Finite automata and formal languages (DIT322, TMV028)

Nils Anders Danielsson

2020-02-24

Today

- Context-free languages.
- Some equivalences.
- Ambiguity.
- Designing grammars.

Context-free languages

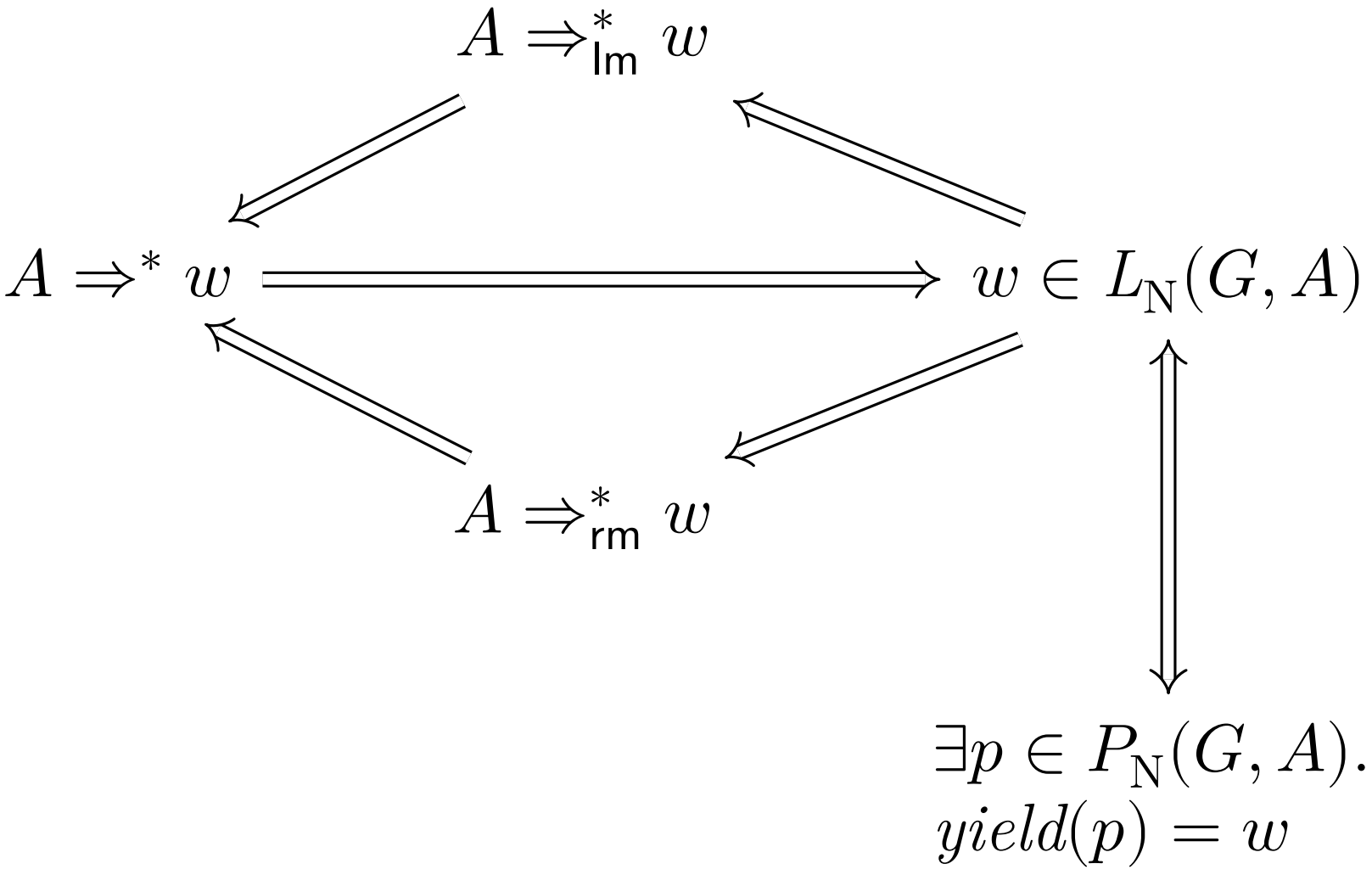
Context-free languages

A language $L \subseteq \Sigma^*$ is context-free if $L = L(G)$, where G is a context-free grammar with Σ as the set of terminals.

Some equivalences

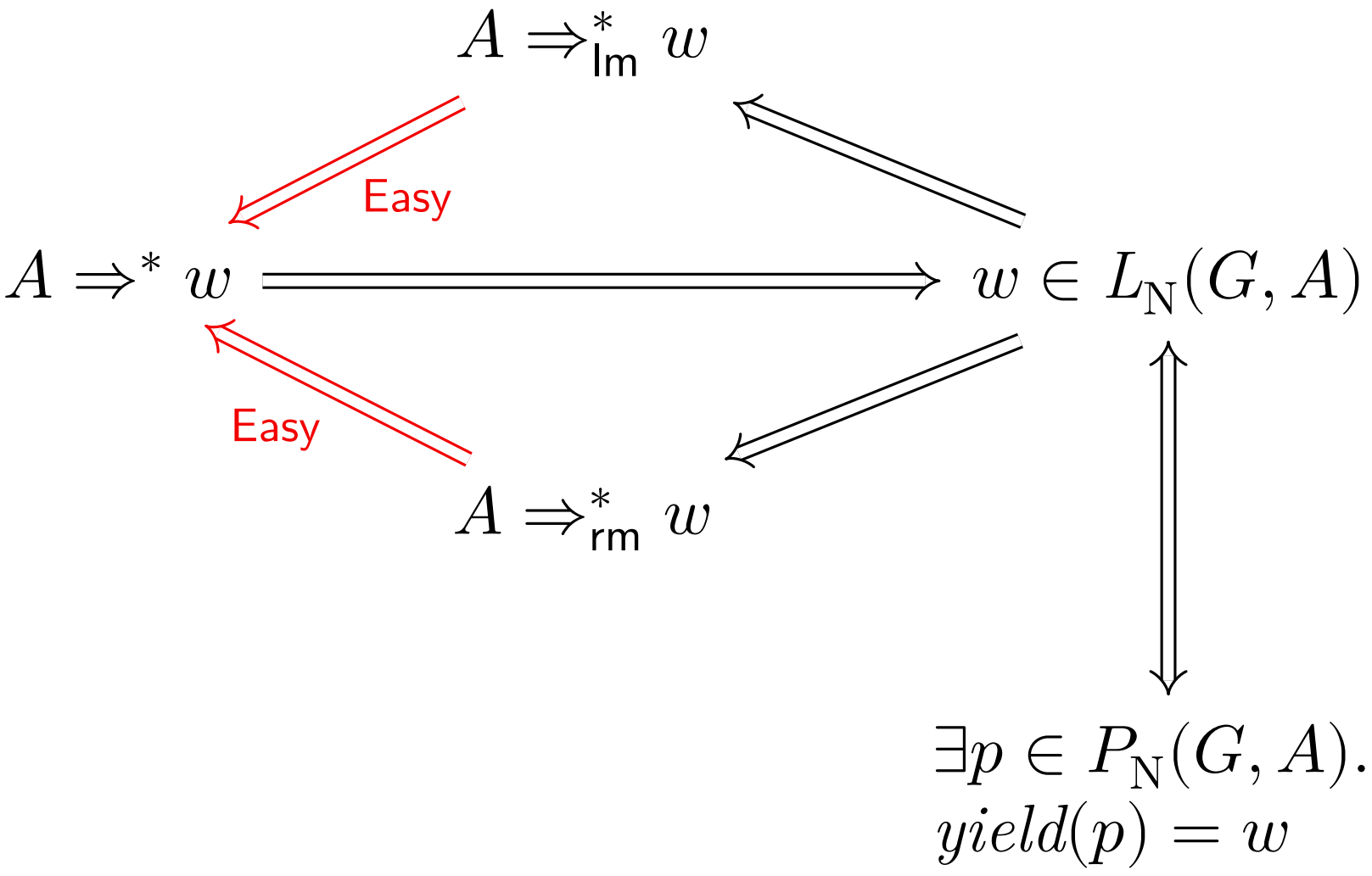
Some equivalences

With $w \in \Sigma^*$:



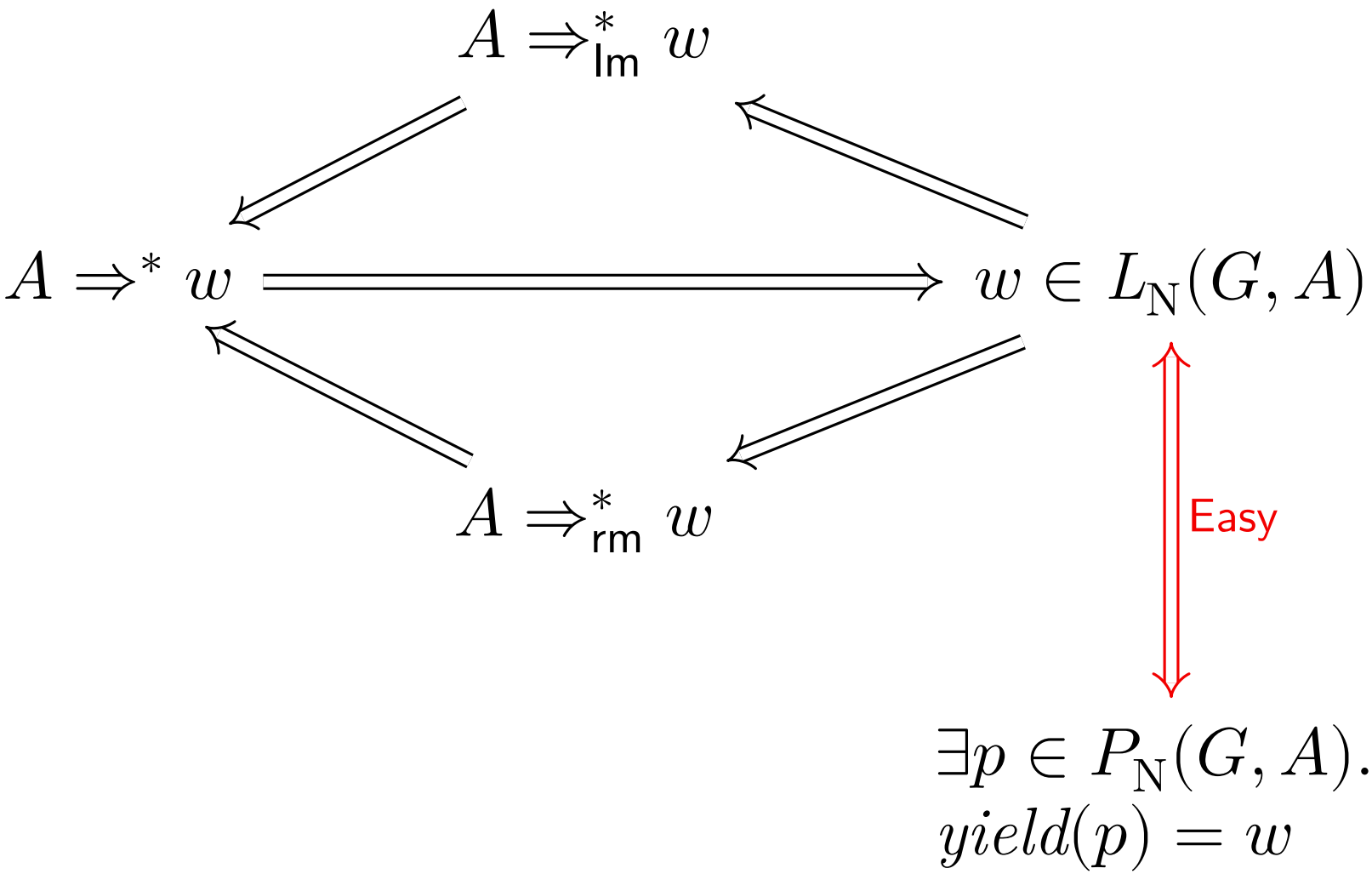
Some equivalences

With $w \in \Sigma^*$:



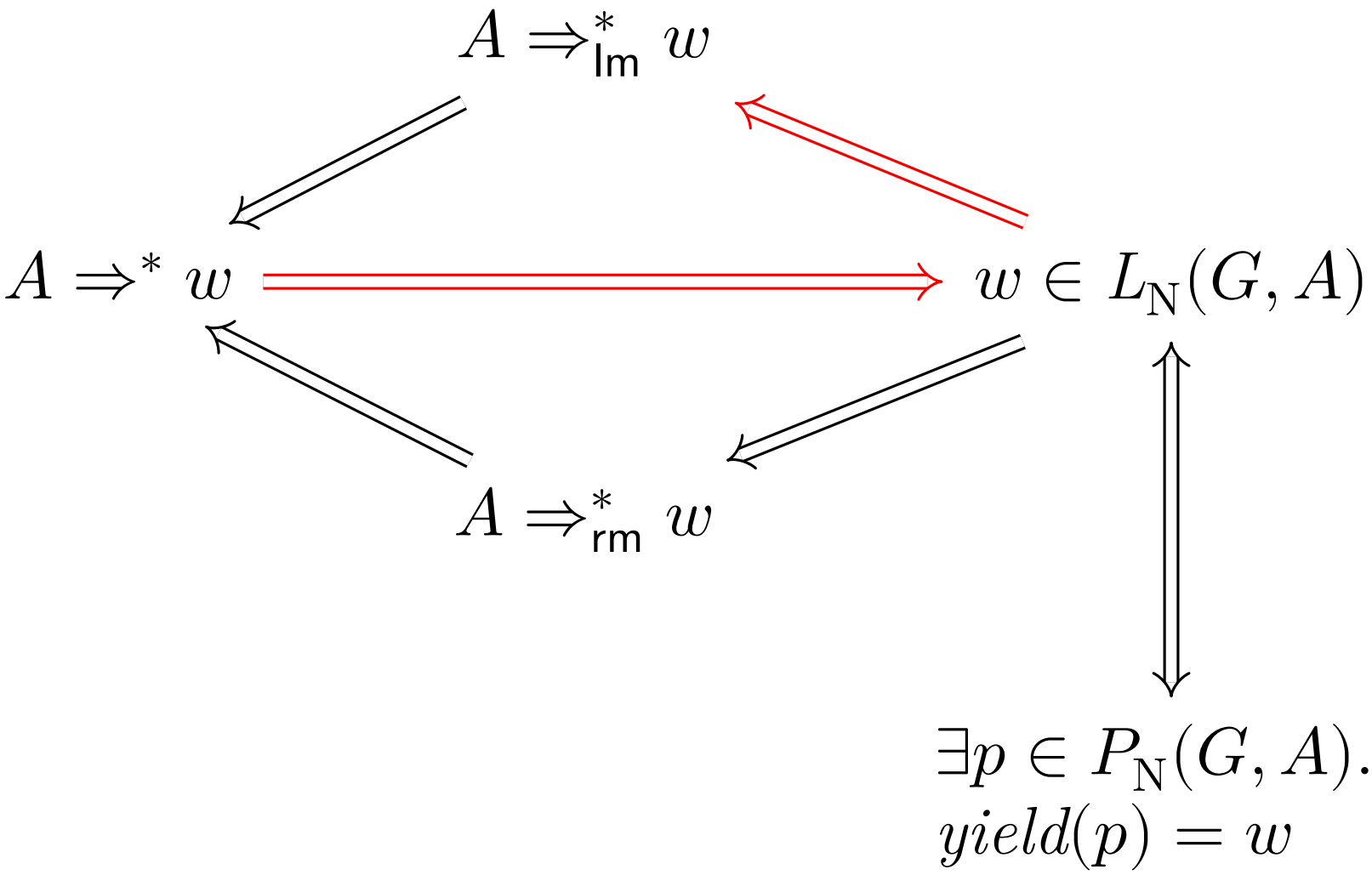
Some equivalences

With $w \in \Sigma^*$:



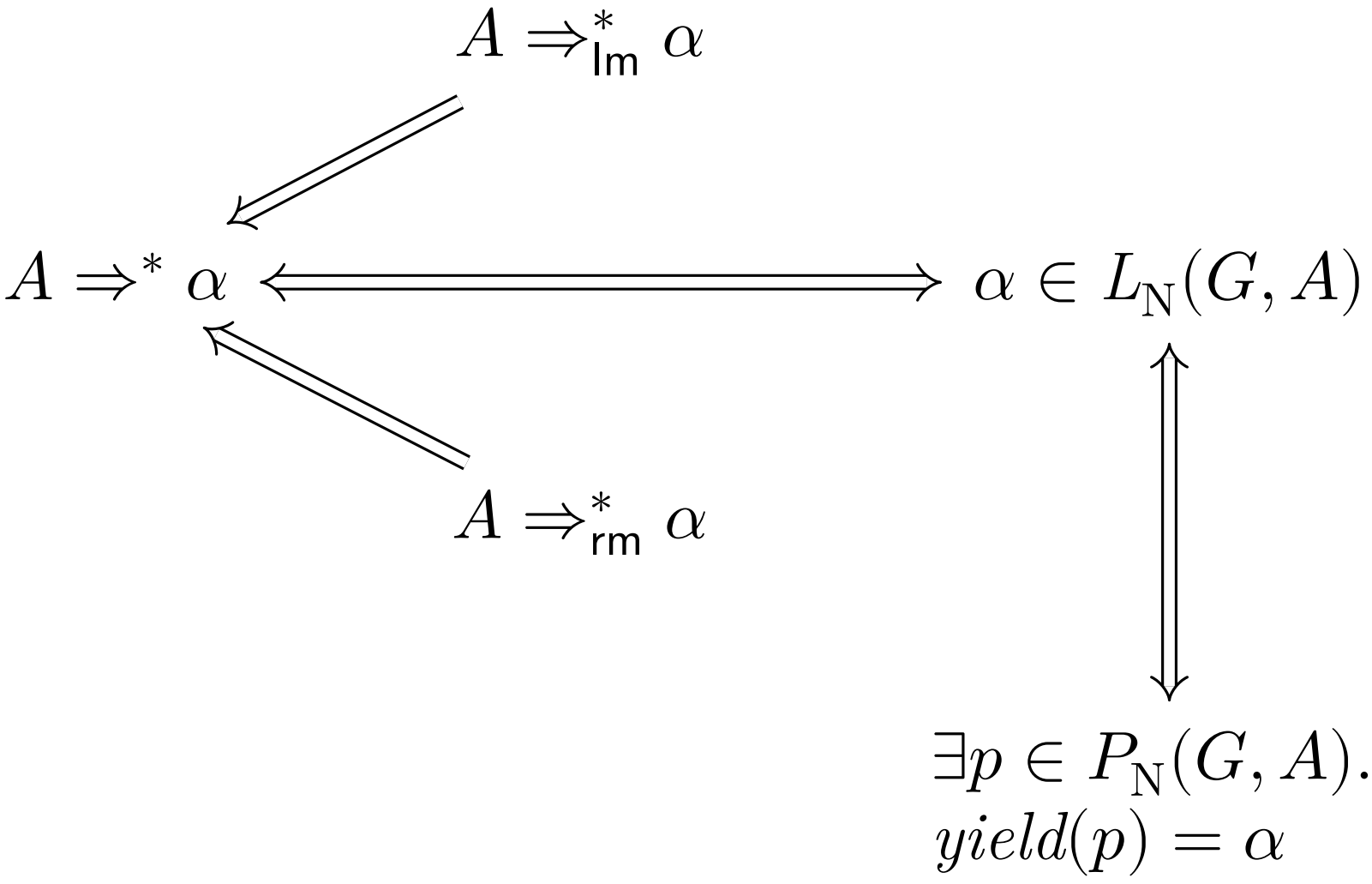
Some equivalences

With $w \in \Sigma^*$:



Some equivalences

With $\alpha \in (N \cup \Sigma)^*$:



Some equivalences

The property

$$\forall \alpha, \gamma \in (N \cup \Sigma)^*.\ (\alpha \Rightarrow^* \gamma) \ \Rightarrow\ \gamma \in L_{\mathrm{NL}}(G, \alpha)$$

can be proved by induction on the structure of the derivation, using the following lemmas:

- $\alpha \in L_{\mathrm{NL}}(G, \alpha)$
- $(\alpha \Rightarrow \beta) \ \Rightarrow\ \beta \in L_{\mathrm{NL}}(G, \alpha)$
- $\beta \in L_{\mathrm{NL}}(G, \alpha) \wedge \gamma \in L_{\mathrm{NL}}(G, \beta) \Rightarrow$
  $\gamma \in L_{\mathrm{NL}}(G, \alpha)$

Some equivalences

The property

$$\forall \alpha, \gamma \in (N \cup \Sigma)^*.\ (\alpha \Rightarrow^* \gamma) \ \Rightarrow\ \gamma \in L_{\mathrm{NL}}(G, \alpha)$$

can be proved by induction on the structure of the derivation, using the following lemmas:

- $\alpha \in L_{\mathrm{NL}}(G, \alpha)$
- $(\alpha \Rightarrow \beta) \ \Rightarrow\ \beta \in L_{\mathrm{NL}}(G, \alpha)$
- $\beta \in L_{\mathrm{NL}}(G, \alpha) \wedge \gamma \in L_{\mathrm{NL}}(G, \beta) \Rightarrow$
  $\gamma \in L_{\mathrm{NL}}(G, \alpha)$

Note that $\Rightarrow$ has two different meanings!

Some equivalences

The property

$$\forall \alpha, \beta \in (N \cup \Sigma)^*.\ (\alpha \Rightarrow \beta) \ \Rightarrow\ \beta \in L_{\mathrm{NL}}(G, \alpha)$$

can be proved using the following additional lemmas:

- $\alpha \in L_{\mathrm{N}}(G, A) \Rightarrow \alpha \in L_{\mathrm{NL}}(G, A)$
- $\beta \in L_{\mathrm{NL}}(G, \alpha) \wedge \beta' \in L_{\mathrm{NL}}(G, \alpha') \Rightarrow \beta\beta' \in L_{\mathrm{NL}}(G, \alpha\alpha')$

Prove

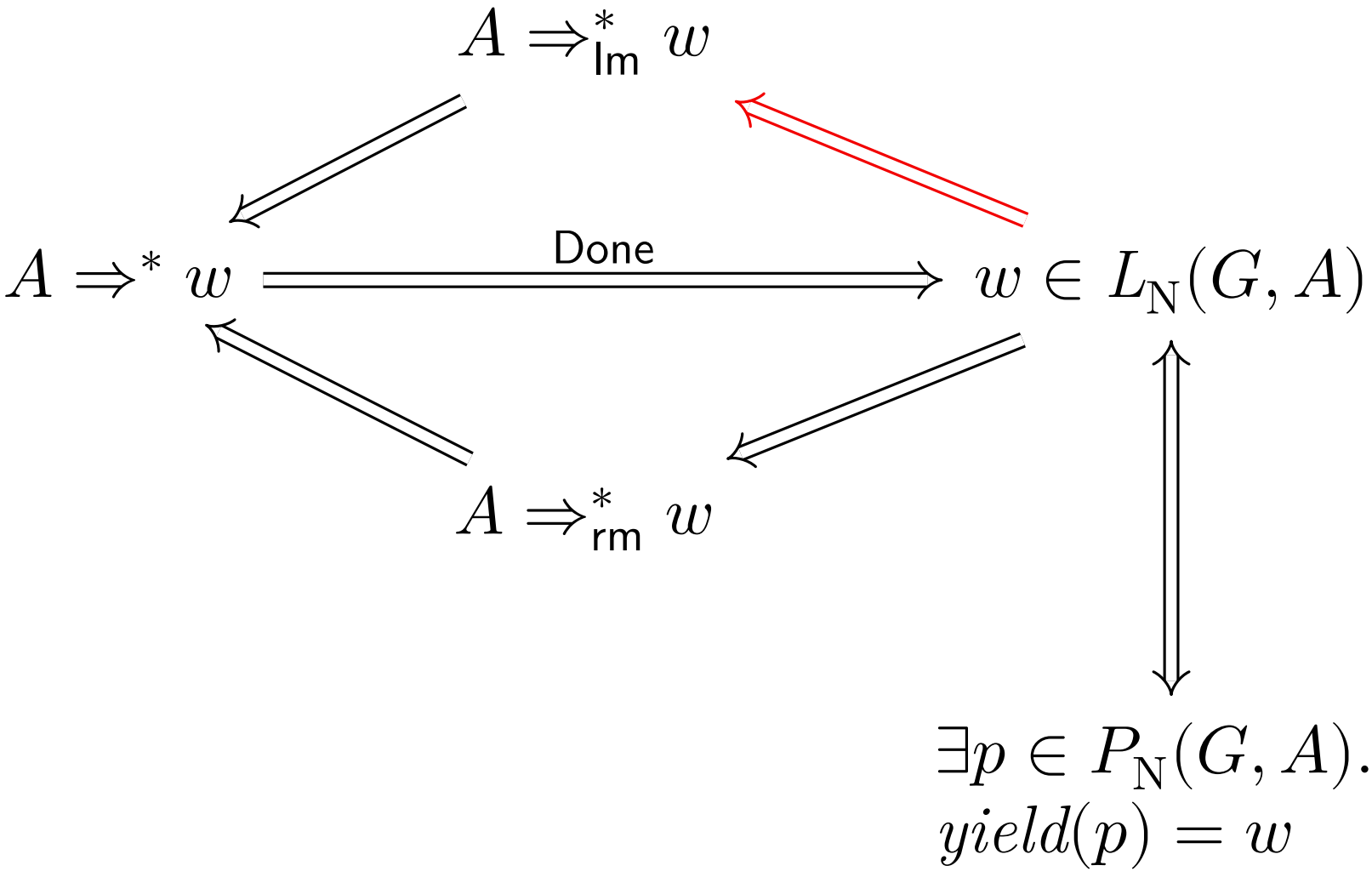
$$\forall \alpha, \beta \in (N \cup \Sigma)^*.\ (\alpha \Rightarrow \beta) \ \Rightarrow\ \beta \in L_{\mathrm{NL}}(G, \alpha).$$

Use the following rules and lemmas:

1. $$\frac{\alpha, \beta \in (N \cup \Sigma)^* \qquad A \in N \qquad (A, \gamma) \in P}{\alpha A \beta \Rightarrow \alpha \gamma \beta}$$
2. $$\frac{(A, \alpha) \in P \qquad \beta \in L_{\mathrm{NL}}(G, \alpha)}{\beta \in L_{\mathrm{N}}(G, A)}$$
3. $\alpha \in L_{\mathrm{NL}}(G, \alpha)$
4. $\alpha \in L_{\mathrm{N}}(G, A) \Rightarrow \alpha \in L_{\mathrm{NL}}(G, A)$
5. $\beta \in L_{\mathrm{NL}}(G, \alpha) \wedge \beta' \in L_{\mathrm{NL}}(G, \alpha') \Rightarrow$
   $\beta\beta' \in L_{\mathrm{NL}}(G, \alpha\alpha')$

Some equivalences

With $w \in \Sigma^*$:



Some equivalences

The property

$$\forall A \in N, w \in \Sigma^*.\ w \in L_{\mathrm{N}}(G, A) \ \Rightarrow\ (A \Rightarrow^*_{\mathsf{lm}} w)$$

can be proved by induction on the structure of the recursive inference, using the following lemmas:

- $(\alpha \Rightarrow^*_{\mathsf{lm}} \alpha') \ \Rightarrow\ (\alpha\beta \Rightarrow^*_{\mathsf{lm}} \alpha'\beta)$
- $(\alpha \Rightarrow^*_{\mathsf{lm}} \alpha') \ \Rightarrow\ (w\alpha \Rightarrow^*_{\mathsf{lm}} w\alpha')$
- $(\alpha \Rightarrow^*_{\mathsf{lm}} \beta) \wedge (\beta \Rightarrow^*_{\mathsf{lm}} \gamma) \ \Rightarrow\ (\alpha \Rightarrow^*_{\mathsf{lm}} \gamma)$

Ambiguity

Ambiguity

A grammar $G = (N, \Sigma, P, S)$ is ambiguous if there is a string $w \in \Sigma^*$ such that there are two different...

- ...parse trees in $P(G, S)$ with yield w.
- ...leftmost derivations $S \Rightarrow_{\mathsf{lm}}^* w$.
- ...rightmost derivations $S \Rightarrow_{\mathsf{rm}}^* w$.
- ...derivations of $w \in L(G, S)$.

Ambiguity

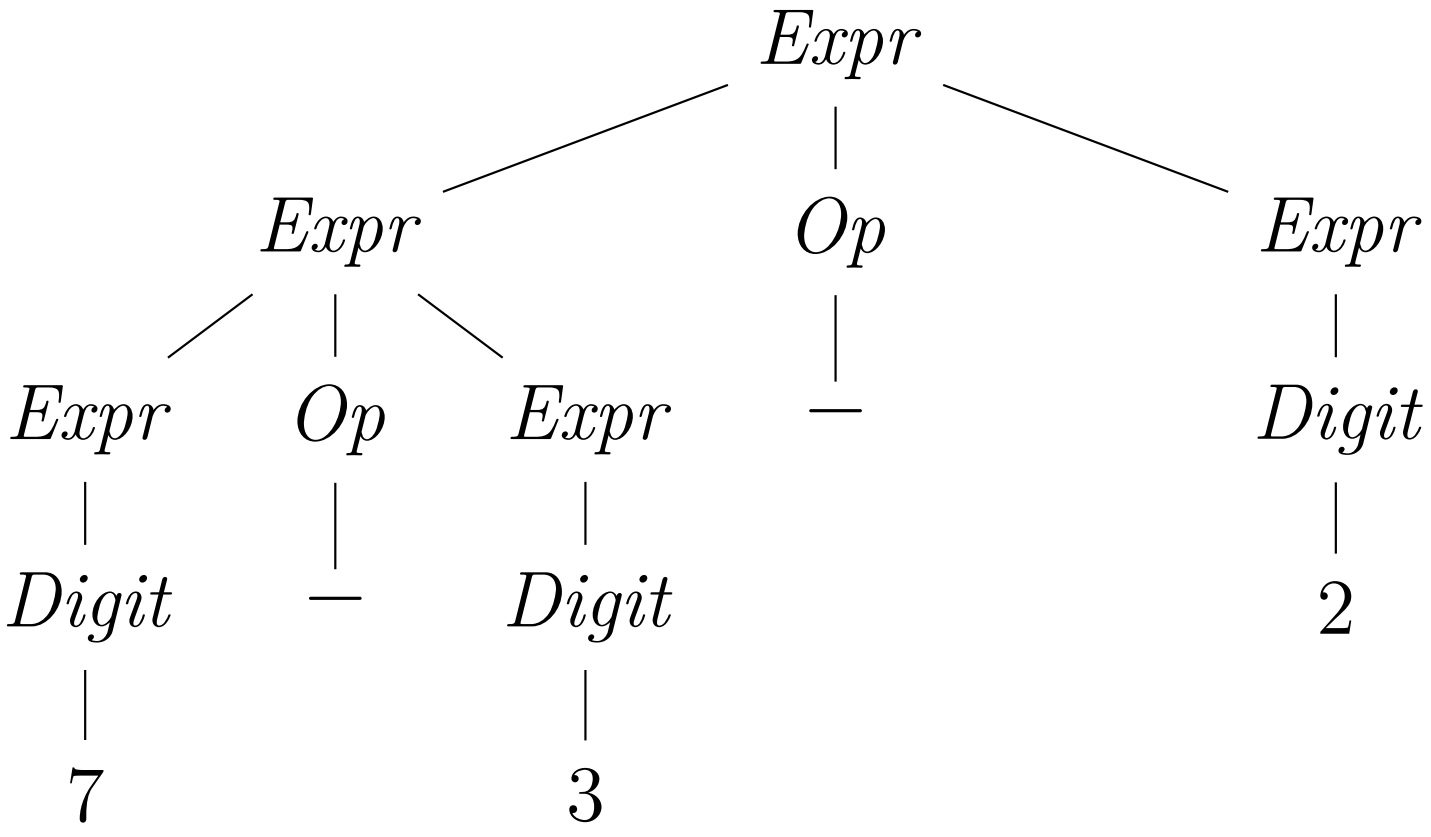
Consider the following (underspecified) context-free grammar over $\{ +, -, \cdot, /, (,) \} \cup \{ 0, 1, ..., 9 \}$:

$$
\begin{aligned}
\mathit{Expr} &\rightarrow \mathit{Expr}\ \mathit{Op}\ \mathit{Expr} \mid \mathit{Digit} \mid (\ \mathit{Expr}\ ) \\
\mathit{Op} &\rightarrow + \mid - \mid \cdot \mid / \\
\mathit{Digit} &\rightarrow 0 \mid 1 \mid ... \mid 9
\end{aligned}
$$

How should $7 - 3 - 2$ be interpreted?

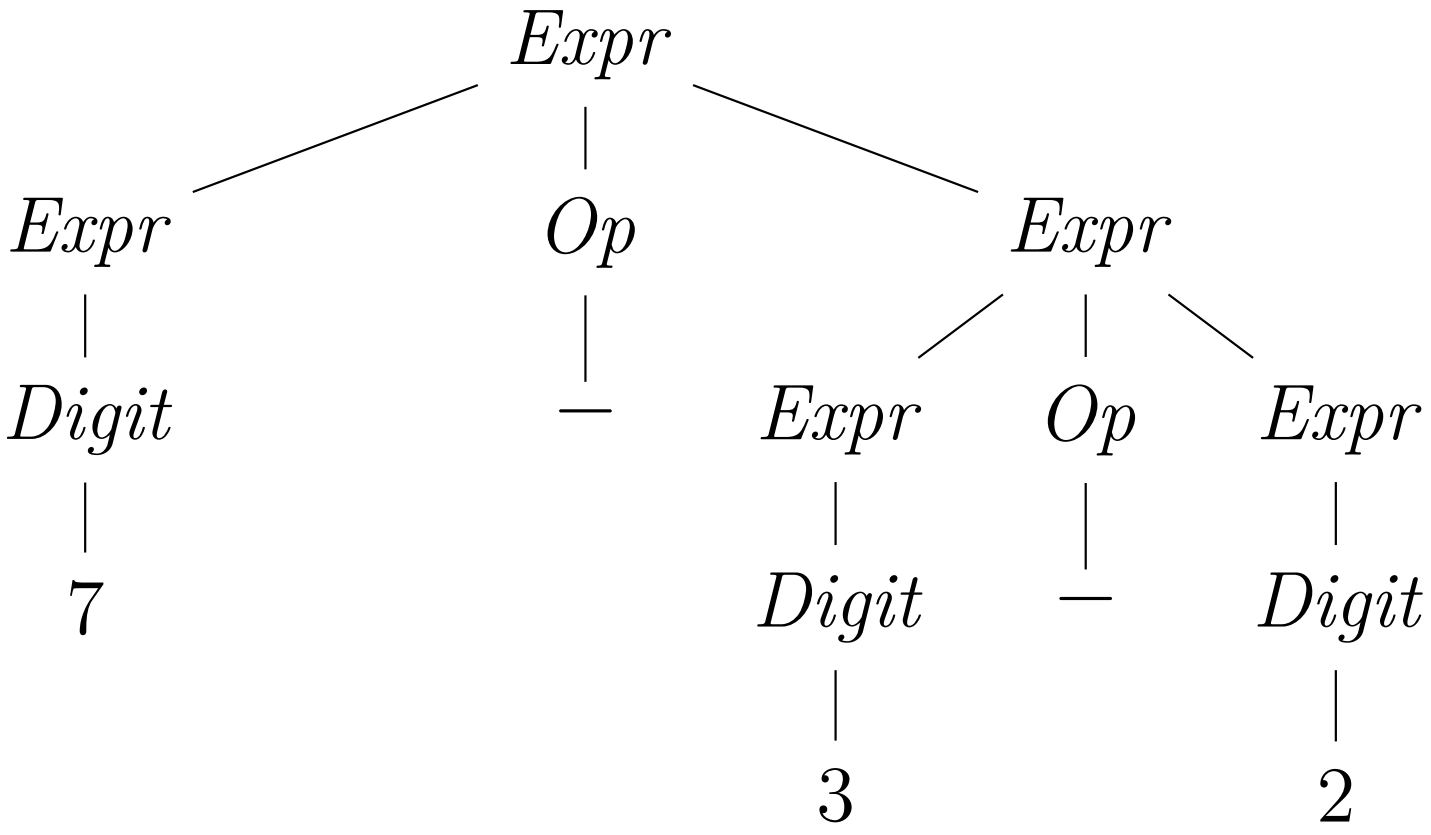
Ambiguity

A parse tree for $7 - 3 - 2$:



Ambiguity

Another parse tree for $7 - 3 - 2$:



Ambiguity

- The values differ: $(7 - 3) - 2 = 2$, but $7 - (3 - 2) = 6$.
- If a grammar is used to determine how to interpret an expression, then it may be unclear how to interpret an ambiguous string.

For which of the following sets of productions P is $(\{ S, A \}, \{ 0, 1 \}, P, S)$ an ambiguous grammar?

1. $S \to S$
2. $S \to S \mid \varepsilon$
3. $S \to 1S1 \mid 0S0 \mid \varepsilon$
4. $S \to 1S1 \mid 1A1 \mid \varepsilon, A \to 1A1 \mid S$
5. $S \to 1S1 \mid 1A1 \mid \varepsilon, A \to 0S0$

Ambiguity

- It is common to interpret $7 - 3 - 2$ as $(7 - 3) - 2$.
- The minus operator is said to "associate to the left".
- Exponentiation typically associates to the right: $3^{3^3} = 3^{(3^3)}$.

Ambiguity

- It is also common to interpret $7 \cdot 3 - 2$ as $(7 \cdot 3) - 2$, and not $7 \cdot (3 - 2)$.
- The multiplication operator is said to "bind tighter than" the subtraction operator, or to have "higher precedence".

Ambiguity

The following (underspecified) context-free grammar over $\{ +, -, \cdot, /, (,) \} \cup \{ 0, 1, ..., 9 \}$ is unambiguous:

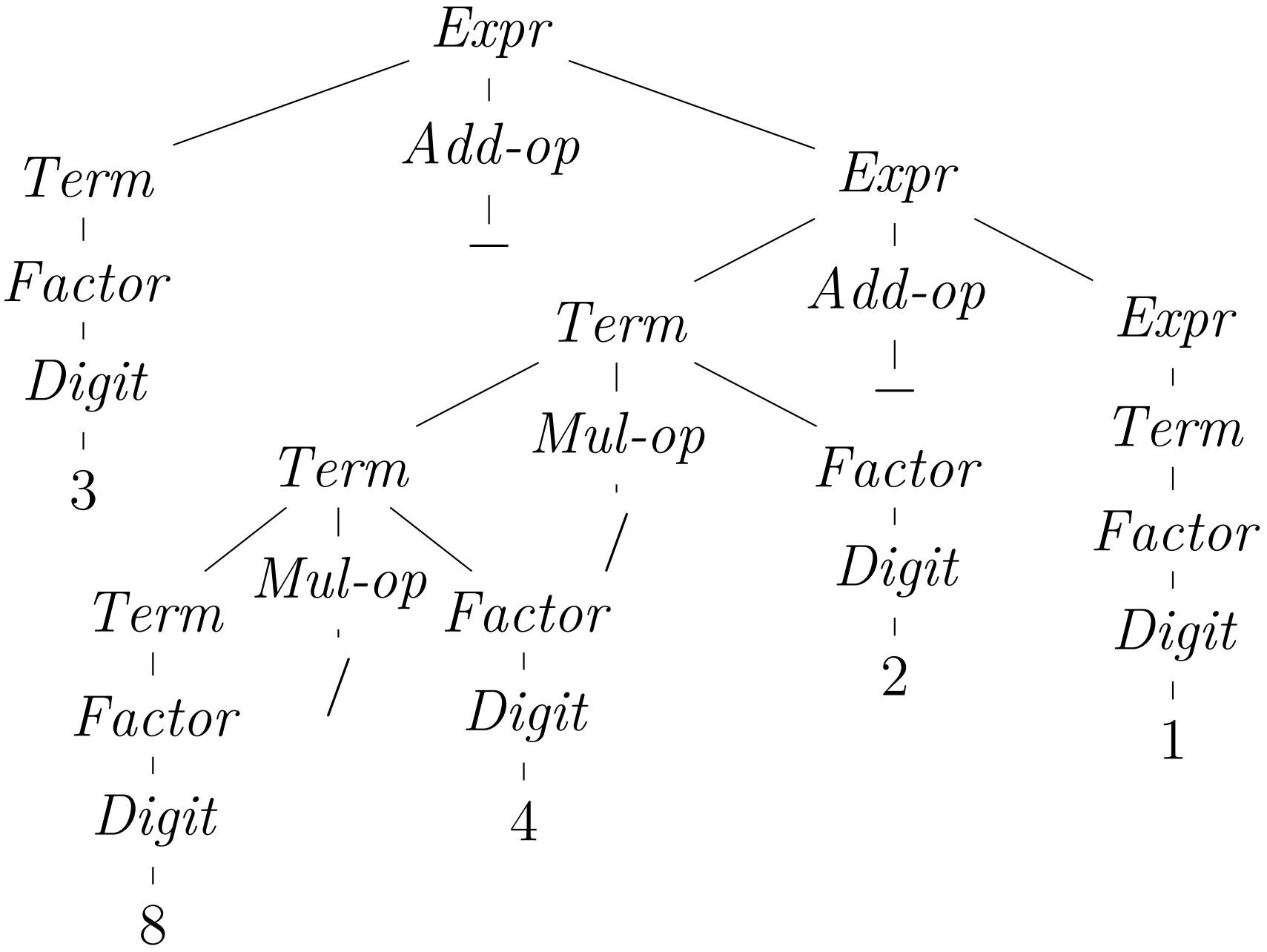
$$
\begin{aligned}
\mathit{Expr} &\rightarrow \mathit{Term}\ \mathit{Add\text{-}op}\ \mathit{Expr} \mid \mathit{Term} \\
\mathit{Term} &\rightarrow \mathit{Term}\ \mathit{Mul\text{-}op}\ \mathit{Factor} \mid \mathit{Factor} \\
\mathit{Factor} &\rightarrow \mathit{Digit} \mid (\ \mathit{Expr}\ ) \\
\mathit{Add\text{-}op} &\rightarrow + \mid - \\
\mathit{Mul\text{-}op} &\rightarrow \cdot \mid / \\
\mathit{Digit} &\rightarrow 0 \mid 1 \mid ... \mid 9
\end{aligned}
$$

Use this grammar to parse the following string. Compute the value of the expression, using the parse tree to guide the evaluation.

$$3 - 8/4/2 - 1$$

$$
\begin{aligned}
\mathit{Expr} &\rightarrow \mathit{Term}\ \mathit{Add\text{-}op}\ \mathit{Expr} \mid \mathit{Term} \\
\mathit{Term} &\rightarrow \mathit{Term}\ \mathit{Mul\text{-}op}\ \mathit{Factor} \mid \mathit{Factor} \\
\mathit{Factor} &\rightarrow \mathit{Digit} \mid (\ \mathit{Expr}\ ) \\
\mathit{Add\text{-}op} &\rightarrow + \mid - \\
\mathit{Mul\text{-}op} &\rightarrow \cdot \mid / \\
\mathit{Digit} &\rightarrow 0 \mid 1 \mid \ldots \mid 9
\end{aligned}
$$

The parse tree



Right associative?

- Subtraction is right associative for this grammar: $3 - (((8/4)/2) - 1) = 3$.
- The usual way of parsing instead leads to $(3 - ((8/4)/2)) - 1 = 1$.

Ambiguity

- It is undecidable whether a context-free grammar is ambiguous.
- However, several parser generators use restricted context-free grammars that are guaranteed to be unambiguous.
- If such a tool complains about a “conflict”, then the problem might be that the grammar is ambiguous.

Suggest some replacement for ??? that ensures that $3 \hat{} 3 \hat{} 3$ is a valid string that is interpreted as $3 \hat{} (3 \hat{} 3)$. The start symbol is E_0.

$$
\begin{array}{ll}
E_0 & \rightarrow E_0 \; \mathit{Add\text{-}op} \; E_1 \mid E_1 \\
E_1 & \rightarrow E_1 \; \mathit{Mul\text{-}op} \; E_2 \mid E_2 \\
E_2 & \rightarrow ??? \\
E_3 & \rightarrow \mathit{Digit} \mid (\; E_0 \;) \\
\mathit{Add\text{-}op} & \rightarrow + \mid - \\
\mathit{Mul\text{-}op} & \rightarrow \cdot \mid / \\
\mathit{Digit} & \rightarrow 0 \mid 1 \mid ... \mid 9
\end{array}
$$

Ambiguity

- There are context-free languages for which there are no unambiguous context-free grammars.
- Such languages are called *inherently ambiguous*.
- See the book for an example.

Designing grammars

Define a grammar for some simple (context-free) language, perhaps a tiny programming language. Try to make the grammar unambiguous.

Designing grammars

If you want to know more about the use of grammars in the specification and implementation of programming languages you might be interested in the course *Programming language technology*.

Today

- Context-free languages.
- Some equivalences.
- Ambiguity.
- Designing grammars.

Next lecture

- Grammar transformations.
- Chomsky normal form.
- The pumping lemma for context-free languages.