

Arithmetic Properties through Grammars

There are specialized algorithms¹ for parsing arithmetic expressions (written programmatically) such as

$$a + bc - 4^x \quad \equiv \quad a + b * c - 4 ** x$$

But important arithmetic properties such as **order of operations** and **associativity** can be easily expressed using **context free grammars**. All non-trivial languages embed the rules of mathematics into their language definition.²

¹[show_shunting.pdf](#)

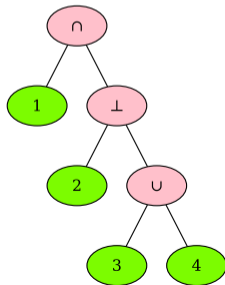
²Would you want to work in a language where $11+3 * 2$ is 28?

Arithmetic Properties through Grammars

We want to avoid our “built-in” knowledge of arithmetic, so we have to focus on the mechanics of this technique. So we’ll use \cup , \perp and \cap to represent our operators.

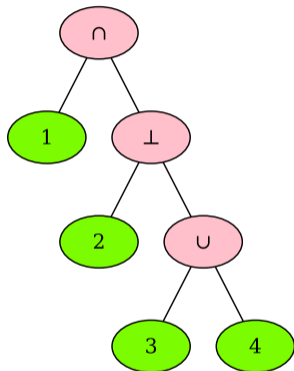
Input: $1 \cap 2 \perp 3 \cup 4$

operator	precedence	associativity
\cap	lowest (<i>L</i>)	?
\perp	middle (<i>M</i>)	?
\cup	highest (<i>H</i>)	?



Grammar Patterns for Arithmetic Operations

Input: $1 \cap 2 \perp 3 \cup 4$



Rules

1 $S \rightarrow L\$$

2 $L \rightarrow L \cap M$

3 $L \rightarrow M$

4 $M \rightarrow H \perp M$

5 $M \rightarrow H$

6 $H \rightarrow V \cup H$

7 $H \rightarrow V$

8 $V \rightarrow num$

op	prec	assoc
\cap	lowest (L)	?
\perp	middle (M)	?
\cup	highest (H)	?

Notice how there is a **nesting** or **layering** of the grammar non-terminals from **lowest to highest precedence operators** (think of numbers as the highest precedence terms)

$L \rightarrow M \rightarrow H \rightarrow V$ (rules 3, 5 and 7)

Grammar Patterns for Arithmetic Operations

Input: $1 \cap 2 \perp 3 \cup 4$

Rules

1 $S \rightarrow L \$$

2 $L \rightarrow L \cap M$

3 $L \rightarrow M$

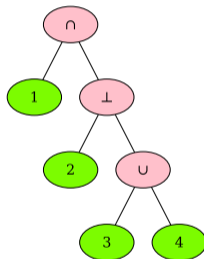
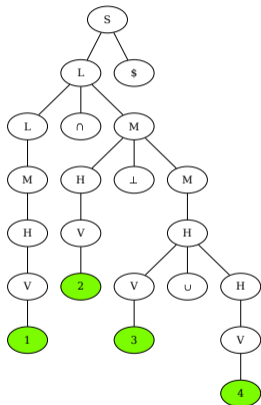
4 $M \rightarrow H \perp M$

5 $M \rightarrow H$

6 $H \rightarrow V \cup H$

7 $H \rightarrow V$

8 $V \rightarrow \text{num}$



op	prec	assoc
\cap	lowest (L)	?
\perp	middle (M)	?
\cup	highest (H)	?

We'll use **syntax directed translation** in the course to perform the simplification from "raw parse tree" to more concise **expression trees** as parsing takes place.

For now we want the larger raw parse tree to see the grammar mechanics in action.

Grammar Patterns for Arithmetic Operations

#	Rules
1	$S \rightarrow L \$$
2	$L \rightarrow L \cap M$
3	$L \rightarrow M$
4	$M \rightarrow H \perp M$
5	$M \rightarrow H$
6	$H \rightarrow V \cup H$
7	$H \rightarrow V$
8	$V \rightarrow \text{num}$

RECALL! $H \Rightarrow^+ \delta V \pi$ means the grammar symbol V can be **derived from** non-terminal H with one or more substitutions.

Both rules 6 and 7 of this expression grammar satisfy the $H \Rightarrow^+ \delta V \pi$ assertion, since δ and π are $(N \cup \Sigma)^*$.

But only rule 7 satisfies $H \Rightarrow^+ V$, since the absence of δ and π means we can't match (or generate) any other language symbols from $(N \cup \Sigma)^*$ into the derivation.

op	prec	assoc
\cap	lowest (L)	?
\perp	middle (M)	?
\cup	highest (H)	?

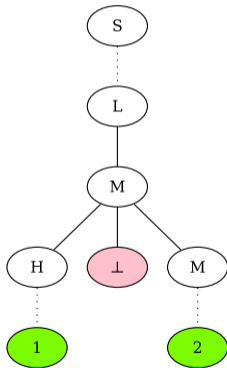
Higher Precedence Followed by Lower Precedence

Rules

- 1 $S \rightarrow L \$$
- 2 $L \rightarrow L \cap M$
- 3 $L \rightarrow M$
- 4 $M \rightarrow H \perp M$
- 5 $M \rightarrow H$
- 6 $H \rightarrow V \cup H$
- 7 $H \rightarrow V$
- 8 $V \rightarrow num$

op	prec	assoc
\cap	lowest (L)	?
\perp	middle (M)	?
\cup	highest (H)	?

Input: $1 \perp 2 \cap 3$



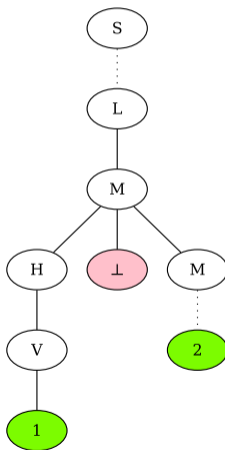
The $1 \perp$ is “captured” by rule 4, because rule 4 is the only way to incorporate the \perp operator into our parse.

Higher Precedence Followed by Lower Precedence

#	Rules
1	$S \rightarrow L \$$
2	$L \rightarrow L \cap M$
3	$L \rightarrow M$
4	$M \rightarrow H \perp M$
5	$M \rightarrow H$
6	$H \rightarrow V \cup H$
7	$H \rightarrow V$
8	$V \rightarrow num$

op	prec	assoc
\cap	lowest (<i>L</i>)	?
\perp	middle (<i>M</i>)	?
\cup	highest (<i>H</i>)	?

Input: 1 \perp 2 \cap 3



We need $H \Rightarrow^+ num$ to successfully parse the initial 1. That's not too difficult:

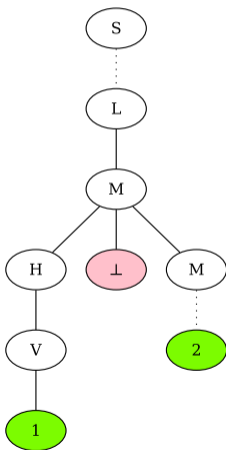
$H \rightarrow V \rightarrow num$

Higher Precedence Followed by Lower Precedence

#	Rules
1	$S \rightarrow L \$$
2	$L \rightarrow L \cap M$
3	$L \rightarrow M$
4	$M \rightarrow H \perp M$
5	$M \rightarrow H$
6	$H \rightarrow V \cup H$
7	$H \rightarrow V$
8	$V \rightarrow num$

op	prec	assoc
\cap	lowest (L)	?
\perp	middle (M)	?
\cup	highest (H)	?

Input: $1 \perp 2 \cap 3$



We need $H \Rightarrow^+ num$ to successfully parse the initial 1. That's not too difficult:

$$H \rightarrow V \rightarrow num$$

Now we need $M \Rightarrow^+ \delta num \pi$ in order to parse the 2 from the input. We could use several sequences of substitutions to accomplish this:

1. Rules 4, 7, 8
2. Rules 5, 6, 8

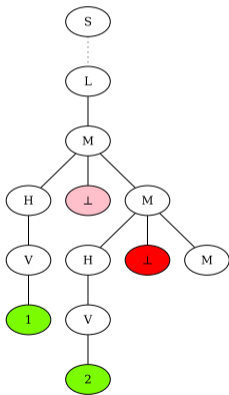
Higher Precedence Followed by Lower Precedence

Rules

- 1 $S \rightarrow L \$$
- 2 $L \rightarrow L \cap M$
- 3 $L \rightarrow M$
- 4 $M \rightarrow H \perp M$
- 5 $M \rightarrow H$
- 6 $H \rightarrow V \cup H$
- 7 $H \rightarrow V$
- 8 $V \rightarrow num$

op	prec	assoc
\cap	lowest (L)	?
\perp	middle (M)	?
\cup	highest (H)	?

Input: 1 \perp 2 \cap 3



Now we need $M \Rightarrow^+ \delta num \pi$ in order to parse the 2 from the input. We could use several sequences of substitutions to accomplish this:

1. Rules 4, 7, 8

This would require a \perp after the 2 in the input (there isn't one).

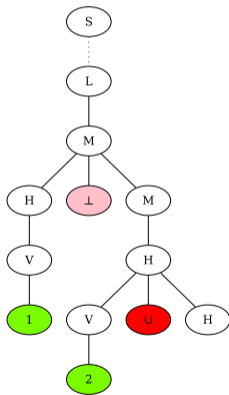
2. Rules 5, 6, 8

Higher Precedence Followed by Lower Precedence

#	Rules
1	$S \rightarrow L \$$
2	$L \rightarrow L \cap M$
3	$L \rightarrow M$
4	$M \rightarrow H \perp M$
5	$M \rightarrow H$
6	$H \rightarrow V \cup H$
7	$H \rightarrow V$
8	$V \rightarrow \text{num}$

op	prec	assoc
\cap	lowest (L)	?
\perp	middle (M)	?
\cup	highest (H)	?

Input: $1 \perp 2 \cap 3$



Now we need $M \Rightarrow^+ \delta \text{num} \pi$ in order to parse the 2 from the input. We could use several sequences of substitutions to accomplish this:

1. Rules 4, 7, 8
2. Rules 5, 6, 8

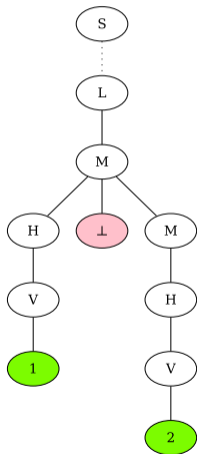
This would require a \cup after the 2 in the input (there isn't one).

Higher Precedence Followed by Lower Precedence

#	Rules
1	$S \rightarrow L \$$
2	$L \rightarrow L \cap M$
3	$L \rightarrow M$
4	$M \rightarrow H \perp M$
5	$M \rightarrow H$
6	$H \rightarrow V \cup H$
7	$H \rightarrow V$
8	$V \rightarrow num$

op	prec	assoc
\cap	lowest (L)	?
\perp	middle (M)	?
\cup	highest (H)	?

Input: $1 \perp 2 \cap 3$



The only way to accomplish $M \Rightarrow^+ \delta num \pi$ for the 2 in the input is using rules 5, 7 and 8 which don't introduce terminals **absent from the input**.

The remaining input, $\cap 3$, **must be incorporated into the parse tree between the root S and its child L tree** created to parse the \perp operator.

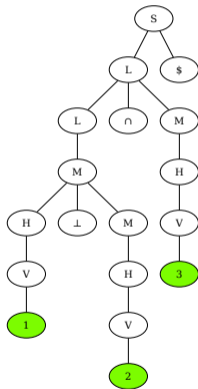
Higher Precedence Followed by Lower Precedence

Rules

- 1 $S \rightarrow L \$$
- 2 $L \rightarrow L \cap M$
- 3 $L \rightarrow M$
- 4 $M \rightarrow H \perp M$
- 5 $M \rightarrow H$
- 6 $H \rightarrow V \cup H$
- 7 $H \rightarrow V$
- 8 $V \rightarrow \text{num}$

op	prec	assoc
\cap	lowest (<i>L</i>)	?
\perp	middle (<i>M</i>)	?
\cup	highest (<i>H</i>)	?

Input: $1 \perp 2 \cap 3$



With this insight, it's not difficult to deduce the remaining portion of the tree. We need rule 2 (it's the only rule with the \cap operator) and conveniently we can incorporate it by the recursive property of L in rule 2.

And this is how higher-to-lower precedence can be evaluated in an **unambiguous** manner when the operator grammar is written correctly.

Let's convince ourselves this works for lower-to-higher precedence...

Lower Precedence Followed by Higher Precedence

Rules

- 1 $S \rightarrow L \$$
- 2 $L \rightarrow L \cap M$
- 3 $L \rightarrow M$
- 4 $M \rightarrow H \perp M$
- 5 $M \rightarrow H$
- 6 $H \rightarrow V \cup H$
- 7 $H \rightarrow V$
- 8 $V \rightarrow num$

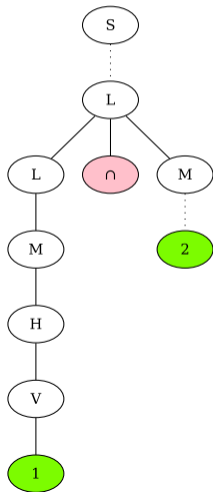
The $1 \cap$ is “captured” by rule 2 (it’s the only rule with the \cap operator in its RHS).

We need $L \Rightarrow^+ num$ to successfully parse the initial 1. That’s not too difficult:

$$L \rightarrow M \rightarrow V \rightarrow num$$

op	prec	assoc
\cap	lowest (<i>L</i>)	?
\perp	middle (<i>M</i>)	?
\cup	highest (<i>H</i>)	?

Input: $1 \cap 2 \perp 3$



Lower Precedence Followed by Higher Precedence

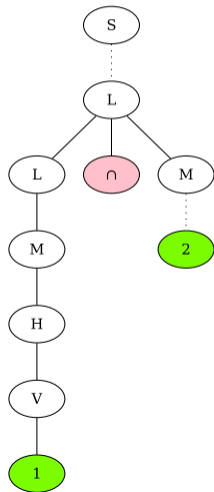
Rules

- 1 $S \rightarrow L \$$
- 2 $L \rightarrow L \cap M$
- 3 $L \rightarrow M$
- 4 $M \rightarrow H \perp M$
- 5 $M \rightarrow H$
- 6 $H \rightarrow V \cup H$
- 7 $H \rightarrow V$
- 8 $V \rightarrow num$

Now we need $M \Rightarrow^+ \delta num \pi$.

Input: $1 \cap 2 \perp 3$

op	prec	assoc
\cap	lowest (L)	?
\perp	middle (M)	?
\cup	highest (H)	?



Lower Precedence Followed by Higher Precedence

Rules

- 1 $S \rightarrow L \$$
- 2 $L \rightarrow L \cap M$
- 3 $L \rightarrow M$
- 4 $M \rightarrow H \perp M$
- 5 $M \rightarrow H$
- 6 $H \rightarrow V \cup H$
- 7 $H \rightarrow V$
- 8 $V \rightarrow num$

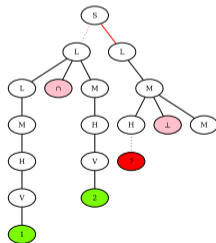
We could use rules 5, 7 and 8; but then we must parse the \perp following 2 in the input.

Could we use rules 3 and 4 for the \perp ?

No:

1. There is no $S \rightarrow LL$ production rule
2. and rule 4's RHS H would have to $H \Rightarrow^+ \delta num \pi$ but there isn't another num between 2 and \perp .

Input: $1 \cap 2 \perp 3$



op	prec	assoc
\cap	lowest (L)	?
\perp	middle (M)	?
\cup	highest (H)	?

Lower Precedence Followed by Higher Precedence

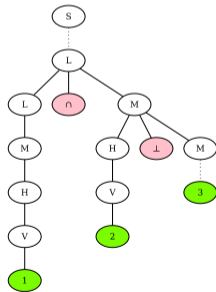
Rules

- 1 $S \rightarrow L \$$
- 2 $L \rightarrow L \cap M$
- 3 $L \rightarrow M$
- 4 $M \rightarrow H \perp M$
- 5 $M \rightarrow H$
- 6 $H \rightarrow V \cup H$
- 7 $H \rightarrow V$
- 8 $V \rightarrow num$

The only way to parse the $2 \perp$ phrase of the input is to resolve $M \Rightarrow^+ \delta num \perp \pi$ using rule 4.

op	prec	assoc
\cap	lowest (L)	?
\perp	middle (M)	?
\cup	highest (H)	?

Input: $1 \cap 2 \perp 3$



Lower Precedence Followed by Higher Precedence

Rules

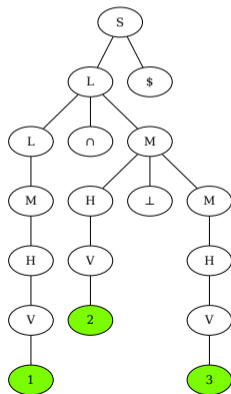
- 1 $S \rightarrow L \$$
- 2 $L \rightarrow L \cap M$
- 3 $L \rightarrow M$
- 4 $M \rightarrow H \perp M$
- 5 $M \rightarrow H$
- 6 $H \rightarrow V \cup H$
- 7 $H \rightarrow V$
- 8 $V \rightarrow num$

The only way to parse the $2 \perp$ phrase of the input is to resolve $M \Rightarrow^+ \delta num \perp \pi$ using rule 4.

And rules 5, 7 and 8 allow $M \Rightarrow^+ num$.

Input: $1 \cap 2 \perp 3$

op	prec	assoc
\cap	lowest (<i>L</i>)	?
\perp	middle (<i>M</i>)	?
\cup	highest (<i>H</i>)	?



Grammar Pattern for Operator Precedence — The Takeaway

Rules

1 $S \rightarrow L \$$

2 $L \rightarrow L \cap M$

3 $L \rightarrow M$

4 $M \rightarrow H \perp M$

5 $M \rightarrow H$

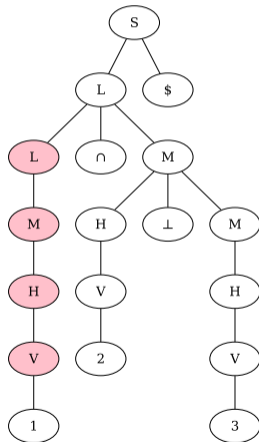
6 $H \rightarrow V \cup H$

7 $H \rightarrow V$

8 $V \rightarrow \text{num}$

op	prec	assoc
\cap	lowest (L)	?
\perp	middle (M)	?
\cup	highest (H)	?

Input: 1 \cap 2 \perp 3



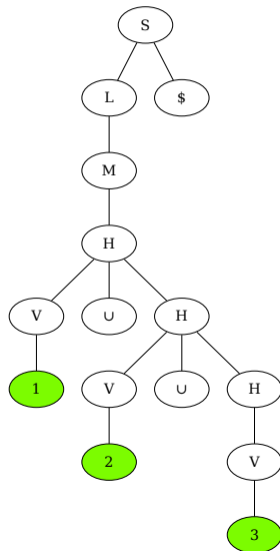
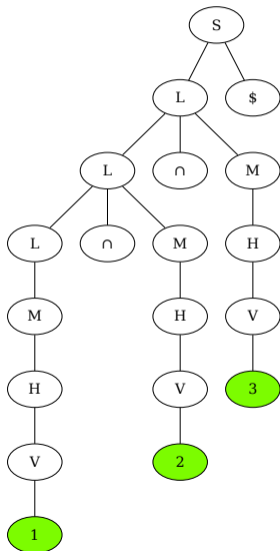
Operator precedence is encoded into a grammar as strict “levels” or “layers”. A higher precedence level **must be rewritten** as the RHS non-terminal of the next lower precedence level in order for its value to “flow up” the expression tree into the final result.

Grammar Pattern for Operator Associativity

Rules

-
- 1 $S \rightarrow L \$$
 - 2 $L \rightarrow L \cap M$
 - 3 $L \rightarrow M$
 - 4 $M \rightarrow H \perp M$
 - 5 $M \rightarrow H$
 - 6 $H \rightarrow V \cup H$
 - 7 $H \rightarrow V$
 - 8 $V \rightarrow num$

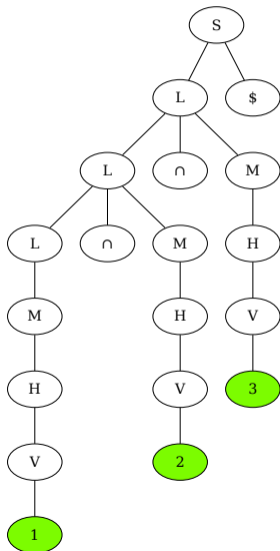
op	prec	assoc
\cap	lowest (<i>L</i>)	?
\perp	middle (<i>M</i>)	?
\cup	highest (<i>H</i>)	?



Left Associative Op: $1 \cap 2 \cap 3$

#	Rules
1	$S \rightarrow L \$$
2	$L \rightarrow L \cap M$
3	$L \rightarrow M$
4	$M \rightarrow H \perp M$
5	$M \rightarrow H$
6	$H \rightarrow V \cup H$
7	$H \rightarrow V$
8	$V \rightarrow \text{num}$

op	prec	assoc
\cap	lowest (L)	left
\perp	middle (M)	?
\cup	highest (H)	?



There are two \cap operators in our input source, so we'll have to use two applications of rule 2 (the only rule permitting the \cap symbol).

The grammar does not permit L to be below M s in the parse tree. Why? There are no production rules permitting $M \Rightarrow^+ \delta L \pi$.

So there is only one way to combine these two rewrite rules — recursing down the LH operand:

$$S \Rightarrow L \$$$

$$S \Rightarrow L \cap_2 M_2 \$$$

$$S \Rightarrow_{lm} L_1 \cap_1 M_1 \cap_2 M_2$$

$$S \Rightarrow^+ 1 \cap_1 2 \cap_2 3$$

This left recursion bound the 2 in the input to \cap_1 , \cap is **left associative!**

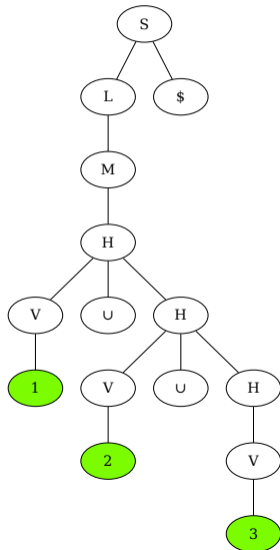
Right Associative Op: 1 U 2 U 3

#	Rules
1	$S \rightarrow L \$$
2	$L \rightarrow L \cap M$
3	$L \rightarrow M$
4	$M \rightarrow H \perp M$
5	$M \rightarrow H$
6	$H \rightarrow V \cup H$
7	$H \rightarrow V$
8	$V \rightarrow num$

Likewise, the **right recursive rules 4** and 6 means means \perp and \cup are **right associative** operators.

$$\begin{aligned}
 S &\Rightarrow^+ H \$ \\
 S &\Rightarrow V_1 \cup_1 H \$ \\
 S &\Rightarrow_{rm}^+ V_1 \cup_1 V_2 \cup_2 H \\
 S &\Rightarrow^+ 1 \cup_1 2 \cup_2 3
 \end{aligned}$$

(There are no rules permitting $V \Rightarrow^+ \delta \cup_1 \pi$, we must incorporate \cup_2 in a tree rooted at the RH operand (H) of rule 6.)



Grammar Pattern for Parenthetical Precedence (Rule 9)

#	Rules
1	$S \rightarrow L \$$
2	$L \rightarrow L \cap M$
3	$L \rightarrow M$
4	$M \rightarrow H \perp M$
5	$M \rightarrow H$
6	$H \rightarrow V \cup H$
7	$H \rightarrow V$
8	$V \rightarrow \text{num}$
9	$V \rightarrow (L)$

Parenthetical overrides of precedence are straightforward to express in a grammar.

1. Identify the **non-terminal** that captures the lowest precedence expressions (L in this grammar).
2. Identify the **non-terminal** that captures values, variables, numbers, and literals (in this grammar, non-terminal V).
3. Modify the grammar to treat (L) as if it were a variable or value (rule 9); where $($ and $)$ are the open and closing bracket symbols used for grouping in the language.

op	prec	assoc
\cap	lowest (L)	left
\perp	middle (M)	right
\cup	high (H)	right
(\cdot)	highest (V)	none

Grammar Pattern for Parenthetical Precedence (Rule 9)

Precedence: $(1 \cap 2) \cup 3$

Associativity: $(1 \perp 2) \perp 3$

Rules

1 $S \rightarrow L \$$

2 $L \rightarrow L \cap M$

3 $L \rightarrow M$

4 $M \rightarrow H \perp M$

5 $M \rightarrow H$

6 $H \rightarrow V \cup H$

7 $H \rightarrow V$

8 $V \rightarrow \text{num}$

9 $V \rightarrow (L)$

op	prec	assoc
\cap	lowest (L)	left
\perp	middle (M)	right
\cup	high (H)	right
(\cdot)	highest (V)	none

