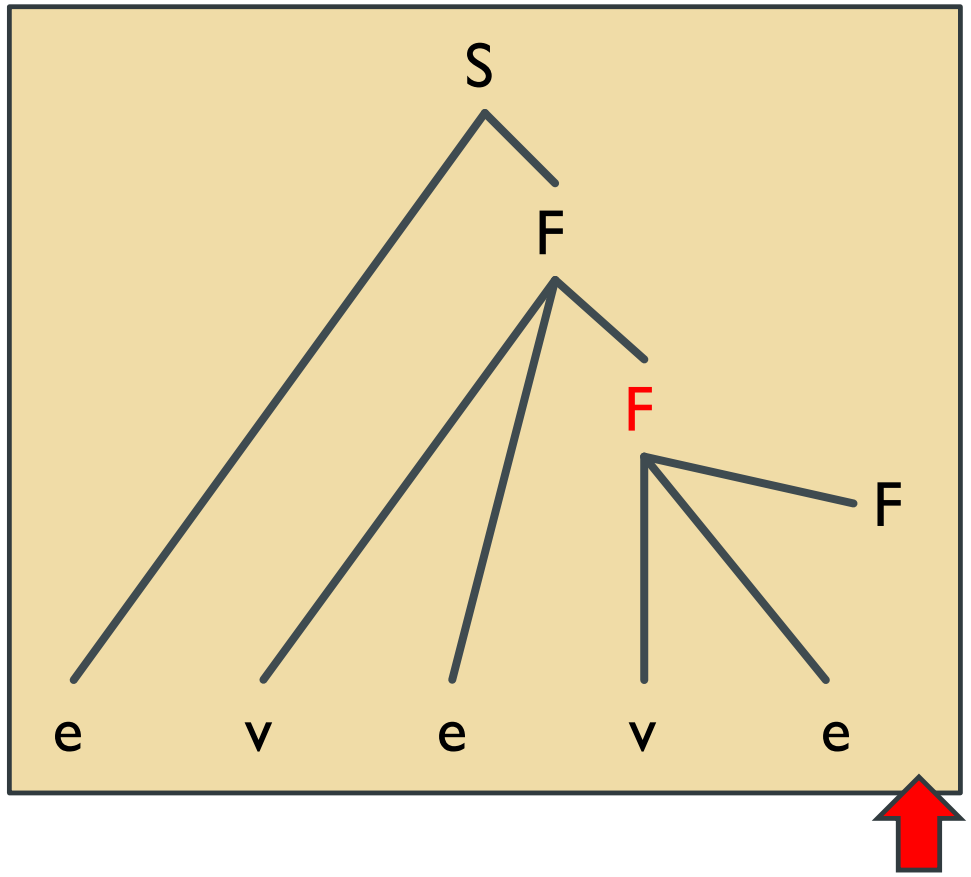


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

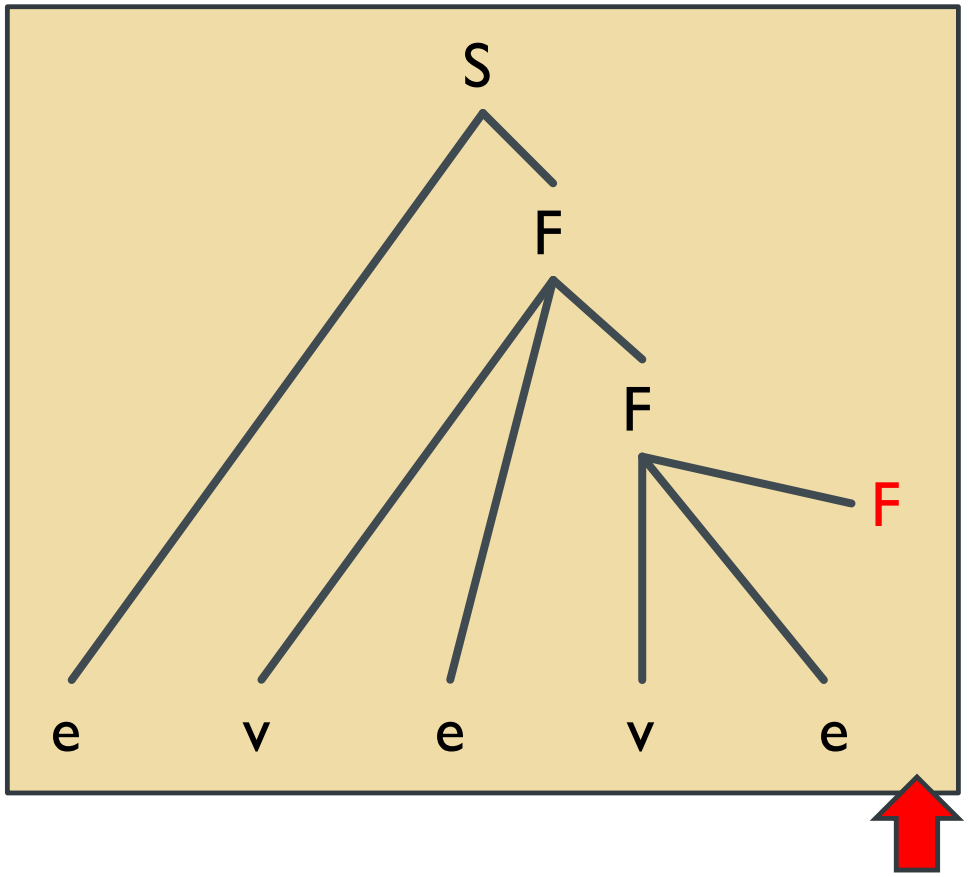


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

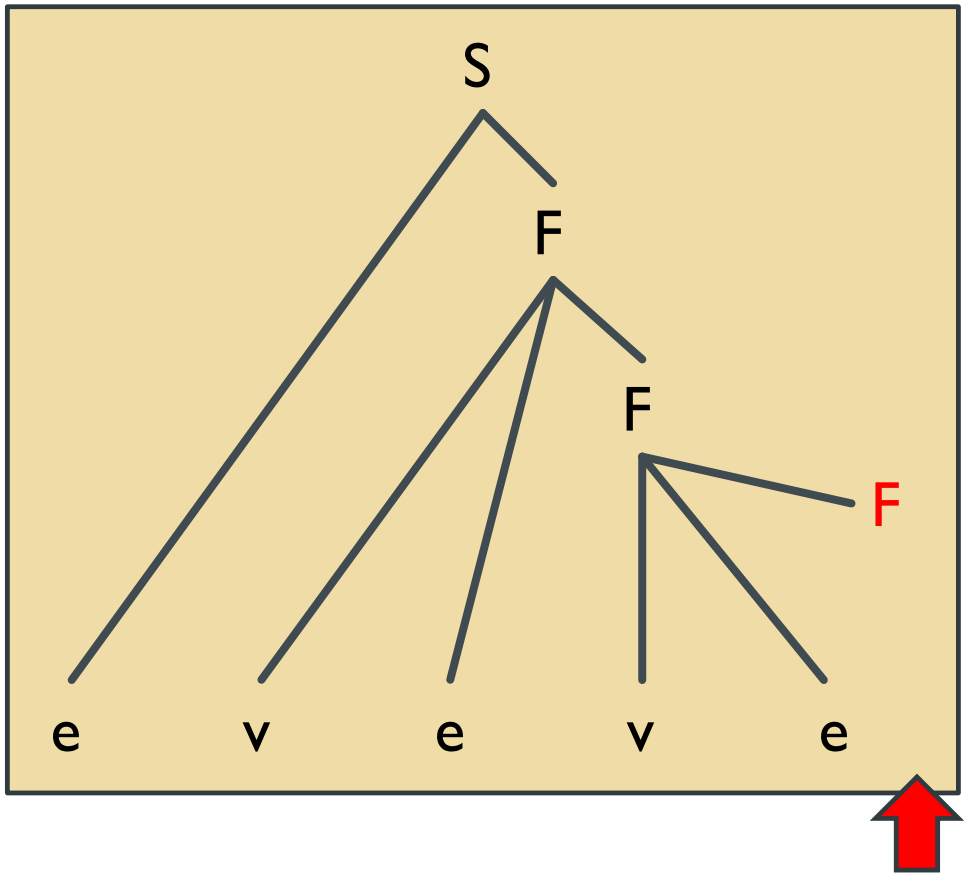


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

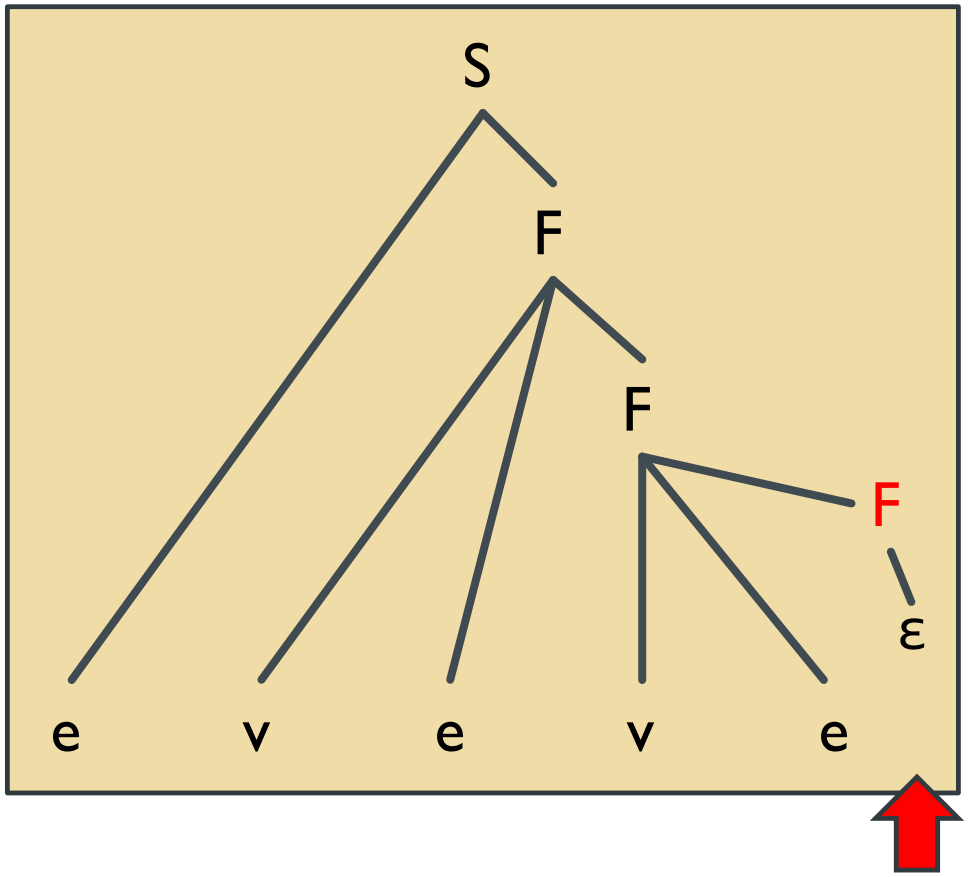


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

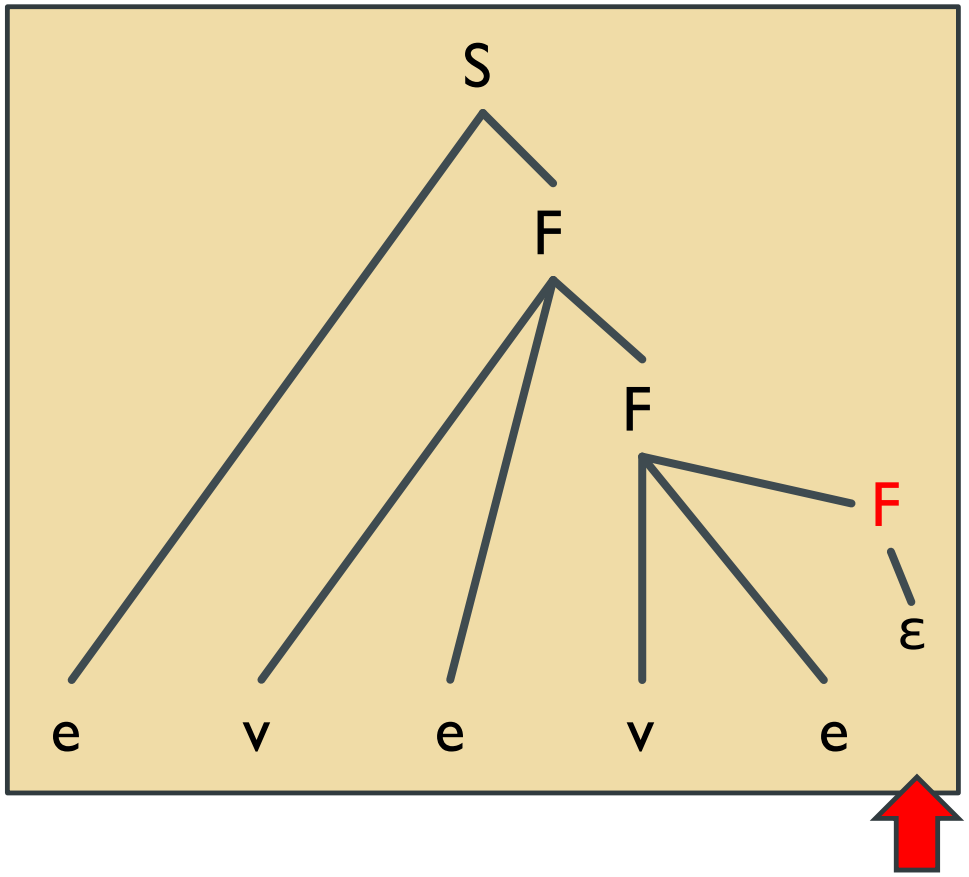


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

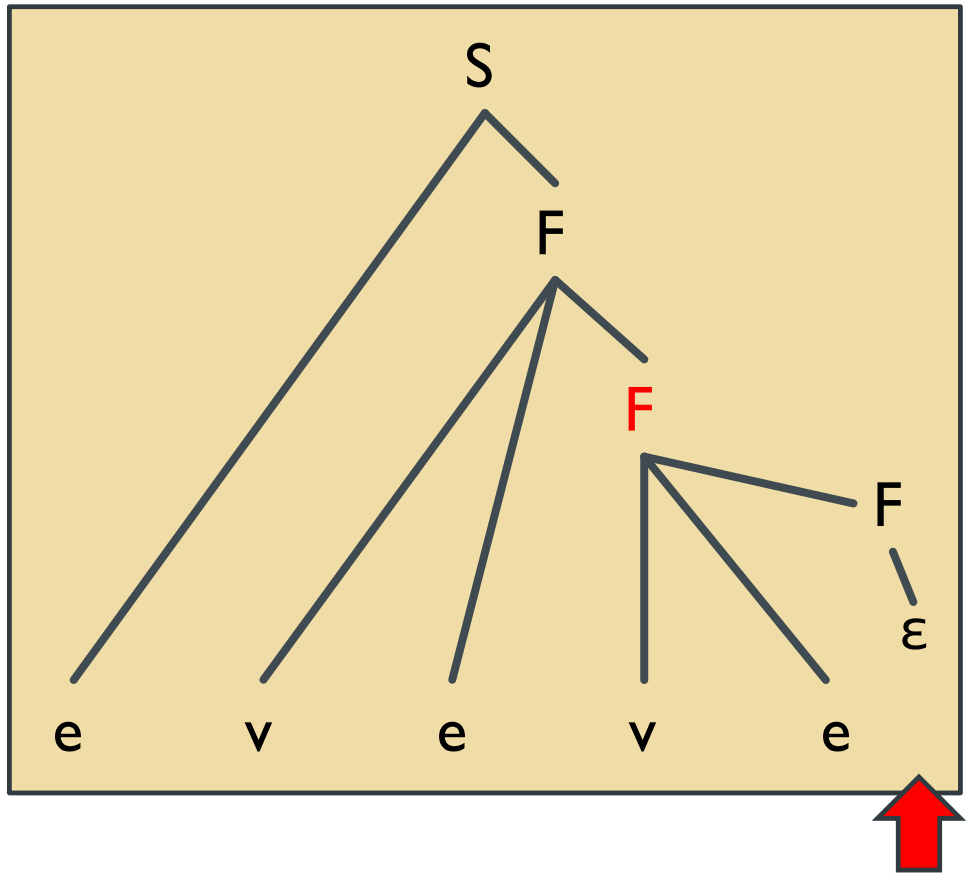


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

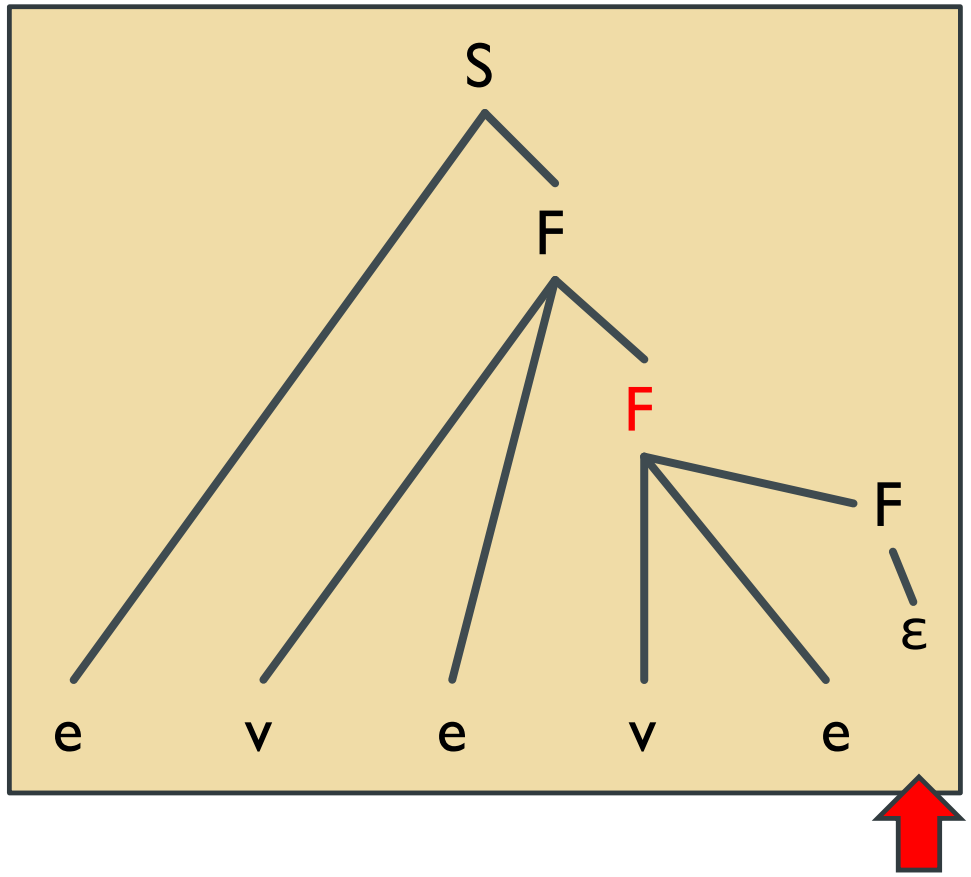


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

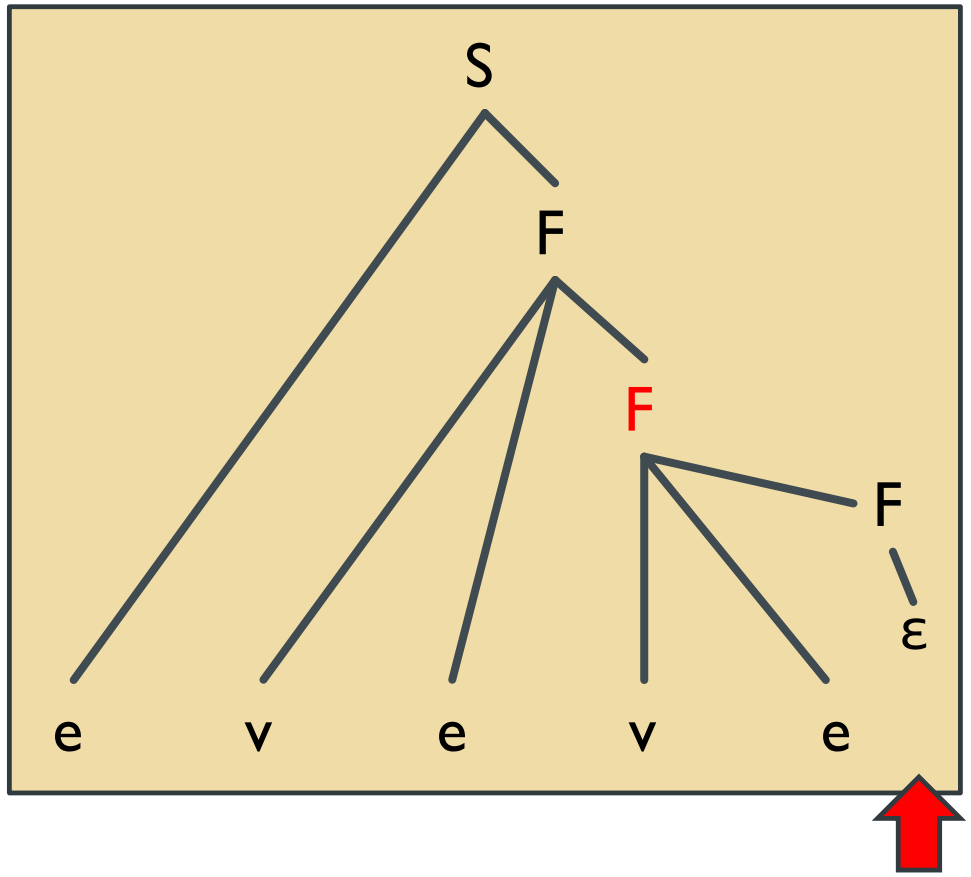


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

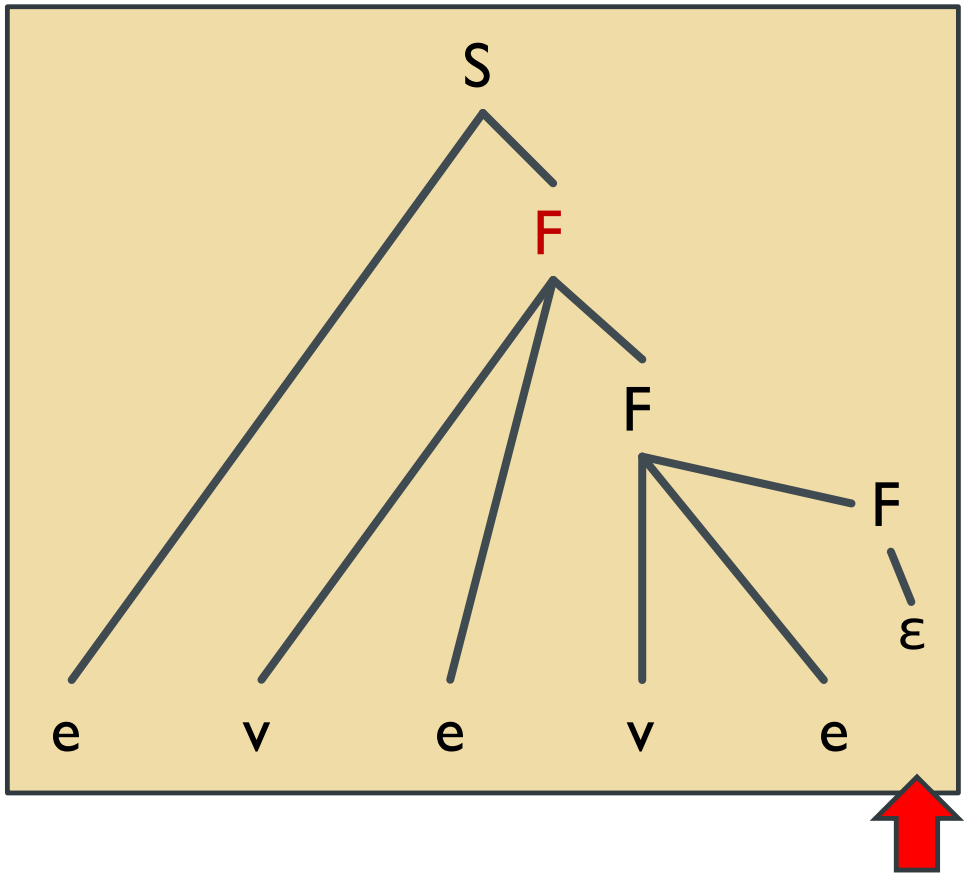


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

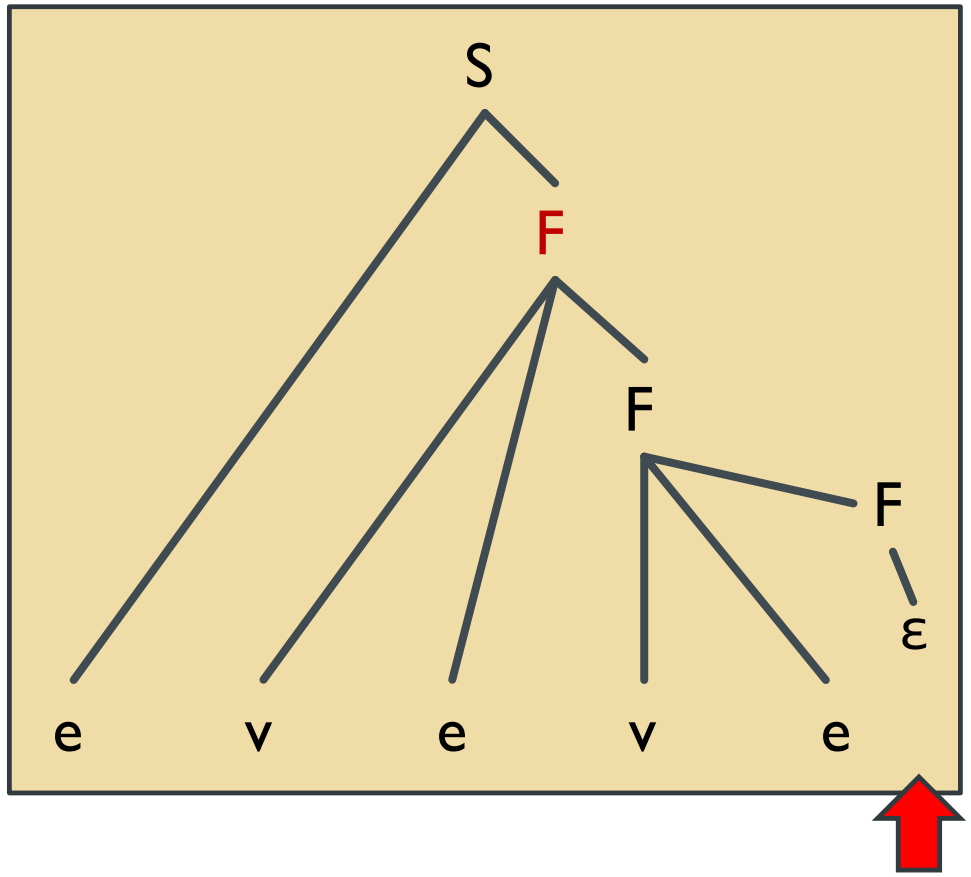


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

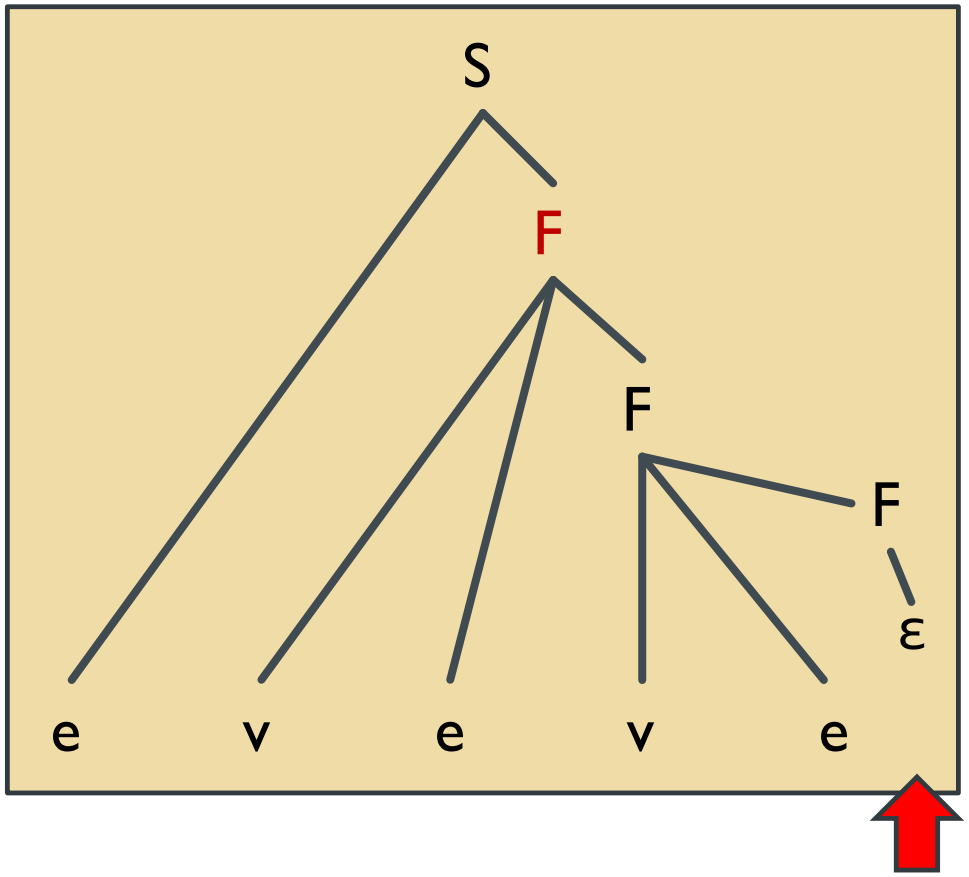


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

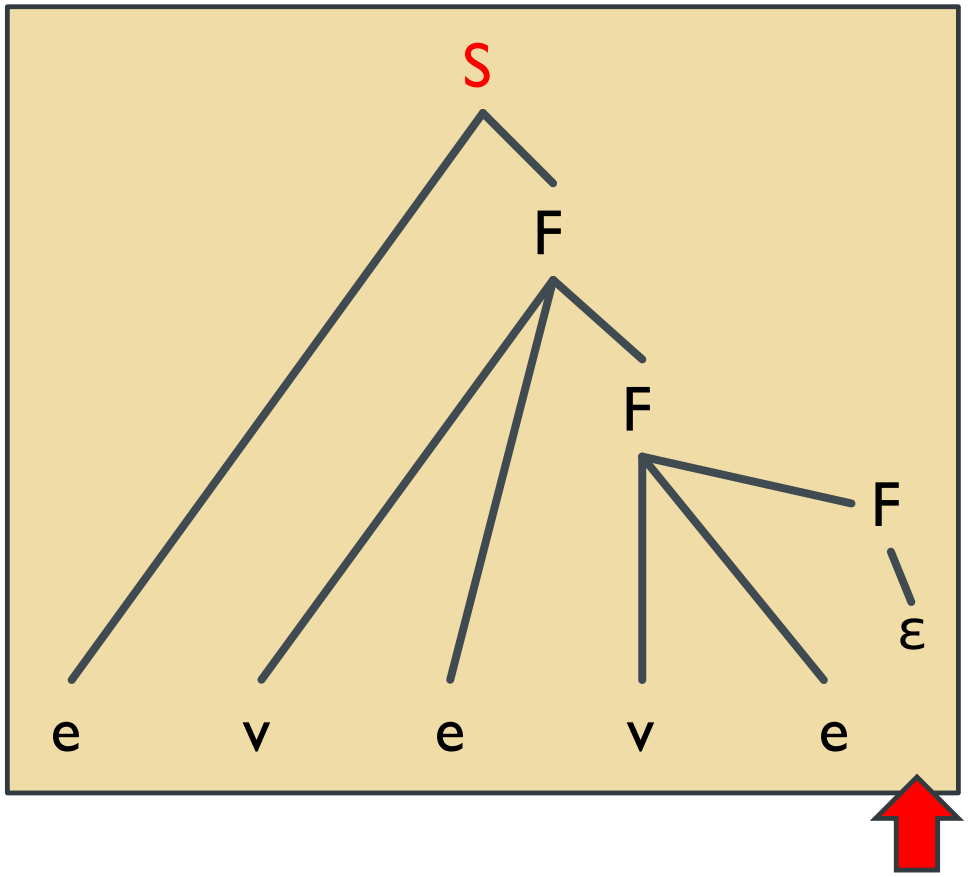


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

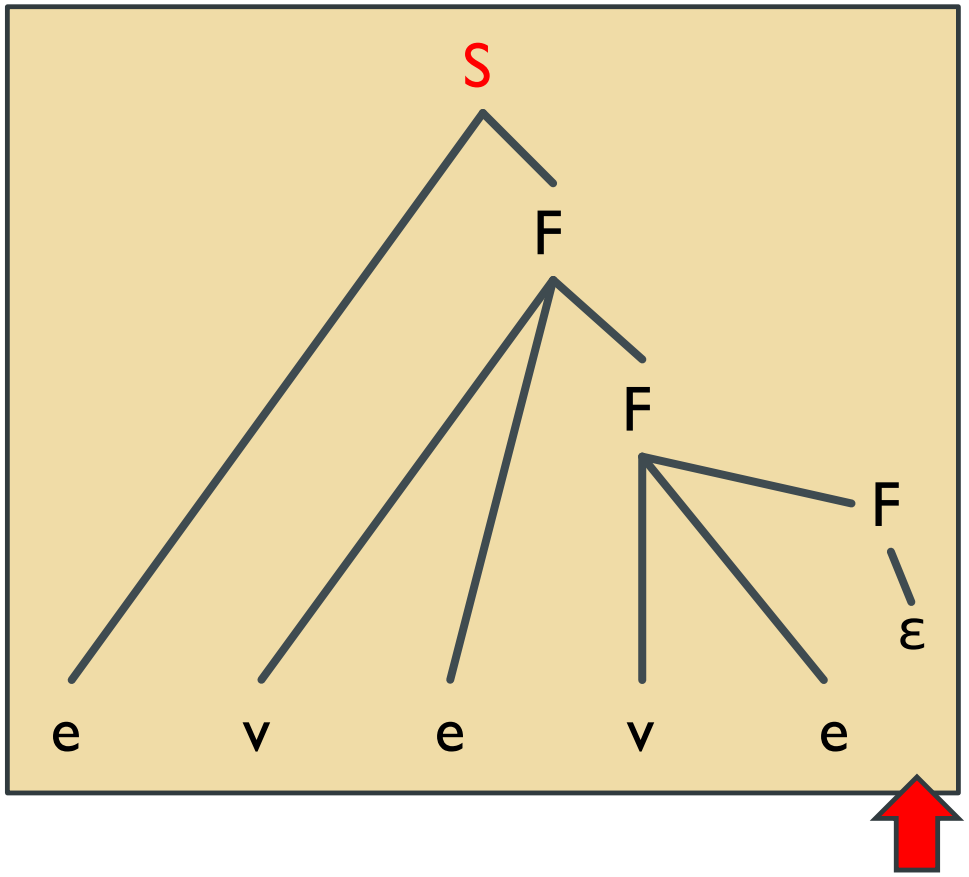


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE

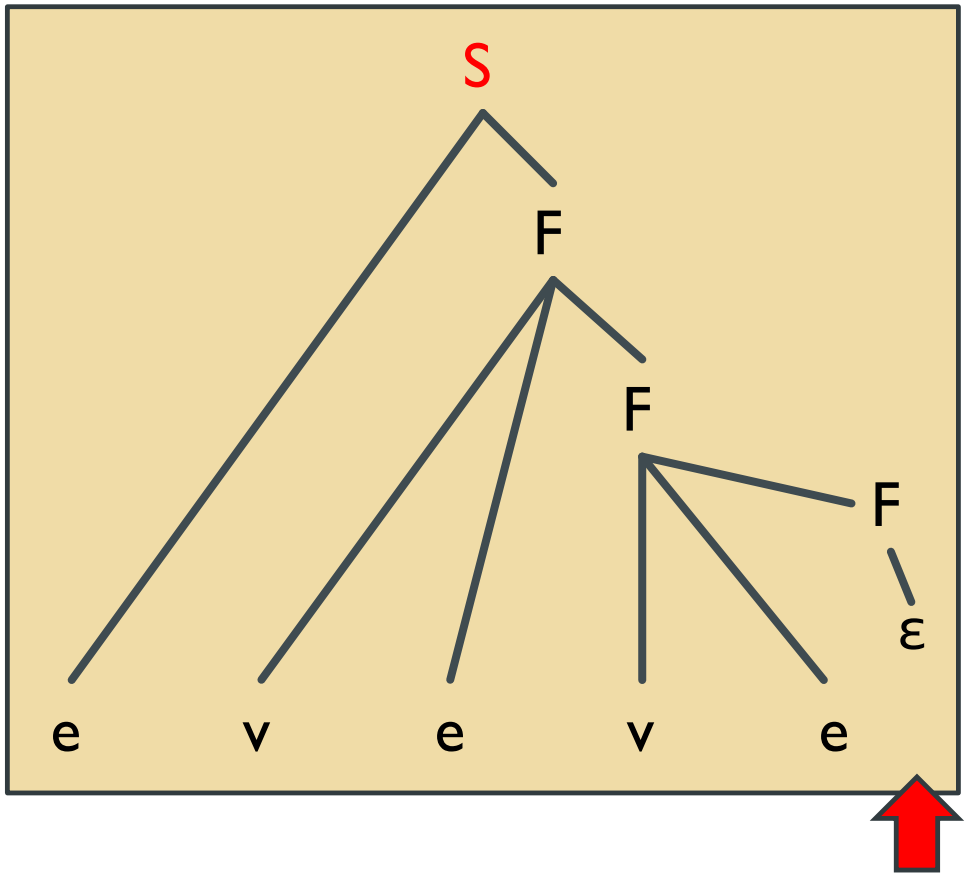


$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

REKURZÍV LESZÁLLÁS MŰKÖDÉSE



$S \rightarrow \varepsilon \mid eF$ $F \rightarrow \varepsilon \mid veF$

```
void S() {
        if(next == end) {
                // empty
        } else if(next == e) {
                accept(e);
                F();
        } else {
                error();
        }
}
```

```
void F() {
        if(next == end) {
                // empty
        } else if(next == v) {
                accept(v);
                accept(e);
                F();
        } else {
                error();
        }
}
```

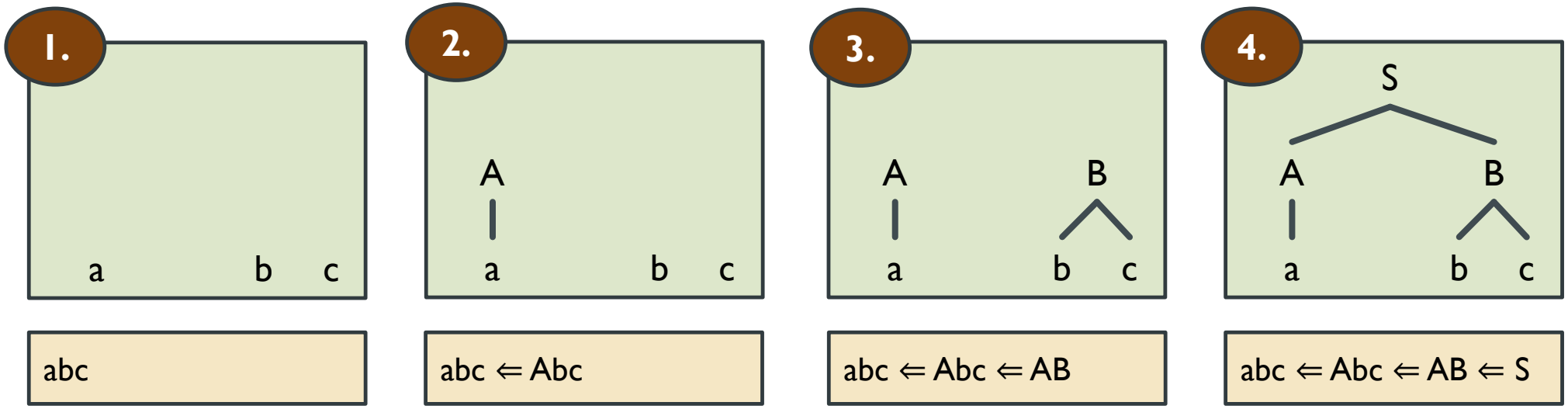
ALULRÓL FELFELÉ ELEMZÉS

LR ELEMZÉSEK

EMLÉKEZTETŐ: ALULRÓL FELFELÉ ELEMZÉS

Nyelvtan:
S → AB
A → a
B → bc

Input: abc



- A terminálisoktól indulva a startszimbólum felé építjük a fát
- Az input feldolgozása balról jobbra történik itt is
- Legjobboldalibb levezetés inverzét állítja elő
  - A *legjobboldalibb levezetés* több terminális esetén a legjobboldalibbat helyettesíti: S ⇒ AB ⇒ Abc ⇒ abc

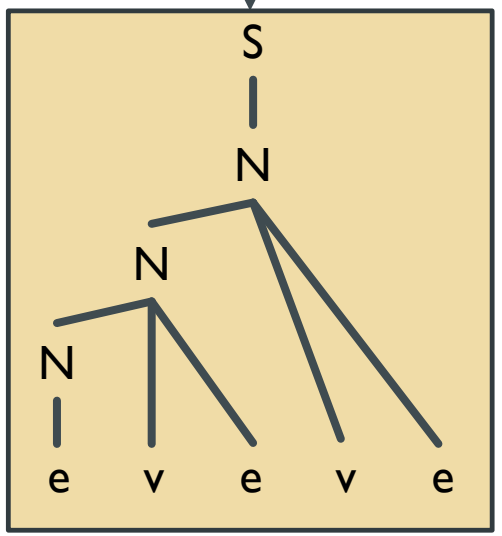
PÉLDA NYELVTAN:VESSZŐVEL ELVÁLASZTOTT LISTA (MÁSIK NYELVTANNAL!)

apple, banana, pear

Lexikális elemző:
[a-z]+ → e (elem)
“,” → v (vessző)
[\t\n] →

eveve

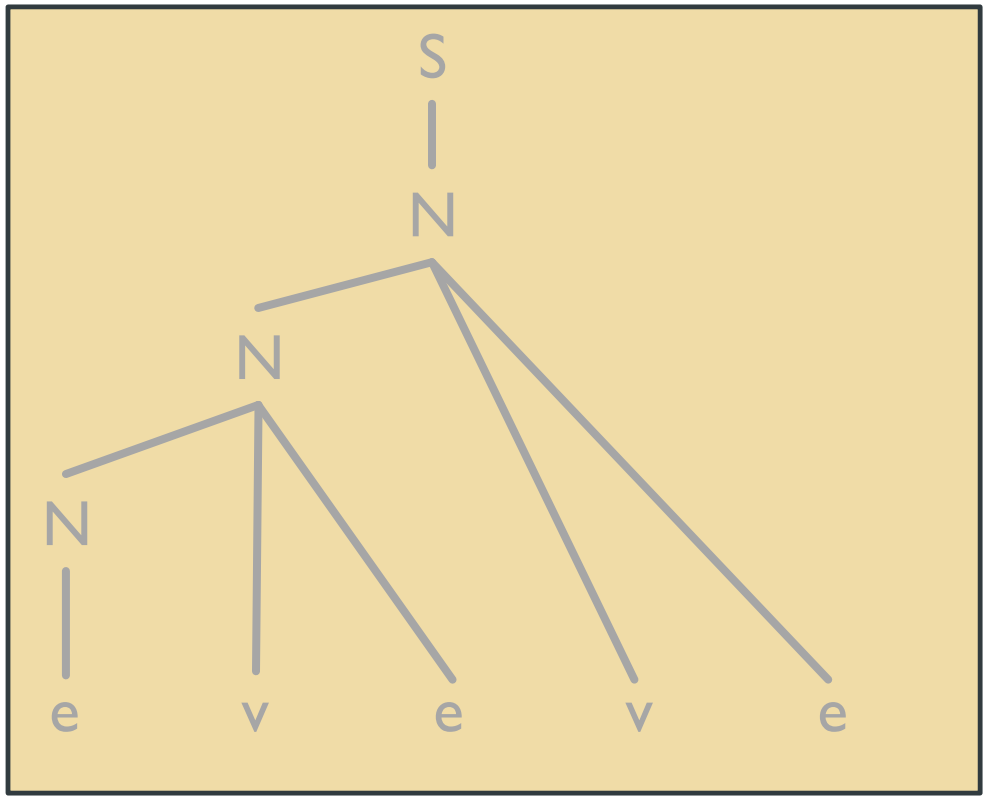
Szintaktikus elemző:
S → ε | N
N → e | Nve



AZ LR ELEMZÉSEK ALAPMŰVELETEI: LÉPTETÉS, REDUKCIÓ

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme:



Az elemző verme:
A szimbólumokat ebben gyűjtjük egészen addig, amíg a megfelelő szabályjobboldal megjelenik benne.

Léptetés:
A következő token elhelyezése a verem tetején.

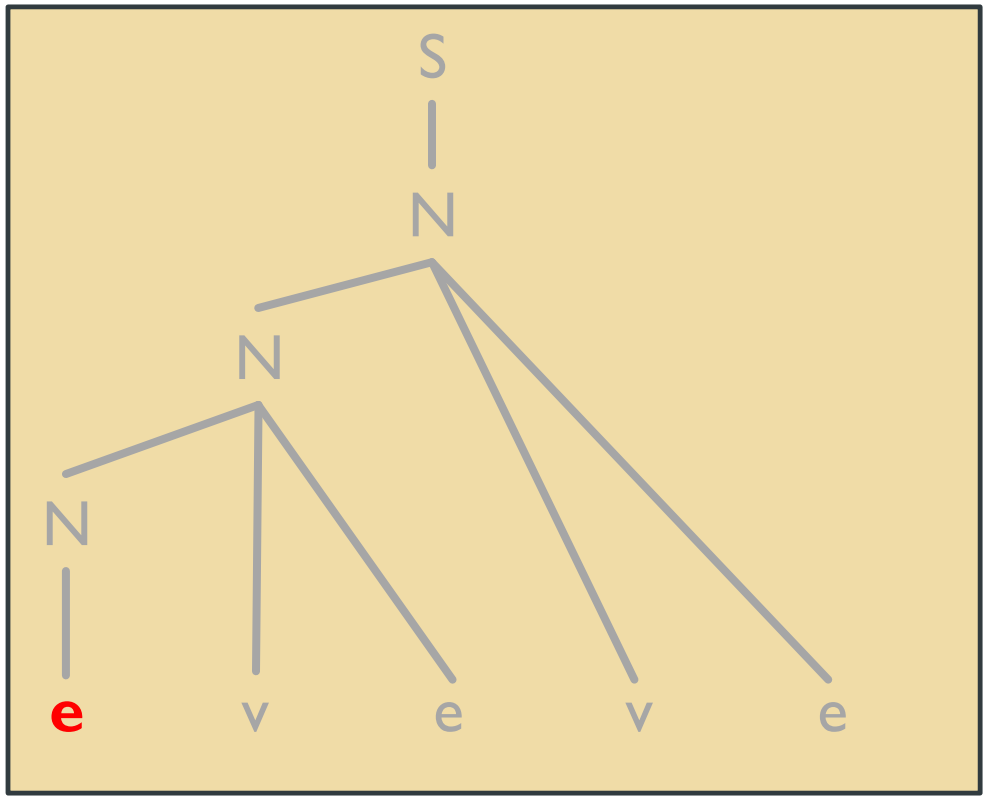
Redukció:
A szabályjobboldal helyettesítése szabálybaloldallal a veremben, közben a szintaxisfa bővítése.

AZ LR ELEMZÉSEK ALAPMŰVELETEI: LÉPTETÉS, REDUKCIÓ

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme:

e



Az elemző verme:
A szimbólumokat ebben gyűjtjük egészen addig, amíg a megfelelő szabályjobboldal megjelenik benne.

Léptetés:
A következő token elhelyezése a verem tetején.

Redukció:
A szabályjobboldal helyettesítése szabálybaloldallal a veremben, közben a szintaxisfa bővítése.

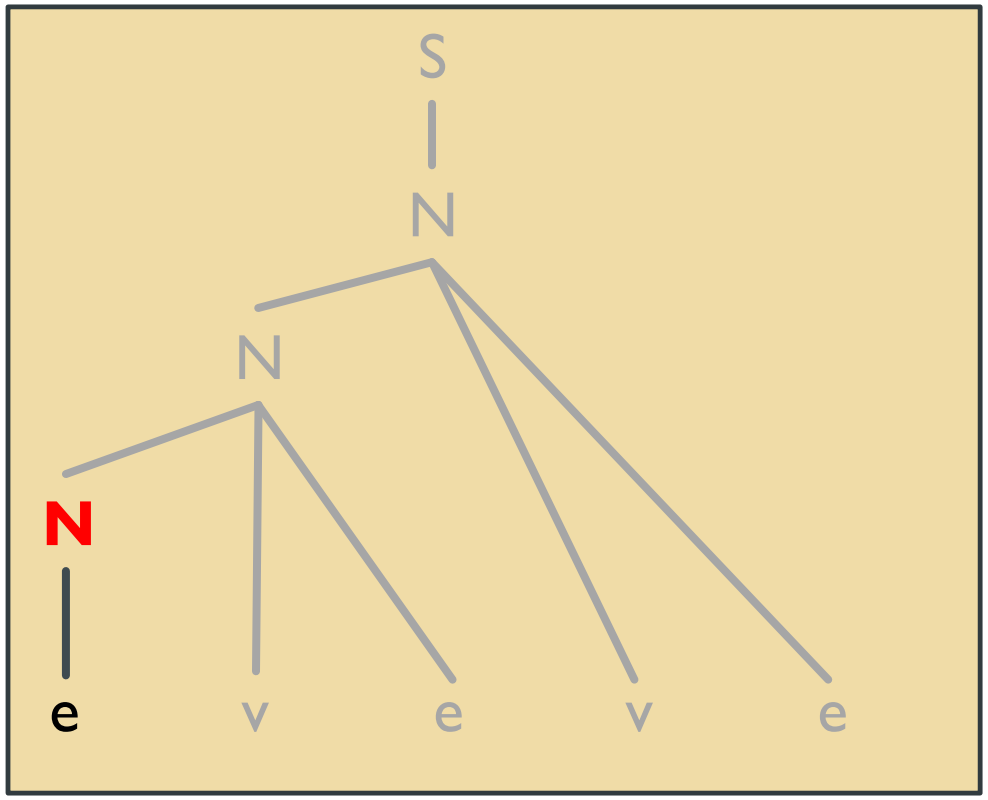
AZ LR ELEMZÉSEK ALAPMŰVELETEI: LÉPTETÉS, REDUKCIÓ

$S \rightarrow \varepsilon \mid N$

$N \rightarrow e \mid Nve$

Az elemző verme:

N



Az elemző verme:
A szimbólumokat ebben gyűjtjük egészen addig, amíg a megfelelő szabályjobboldal megjelenik benne.

Léptetés:
A következő token elhelyezése a verem tetején.

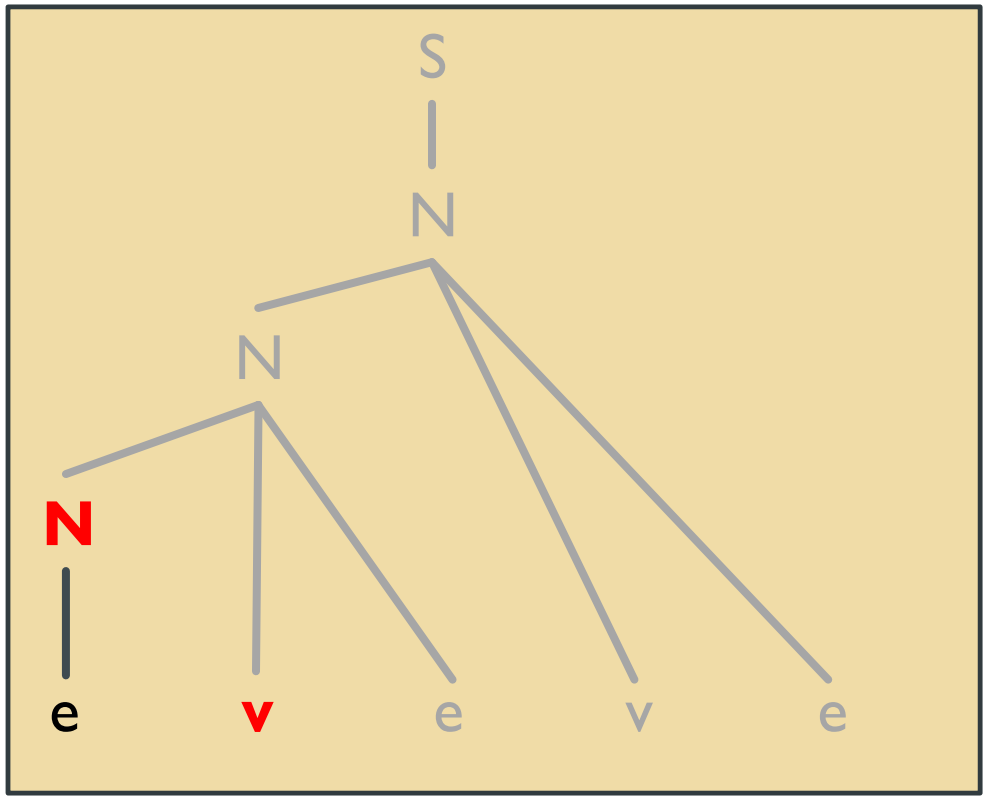
Redukció:
A szabályjobboldal helyettesítése szabálybaloldallal a veremben, közben a szintaxisfa bővítése.

AZ LR ELEMZÉSEK ALAPMŰVELETEI: LÉPTETÉS, REDUKCIÓ

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme:

Nv



Az elemző verme:
A szimbólumokat ebben gyűjtjük egészen addig, amíg a megfelelő szabályjobboldal megjelenik benne.

Léptetés:
A következő token elhelyezése a verem tetején.

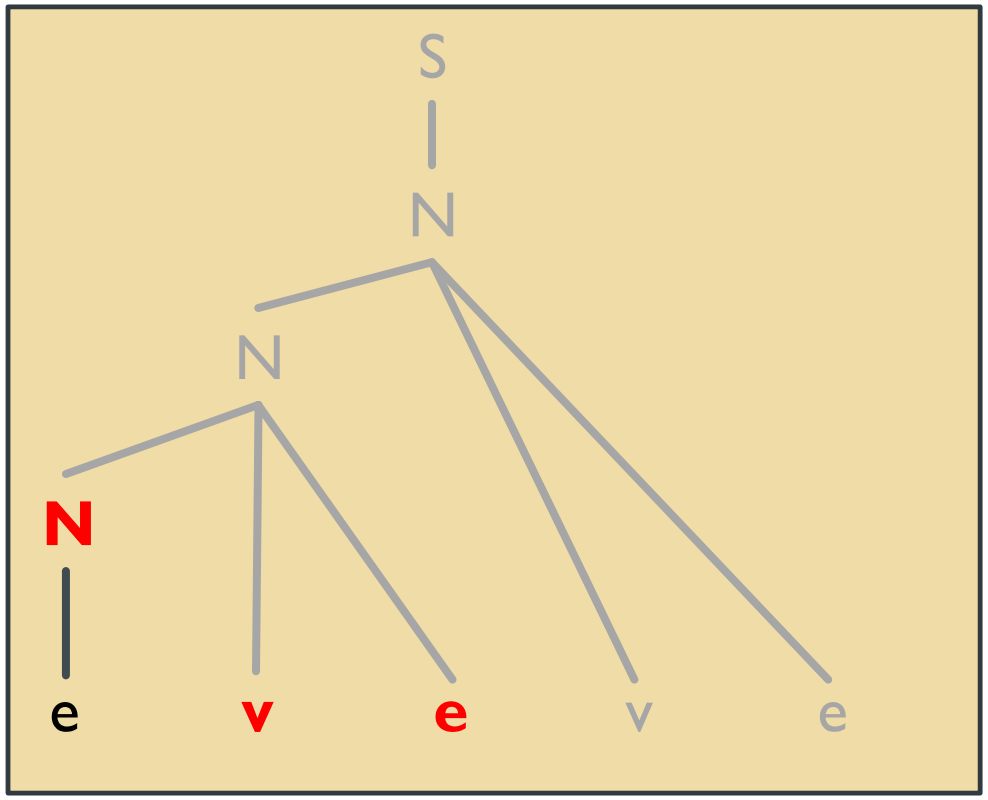
Redukció:
A szabályjobboldal helyettesítése szabálybaloldallal a veremben, közben a szintaxisfa bővítése.

AZ LR ELEMZÉSEK ALAPMŰVELETEI: LÉPTETÉS, REDUKCIÓ

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme:

Nve



Az elemző verme:
A szimbólumokat ebben gyűjtjük egészen addig, amíg a megfelelő szabályjobboldal megjelenik benne.

Léptetés:
A következő token elhelyezése a verem tetején.

Redukció:
A szabályjobboldal helyettesítése szabálybaloldallal a veremben, közben a szintaxisfa bővítése.

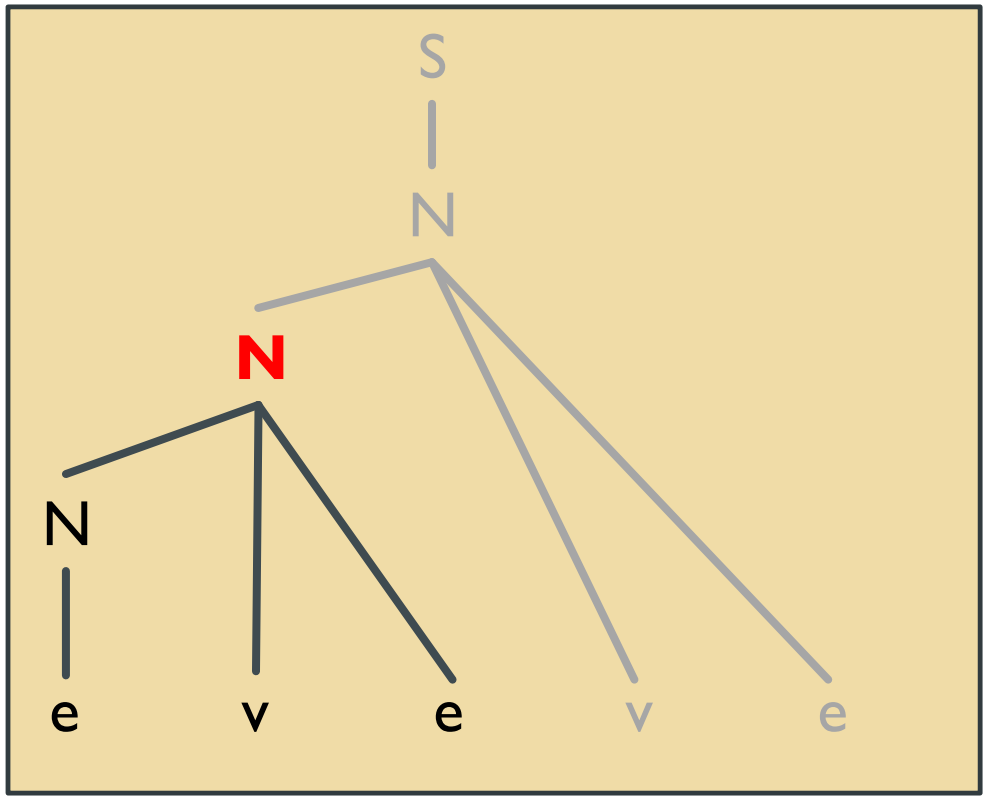
AZ LR ELEMZÉSEK ALAPMŰVELETEI: LÉPTETÉS, REDUKCIÓ

$S \rightarrow \varepsilon \mid N$

$N \rightarrow e \mid Nve$

Az elemző verme:

N



Az elemző verme:
A szimbólumokat ebben gyűjtjük egészen addig, amíg a megfelelő szabályjobboldal megjelenik benne.

Léptetés:
A következő token elhelyezése a verem tetején.

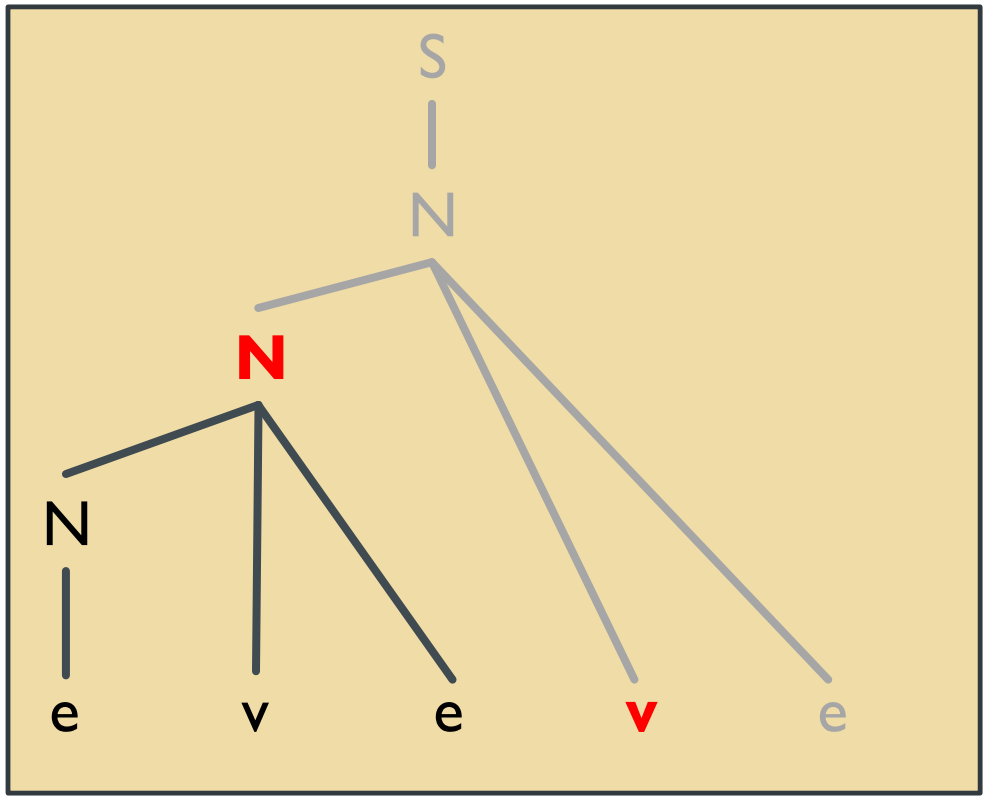
Redukció:
A szabályjobboldal helyettesítése szabálybaloldallal a veremben, közben a szintaxisfa bővítése.

AZ LR ELEMZÉSEK ALAPMŰVELETEI: LÉPTETÉS, REDUKCIÓ

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme:

Nv



Az elemző verme:
A szimbólumokat ebben gyűjtjük egészen addig, amíg a megfelelő szabályjobboldal megjelenik benne.

Léptetés:
A következő token elhelyezése a verem tetején.

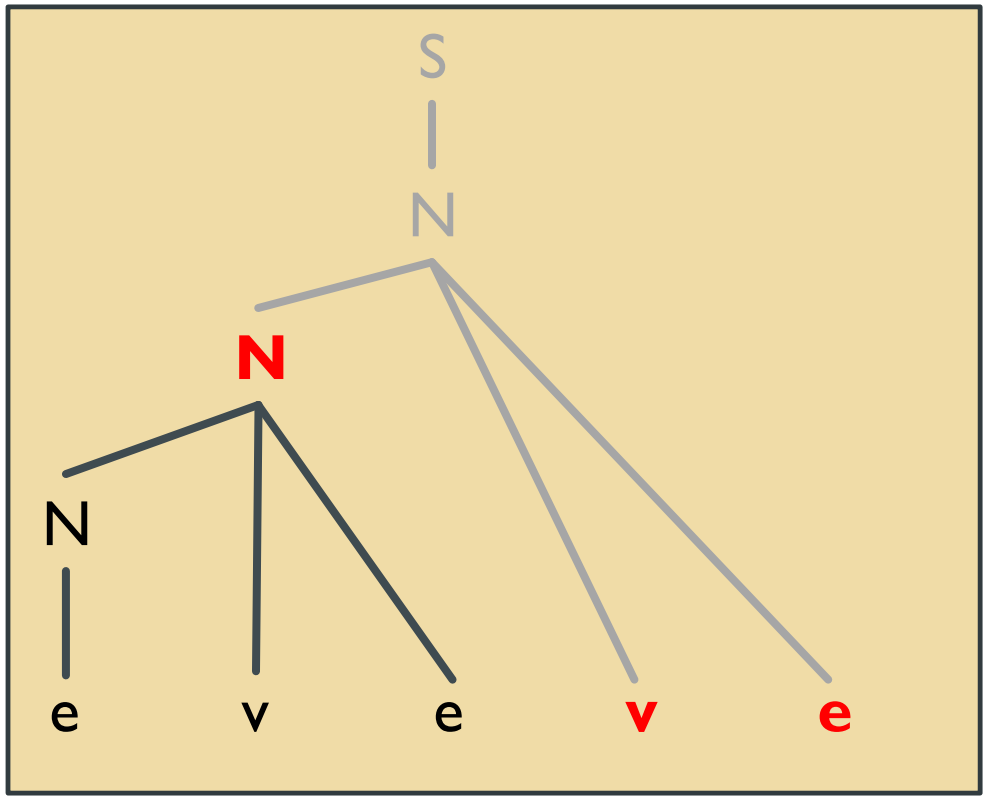
Redukció:
A szabályjobboldal helyettesítése szabálybaloldallal a veremben, közben a szintaxisfa bővítése.

AZ LR ELEMZÉSEK ALAPMŰVELETEI: LÉPTETÉS, REDUKCIÓ

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme:

Nve



Az elemző verme:
A szimbólumokat ebben gyűjtjük egészen addig, amíg a megfelelő szabályjobboldal megjelenik benne.

Léptetés:
A következő token elhelyezése a verem tetején.

Redukció:
A szabályjobboldal helyettesítése szabálybaloldallal a veremben, közben a szintaxisfa bővítése.

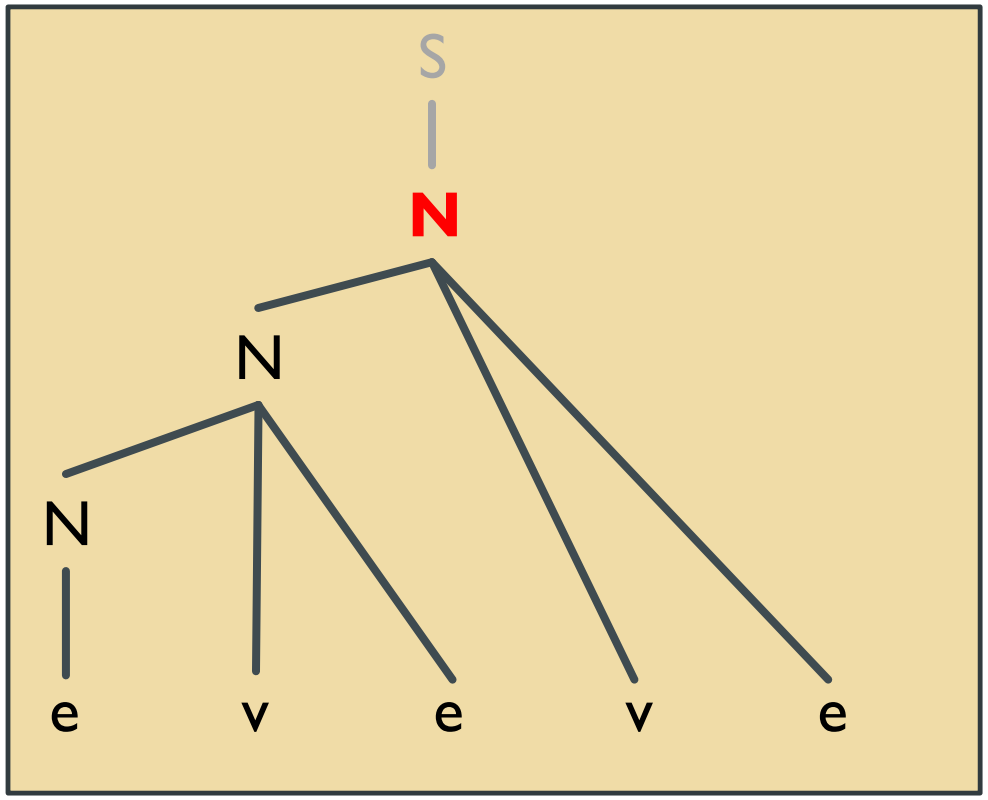
AZ LR ELEMZÉSEK ALAPMŰVELETEI: LÉPTETÉS, REDUKCIÓ

$S \rightarrow \varepsilon \mid N$

$N \rightarrow e \mid Nve$

Az elemző verme:

N



Az elemző verme:
A szimbólumokat ebben gyűjtjük egészen addig, amíg a megfelelő szabályjobboldal megjelenik benne.

Léptetés:
A következő token elhelyezése a verem tetején.

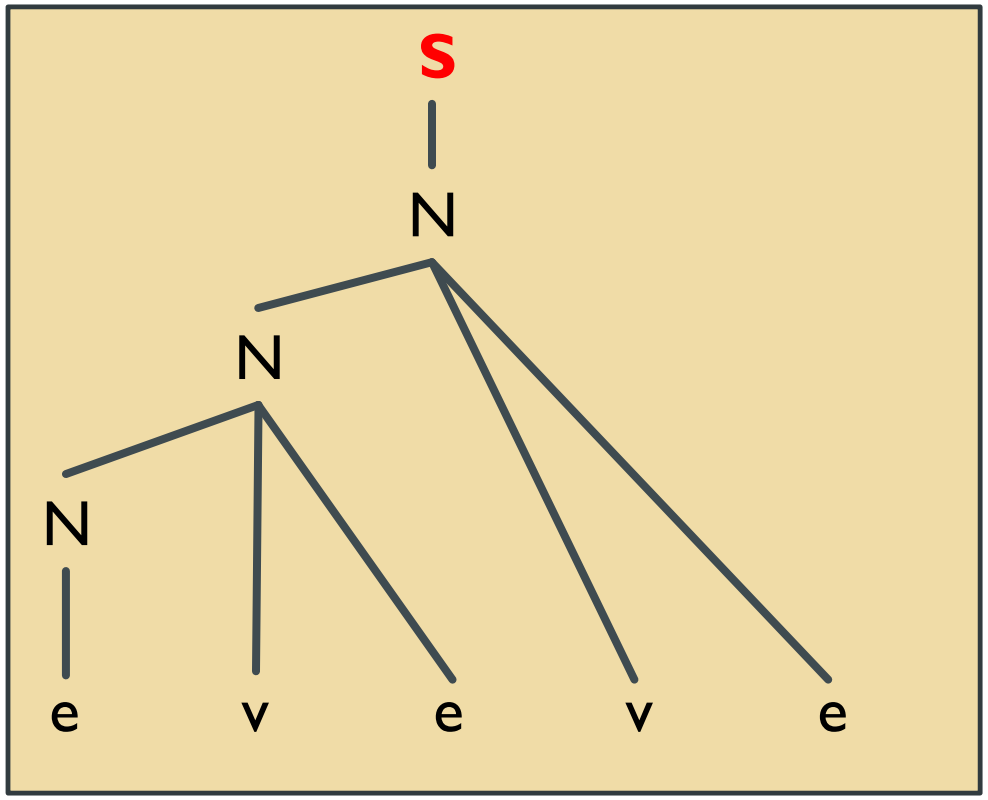
Redukció:
A szabályjobboldal helyettesítése szabálybaloldallal a veremben, közben a szintaxisfa bővítése.

AZ LR ELEMZÉSEK ALAPMŰVELETEI: LÉPTETÉS, REDUKCIÓ

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme:

S



Az elemző verme:
A szimbólumokat ebben gyűjtjük egészen addig, amíg a megfelelő szabályjobboldal megjelenik benne.

Léptetés:
A következő token elhelyezése a verem tetején.

Redukció:
A szabályjobboldal helyettesítése szabálybaloldallal a veremben, közben a szintaxisfa bővítése.

LÉPTETÉS VAGY REDUKCIÓ?

- Az elemzés következő lépésének meghatározásához figyelembe vesszük:
  - A következő *valahány* tokent: **előreolvasás**;
  - Az **elemző állapotát**, ami a verembe bekerülő szimbólumoktól függően változik.

LR(K) NYELVTANOK

Definíció: LR(k) nyelvtan

Egy környezetfüggetlen nyelvtan *LR(k)* tulajdonságú valamely $k \in \mathbb{N}_0$ számra, ha az elemzés pillanatnyi állapotából és legfeljebb *k* token előreolvasásával egyértelműen meghatározható, hogy léptetés vagy redukció következik, és redukció esetén az alkalmazandó nyelvtani szabály is kiderül.

- LR jelentése: Left to right, using Rightmost derivation
  (Balról jobbra legjobboldalibb levezetéssel)
- Az LR(1) elemzéssel foglalkozunk...

LÉPTETÉS VAGY REDUKCIÓ?

- Az LR(1) elemzés következő lépésének meghatározásához figyelembe vesszük:
  - A következő tokent: **előreolvasás**;
  - Az **elemző állapotát**, ami a verembe bekerülő szimbólumoktól függően változik.
- Ezek függvényében a következő lépést az **elemző táblázat** határozza meg:

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás	1
1	Hiba	Léptetés: 3	Elfogadás	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba

AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

0

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba

AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

0

	e	v	Input vége	N
0	**Léptetés: 2**	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba

e

AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

	e	v	Input vége	N
0	**Léptetés: 2**	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba

AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

0	**2**
	e

	e	**v**	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	**Redukció: N→e**	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba

e **v**

AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

0	2
	e

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	**Redukció: N→e**	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba

e v

AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

0

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	**Redukció: N→e**	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba

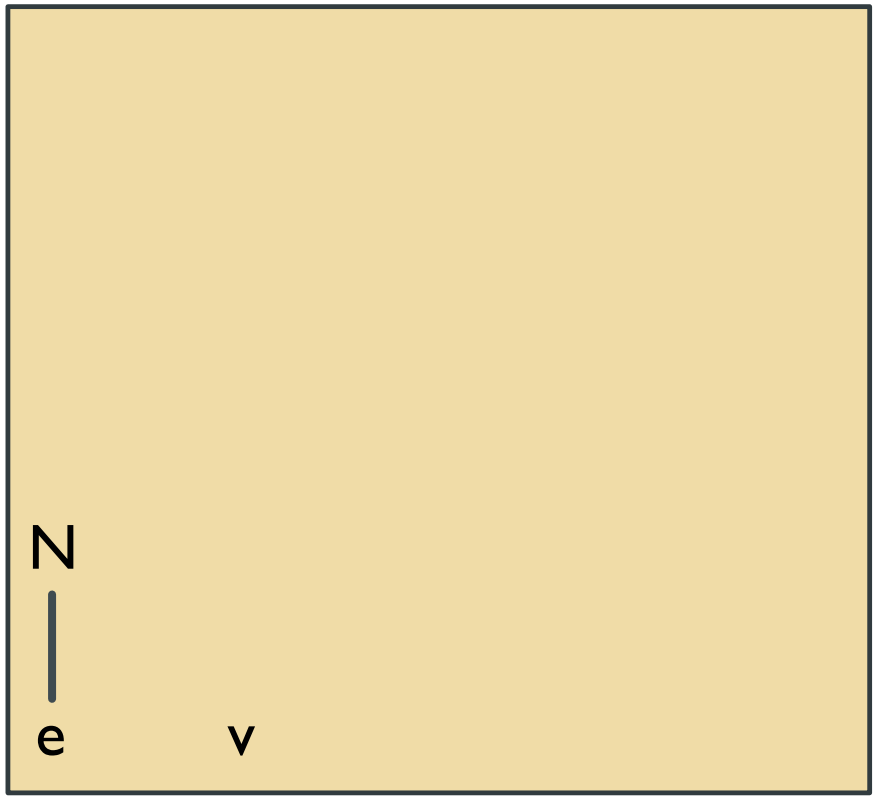
e v

AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme az állapotot is tárolja:

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	**1**
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	**Redukció: N→e**	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba

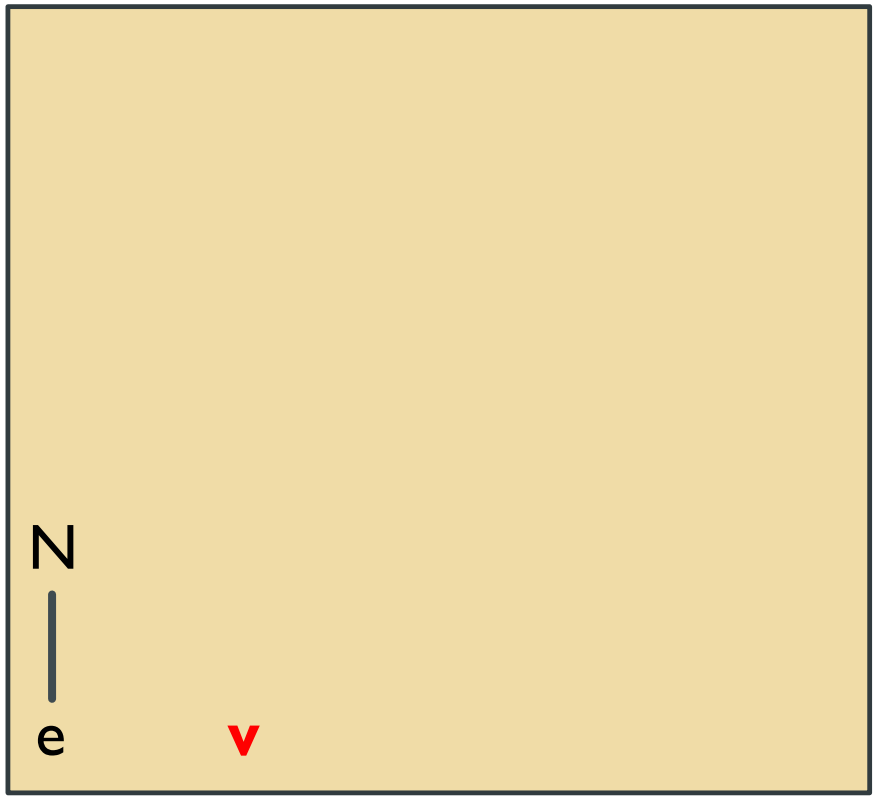


AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

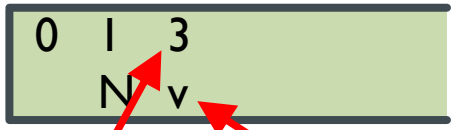
	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	I
I	Hiba	**Léptetés: 3**	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba



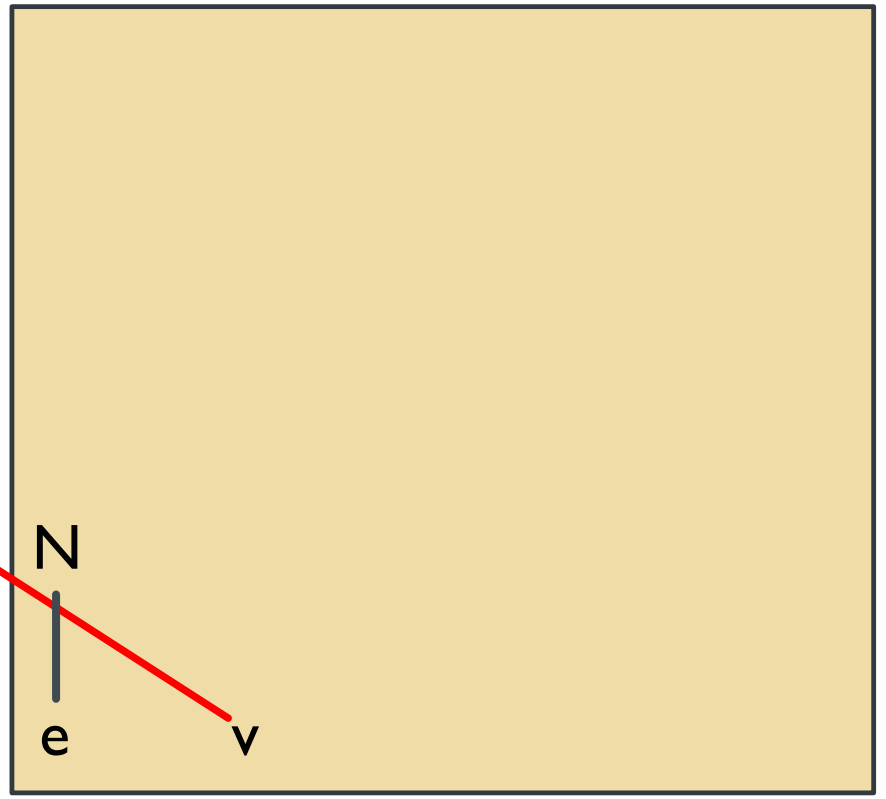
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:



	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	**Léptetés: 3**	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba



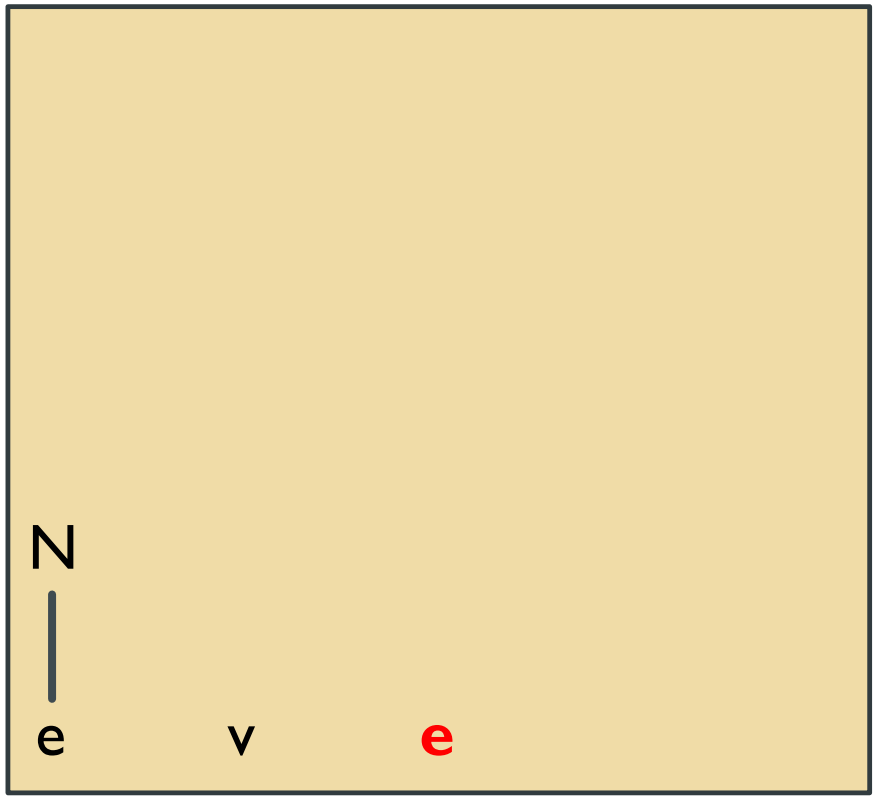
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme az állapotot is tárolja:

0	1	3
	N	v

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	**Léptetés: 4**	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba



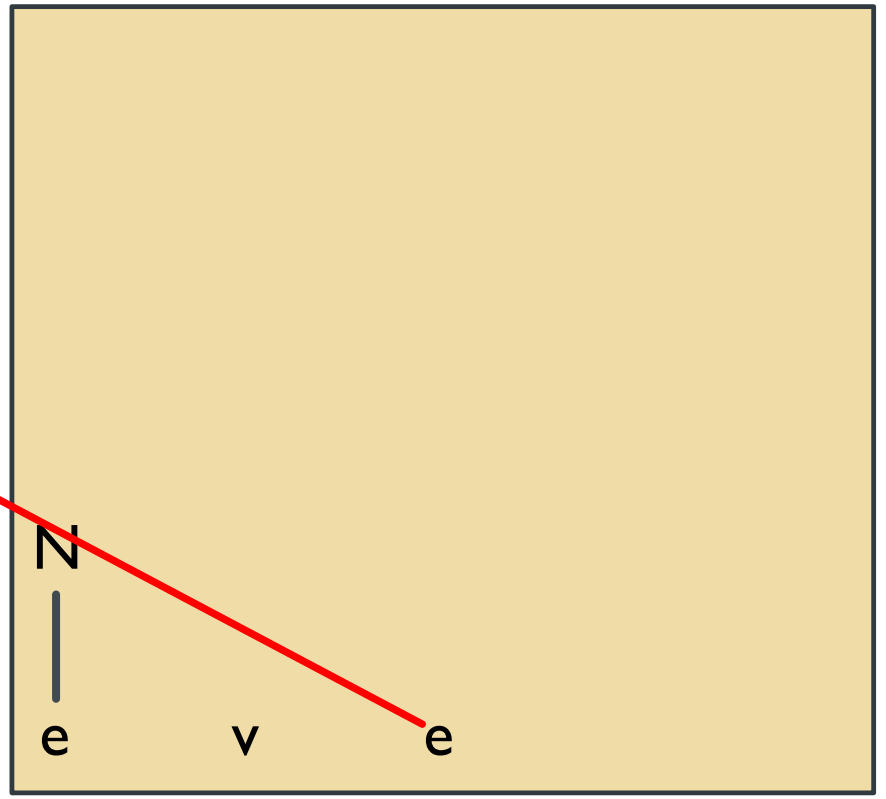
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme az állapotot is tárolja:

0	1	3	4
	N	v	e

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	**Léptetés: 4**	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba



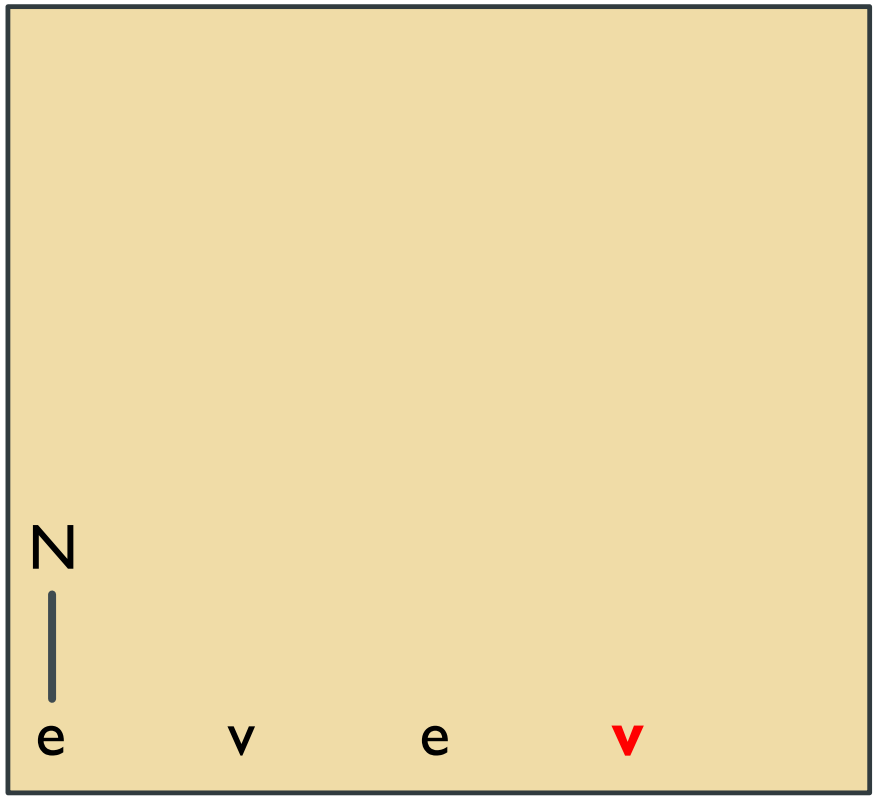
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme az állapotot is tárolja:

0	1	3	4
	N	v	e

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	**Redukció: N→Nve**	Redukció: N→Nve	Hiba



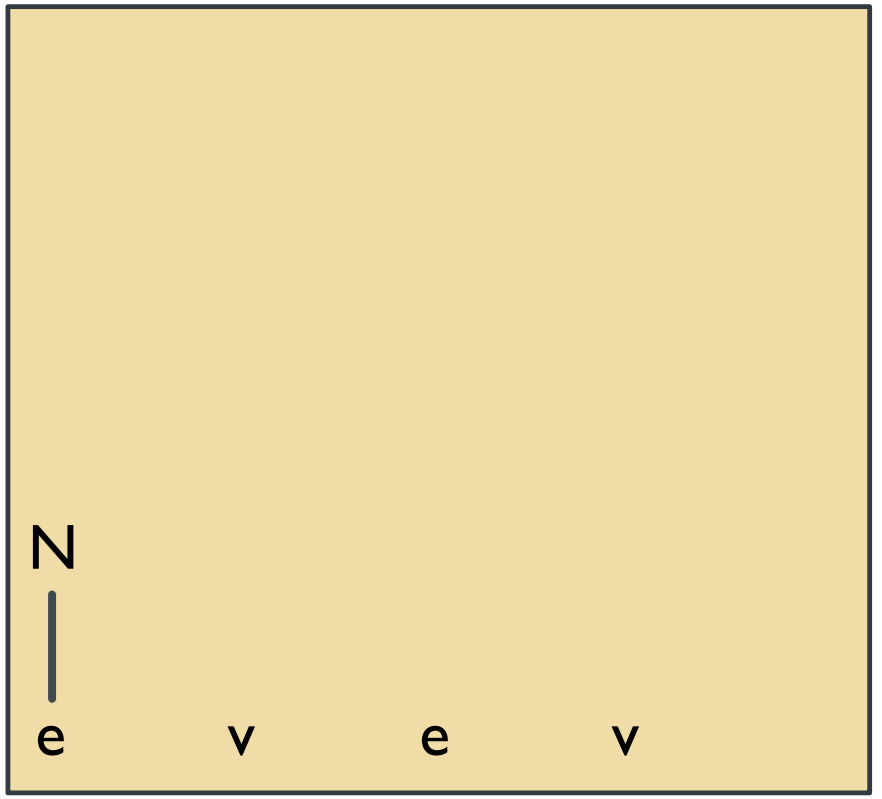
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

0	1	3	4
	N	v	e

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	**Redukció: N→Nve**	Redukció: N→Nve	Hiba



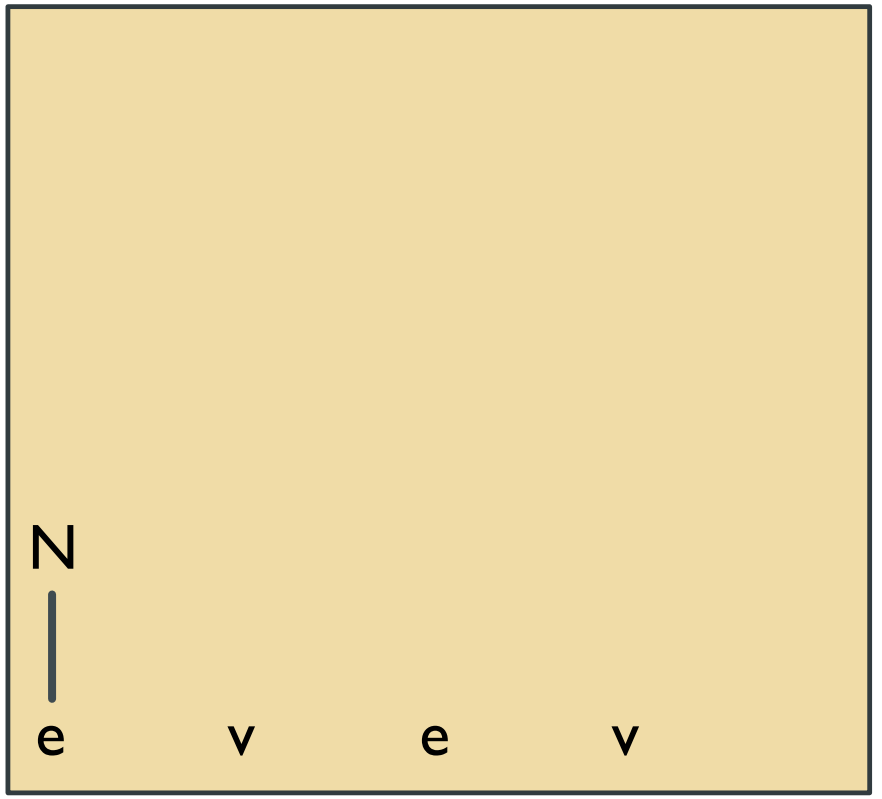
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

0

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	**Redukció: N→Nve**	Redukció: N→Nve	Hiba

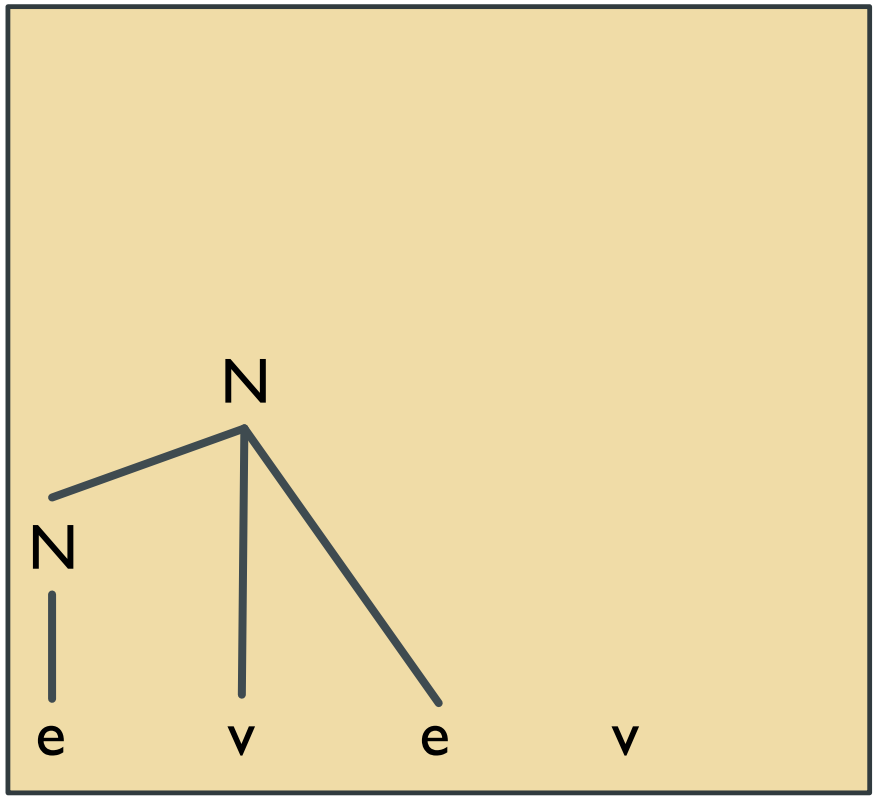


AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme az állapotot is tárolja:

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	**1**
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	**Redukció: N→Nve**	Redukció: N→Nve	Hiba



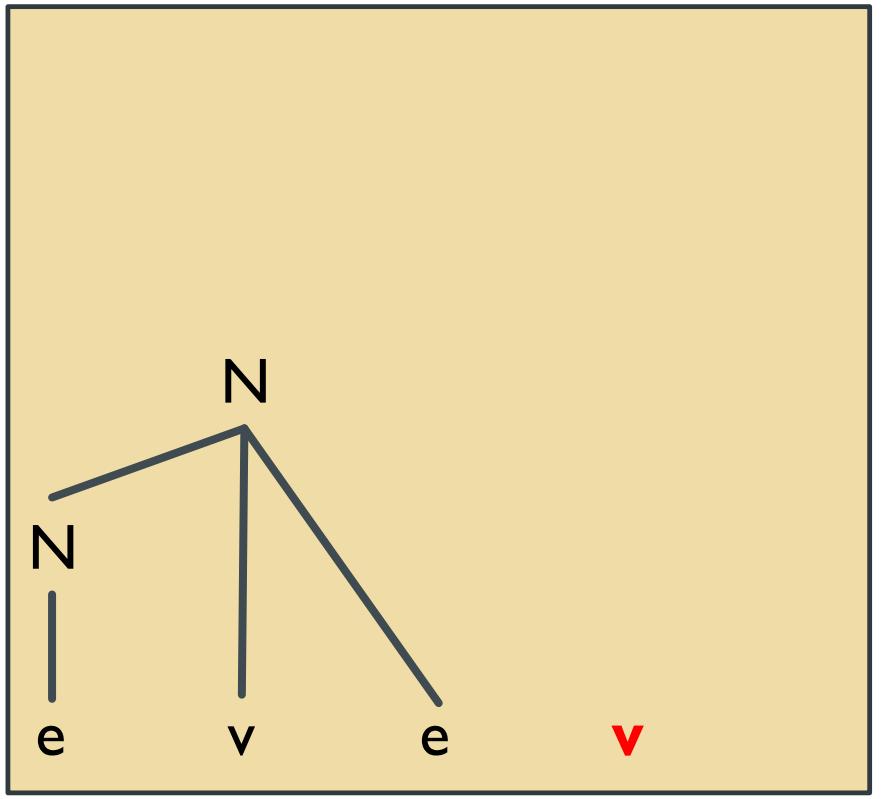
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme az állapotot is tárolja:

0	I
	N

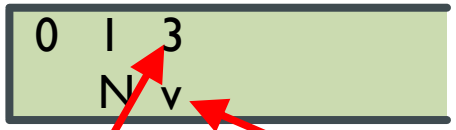
	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	I
I	Hiba	**Léptetés: 3**	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba



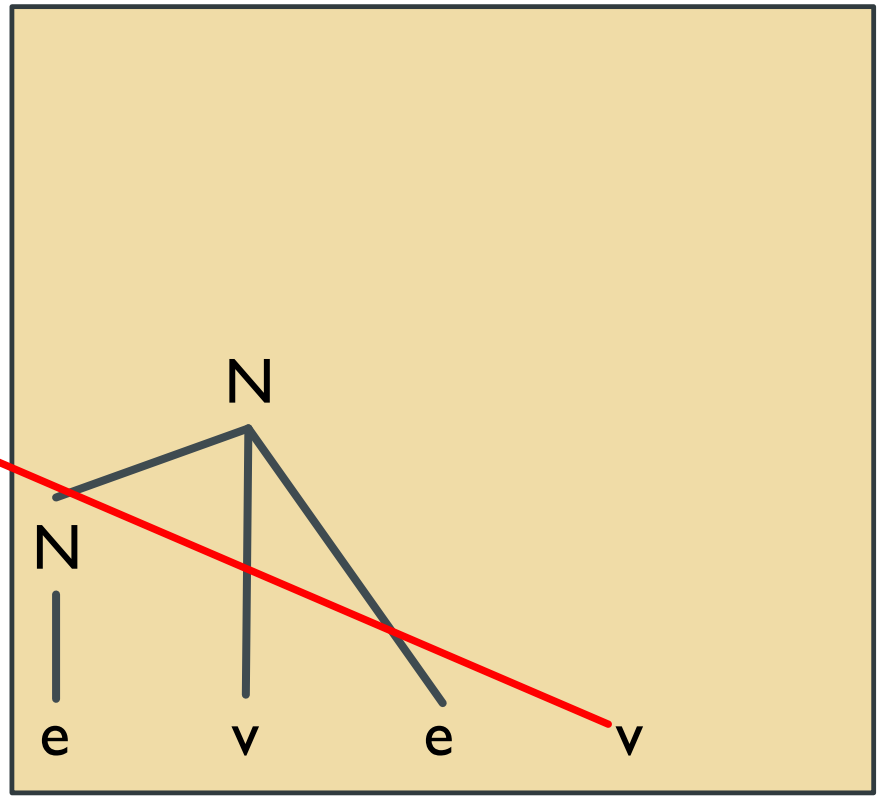
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:



	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	**Léptetés: 3**	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba



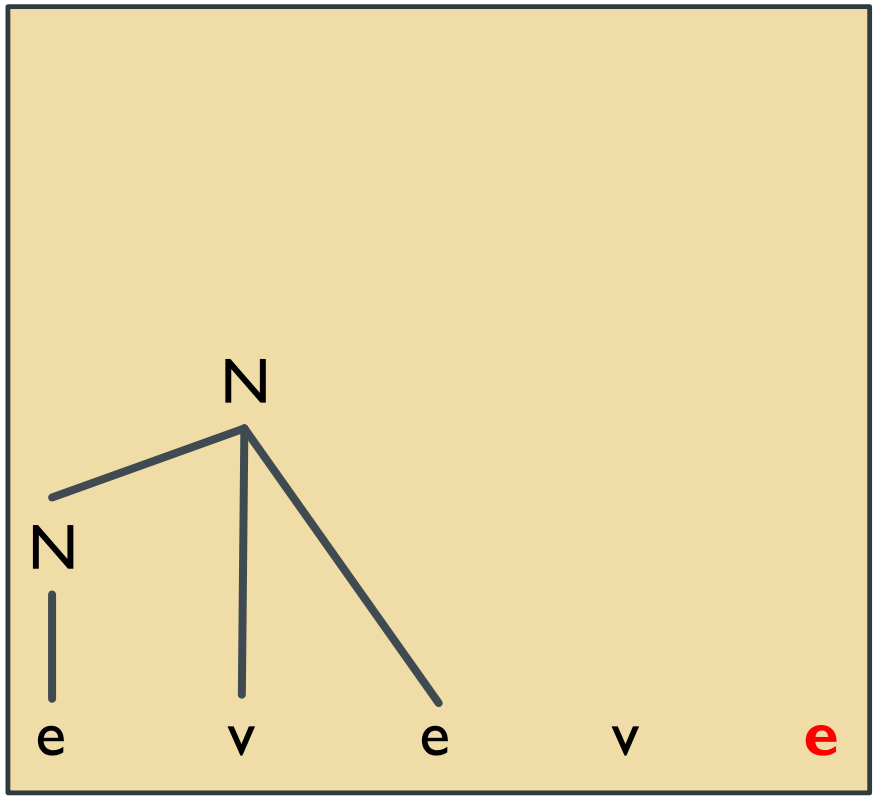
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme az állapotot is tárolja:

0	1	3
	N	v

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	**Léptetés: 4**	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba



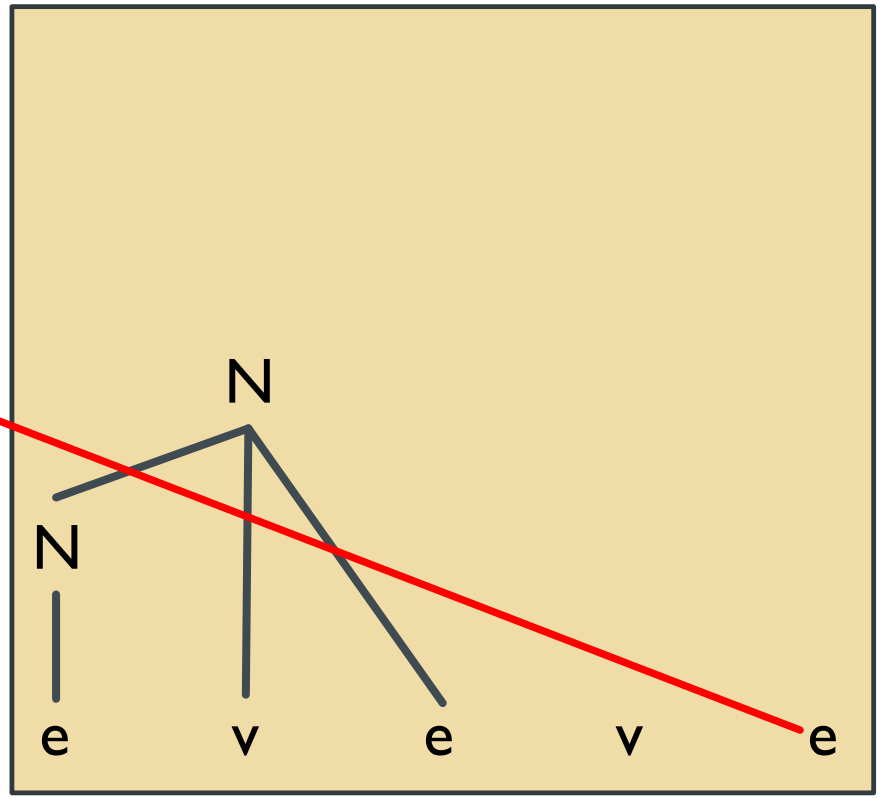
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

0	1	3	4
	N	v	e

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	**Léptetés: 4**	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba



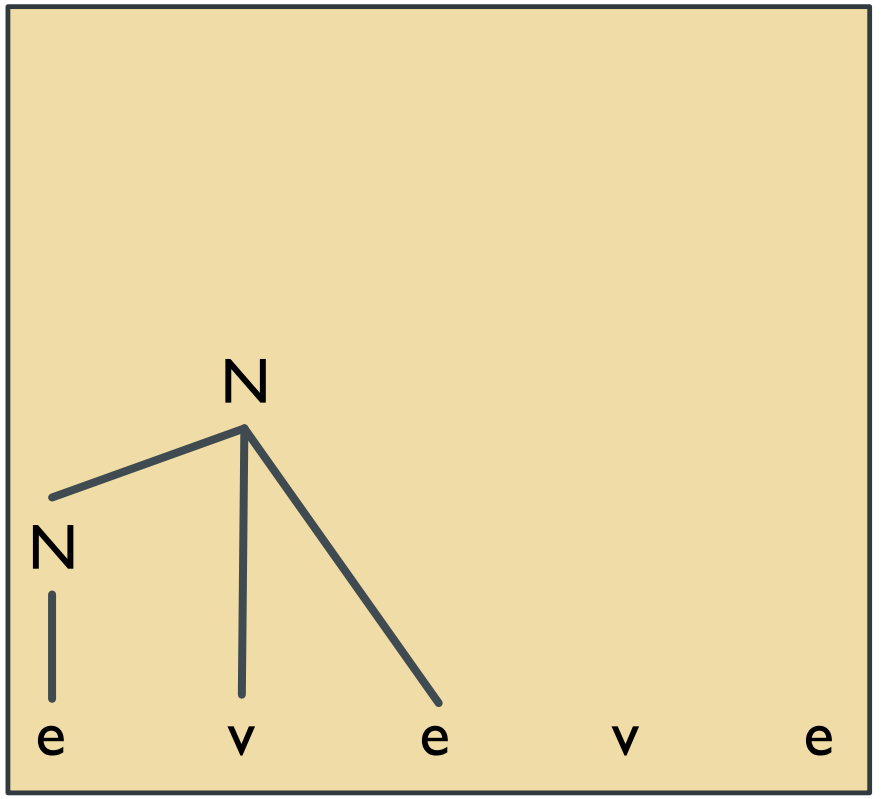
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme az állapotot is tárolja:

0	1	3	4
	N	v	e

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	**Redukció: N→Nve**	Hiba



AZ LR(1) ELEMZŐ MŰKÖDÉSE

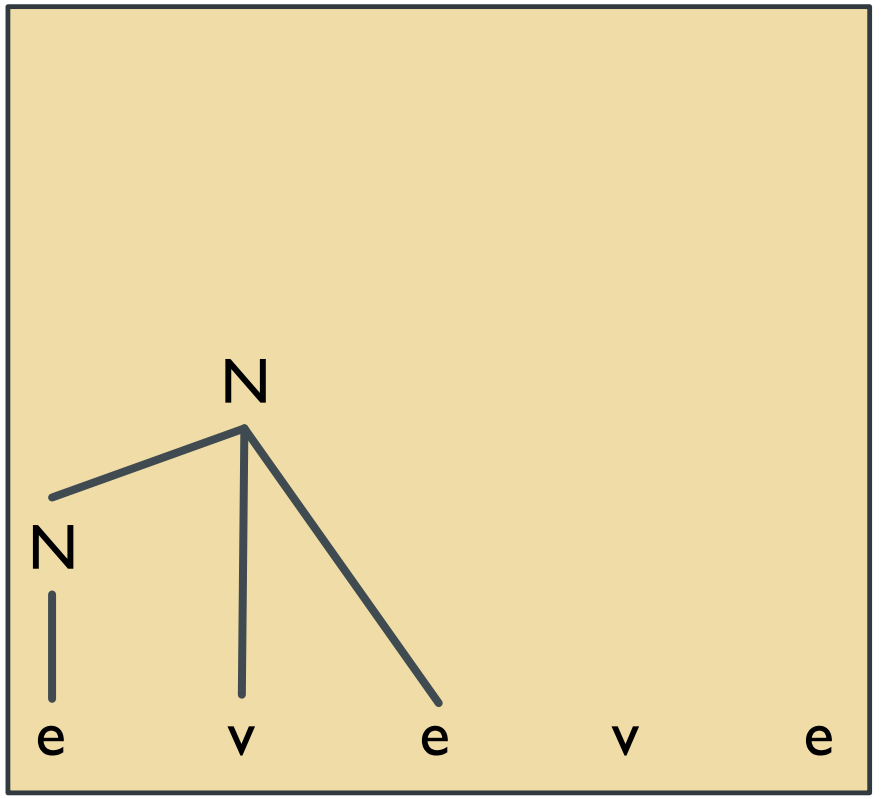
$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

0	1	3	4
	N	v	e

× × ×

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	**Redukció: N→Nve**	Hiba



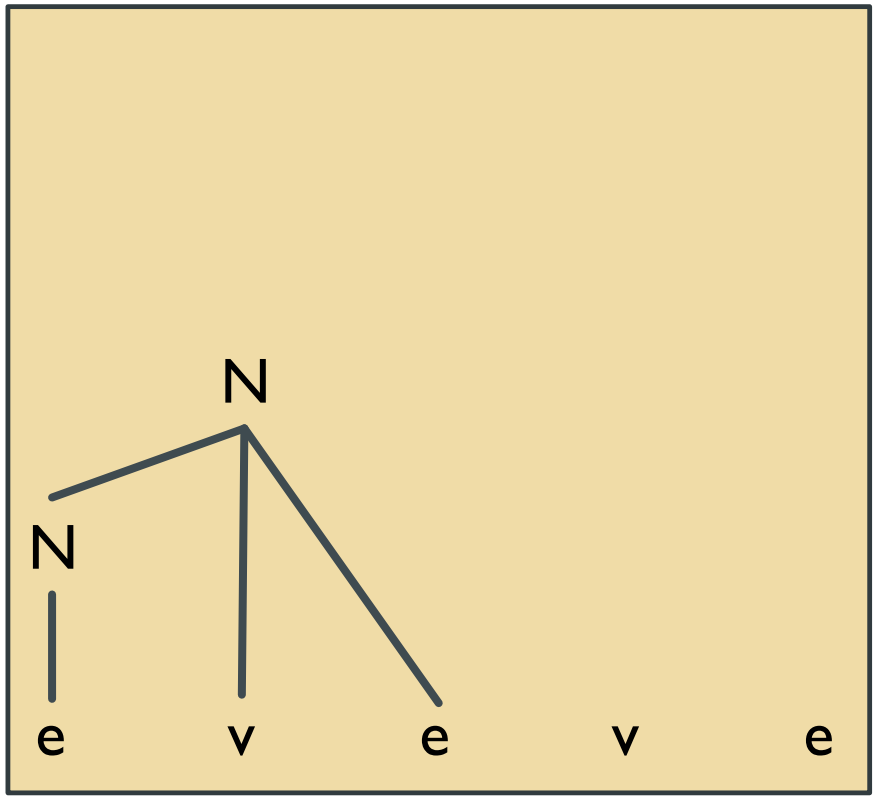
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

0

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	**Redukció: N→Nve**	Hiba

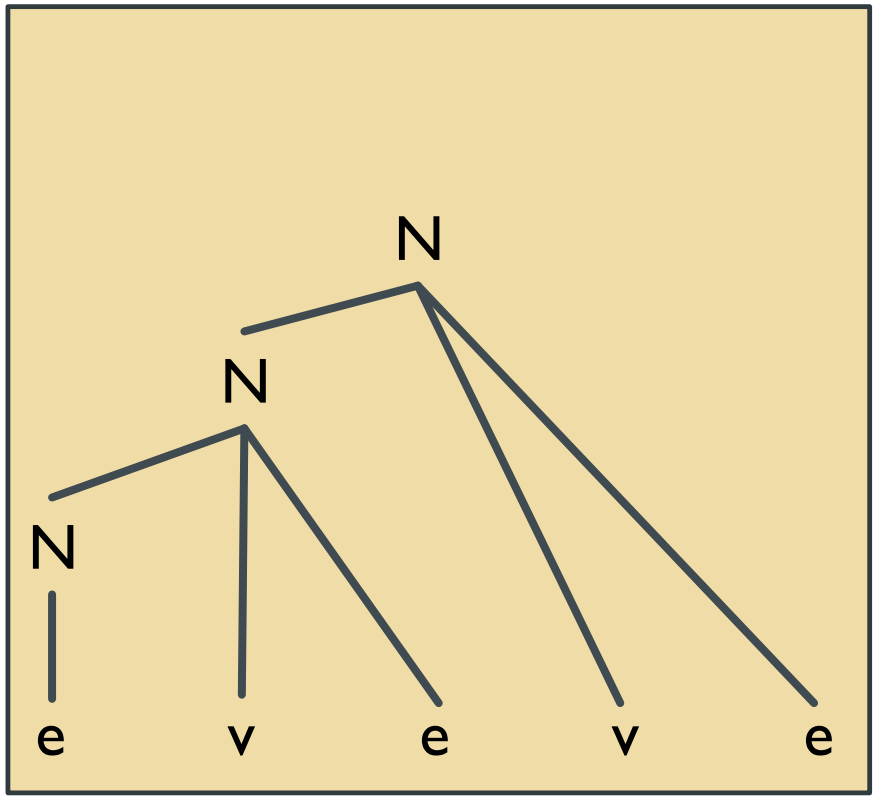


AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme az állapotot is tárolja:

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	**I**
I	Hiba	Léptetés: 3	Elfogadás: S→N	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	**Redukció: N→Nve**	Hiba



input vége

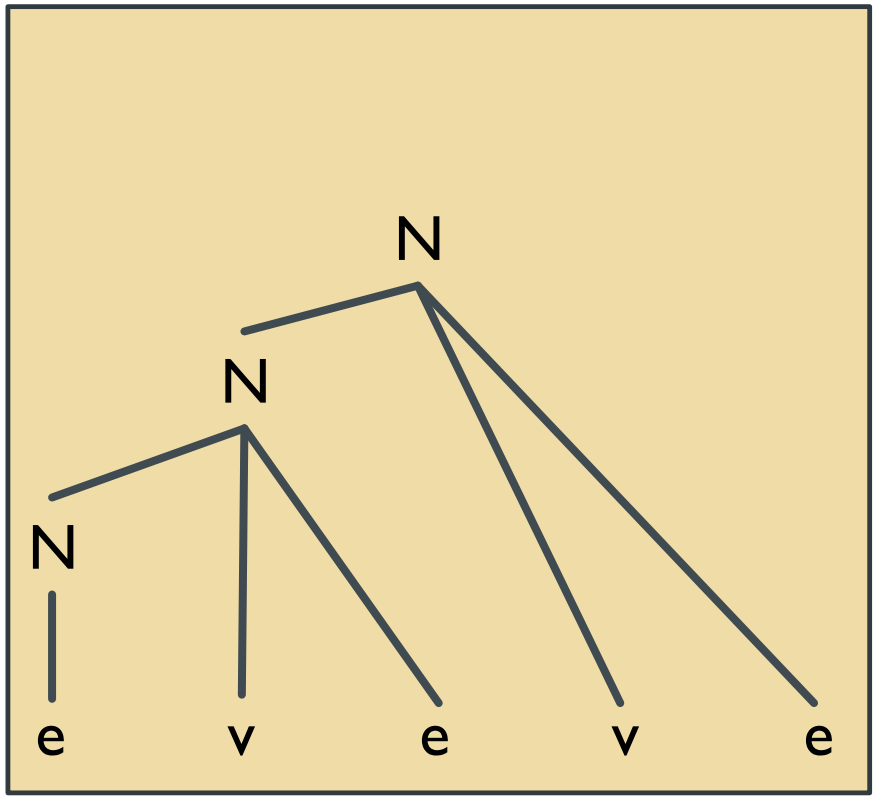
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme az állapotot is tárolja:

0	I
	N

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	I
I	Hiba	Léptetés: 3	**Elfogadás: S→N**	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba



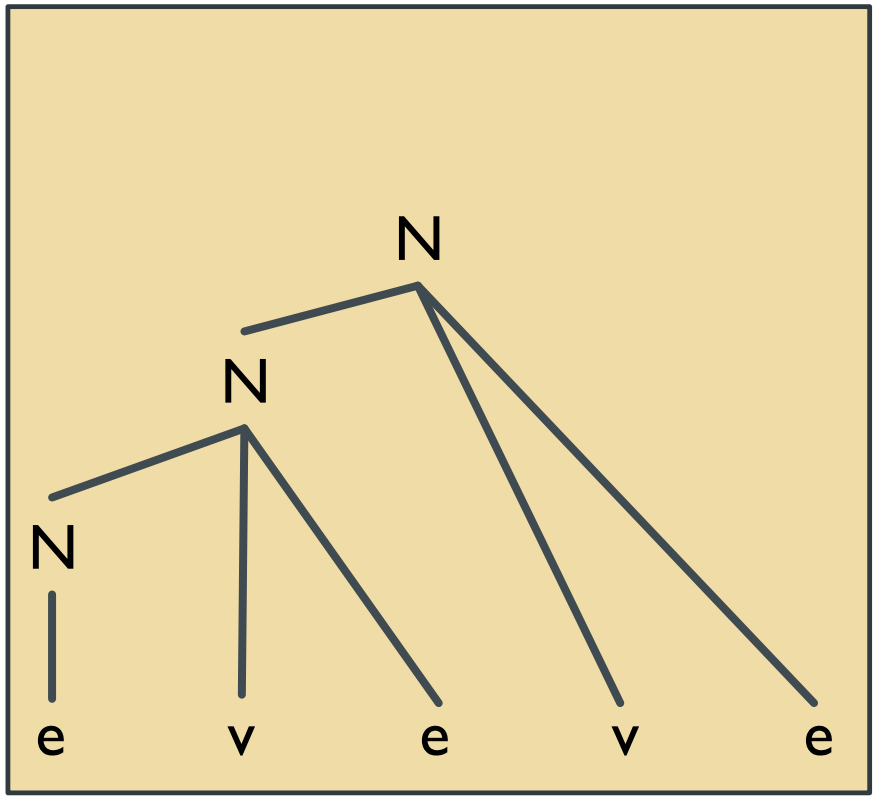
input vége

AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	I
I	Hiba	Léptetés: 3	**Elfogadás: S→N**	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba



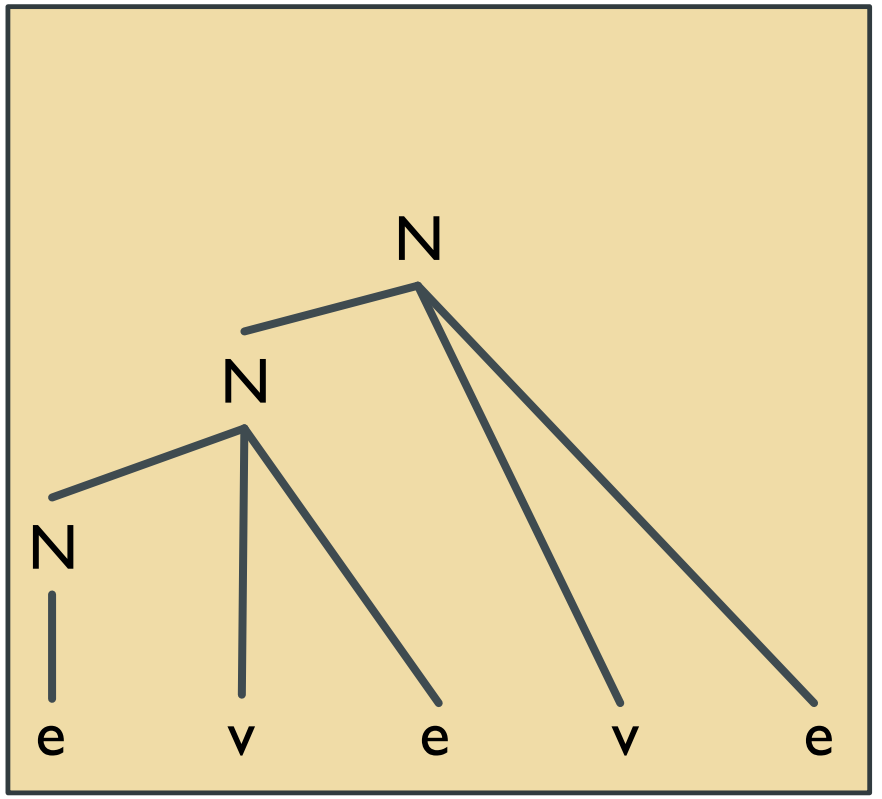
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme
az állapotot is tárolja:

0

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	**Elfogadás: S→N**	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba



input vége

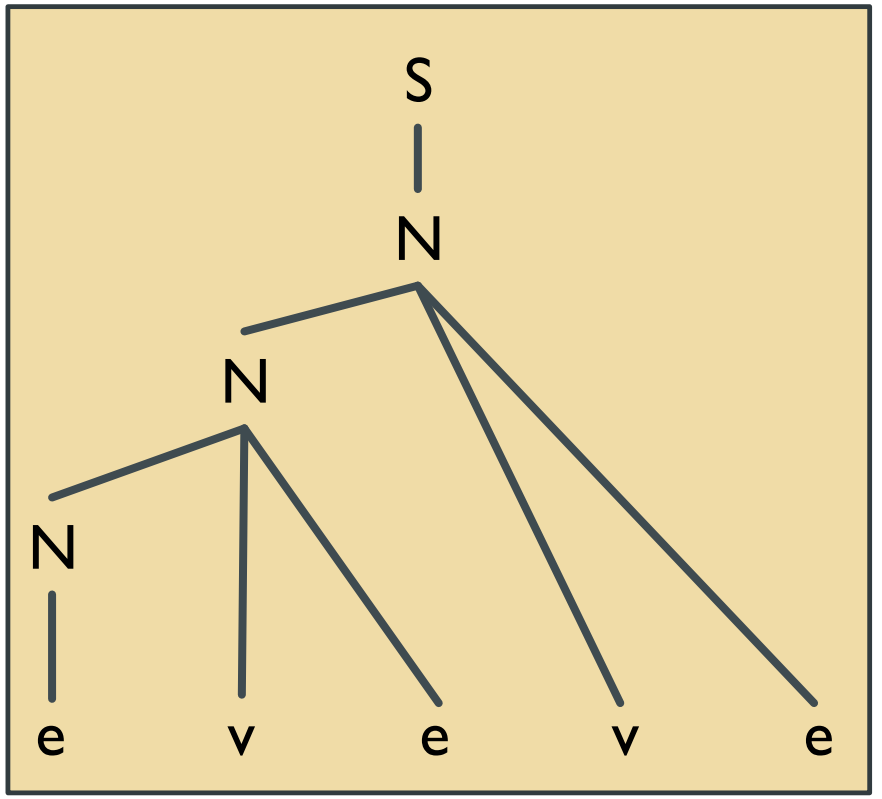
AZ LR(1) ELEMZŐ MŰKÖDÉSE

$S \rightarrow \varepsilon \mid N$
$N \rightarrow e \mid Nve$

Az elemző verme az állapotot is tárolja:

Input elfogadva

	e	v	Input vége	N
0	Léptetés: 2	Hiba	Elfogadás: S→ε	1
1	Hiba	Léptetés: 3	**Elfogadás: S→N**	Hiba
2	Hiba	Redukció: N→e	Redukció: N→e	Hiba
3	Léptetés: 4	Hiba	Hiba	Hiba
4	Hiba	Redukció: N→Nve	Redukció: N→Nve	Hiba



input vége

AZ ELEMZŐ TÁBLÁZAT LÉTREHOZÁSA

- A nyelvtanból algoritmikusan létrehozható
  - Bonyolult algoritmus, nem tananyag...

Tétel: LR(1) tulajdonság ellenőrzése

Egy környezetfüggetlen nyelvtan pontosan akkor LR(1) tulajdonságú, ha az elemző táblázatot kitöltő algoritmus konfliktusmentesen kitölti a táblázatot.

SZINTAKTIKUS ELEMZÉS A GYAKORLATBAN

- Felülről lefelé elemzéshez példaszoftver:
  ANTLR (Another Tool for Language Recognition)
  https://www.antlr.org/
  Az itt bemutatott LL elemzőhöz hasonlót (de okosabbat) generál.
- Alulról felfelé elemzéshez példaszoftver
  GNU Bison
  https://www.gnu.org/software/bison/
  Az itt bemutatott LR elemzőhöz hasonlót (de okosabbat) generál.
  A gyakorlatokon ezt az elemzőgenerátort használjuk.