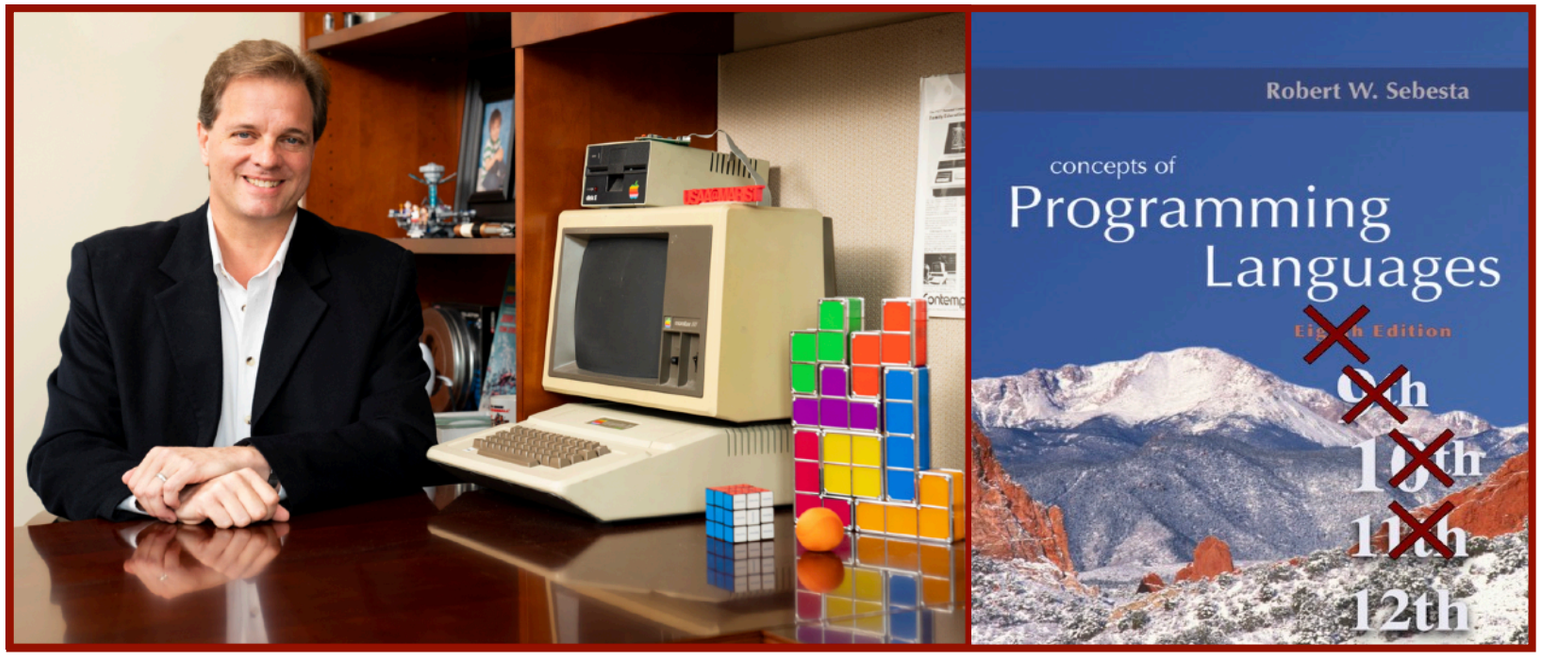
Ambiguous Grammars



Alan G. Labouseur, Ph.D.
Alan.Labouseur@Marist.edu

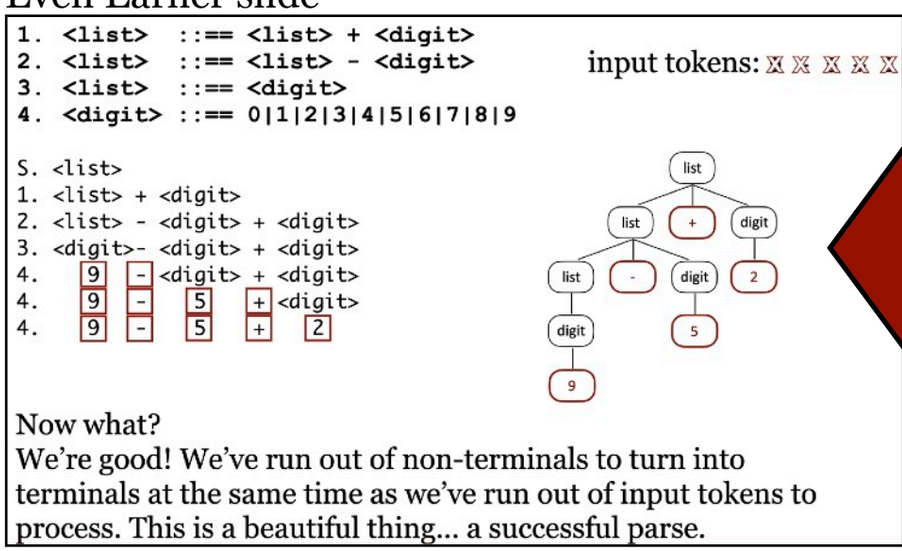
Ambiguous Context-Free Grammars

Remember this unambiguous grammar?

```
1. <list>  ::== <list> + <digit>
2. <list>  ::== <list> - <digit>
3. <list>  ::== <digit>
4. <digit> ::== 0|1|2|3|4|5|6|7|8|9
```

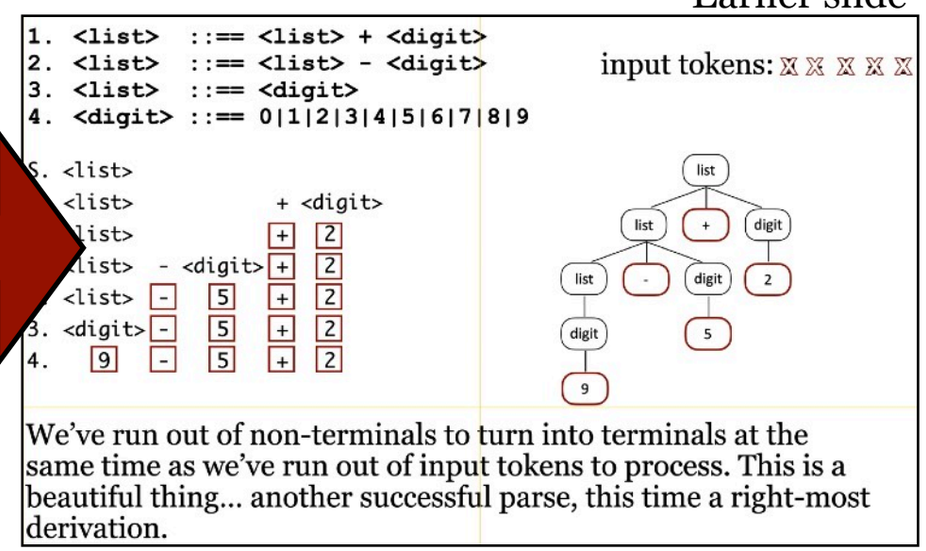
input tokens: `9 - 5 + 2`

Even Earlier slide



Now what?
We're good! We've run out of non-terminals to turn into terminals at the same time as we've run out of input tokens to process. This is a beautiful thing... a successful parse.

Earlier slide



We've run out of non-terminals to turn into terminals at the same time as we've run out of input tokens to process. This is a beautiful thing... another successful parse, this time a right-most derivation.

What if we had a grammar a little less cooperative?

Ambiguous Context-Free Grammars

A less cooperative grammar

```
1.  <expr> ::== <expr> <op> <expr>
2.         ::== num
3.  <op>   ::== ÷
4.         ::== -
```

Ambiguous Context-Free Grammars

A less cooperative grammar

```
1.  <expr>  ::== <expr> <op> <expr>
2.          ::== num
3.  <op>    ::== ÷
4.          ::== -
```

input tokens: `8 - 4 ÷ 2`

Left-most derivation

<expr>

Ambiguous Context-Free Grammars

A less cooperative grammar

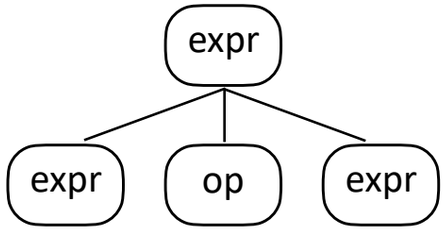
```
1.  <expr>  ::== <expr> <op> <expr>
2.          ::== num
3.  <op>    ::== ÷
4.          ::== -
```

input tokens: `8 - 4 ÷ 2`

Left-most derivation

<expr>

<expr> <op> <expr>



Ambiguous Context-Free Grammars

A less cooperative grammar

```
1.  <expr>  ::== <expr> <op> <expr>
2.          ::== num
3.  <op>    ::== ÷
4.          ::== -
```

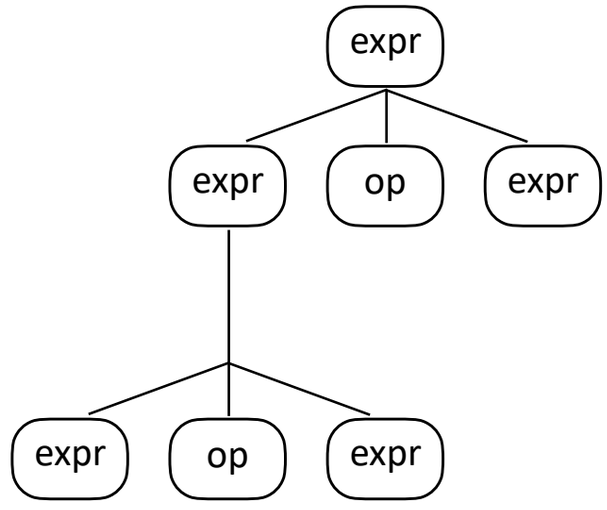
input tokens: `8 - 4 ÷ 2`

Left-most derivation

<expr>

<expr> <op> <expr>

<expr> <op> <expr> <op> <expr>



Ambiguous Context-Free Grammars

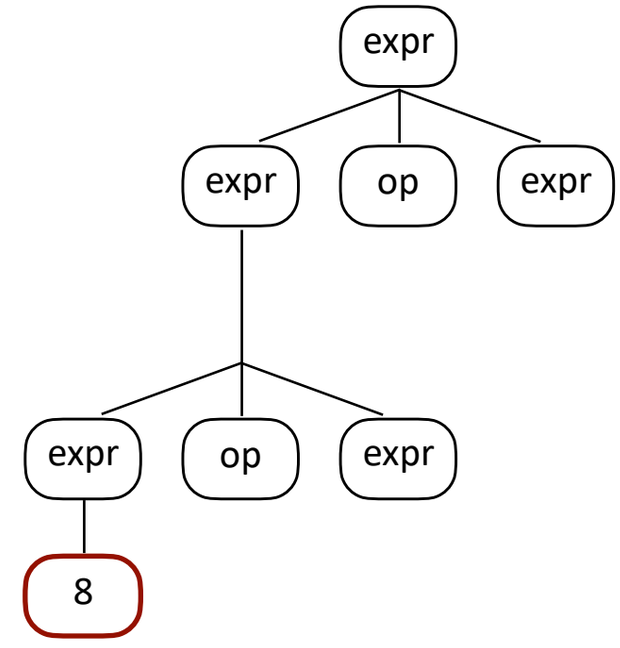
A less cooperative grammar

```
1.  <expr>  ::== <expr> <op> <expr>
2.          ::== num
3.  <op>    ::== ÷
4.          ::== -
```

input tokens: `8 - 4 ÷ 2`

Left-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
[num,8]<op> <expr> <op> <expr>
```



Ambiguous Context-Free Grammars

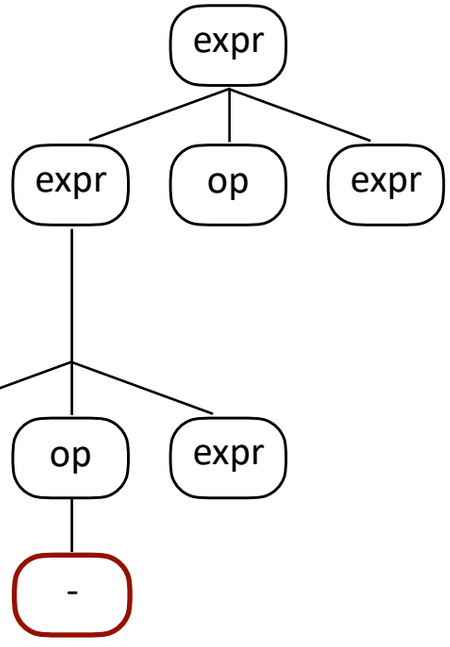
A less cooperative grammar

```
1.  <expr>  ::== <expr> <op> <expr>
2.          ::== num
3.  <op>    ::== ÷
4.          ::== -
```

input tokens: `8 - 4 ÷ 2`

Left-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
[num,8]<op> <expr> <op> <expr>
[num,8]  -  <expr> <op> <expr>
```



Ambiguous Context-Free Grammars

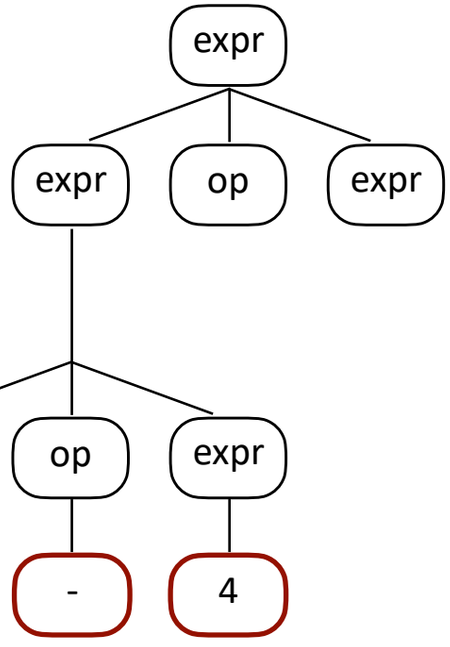
A less cooperative grammar

```
1.  <expr> ::== <expr> <op> <expr>
2.         ::== num
3.  <op>   ::== ÷
4.         ::== -
```

input tokens: `8 - 4 ÷ 2`

Left-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
[num,8]<op> <expr> <op> <expr>
[num,8]  -  <expr> <op> <expr>
[num,8]  -  [num,4]<op> <expr>
```



Ambiguous Context-Free Grammars

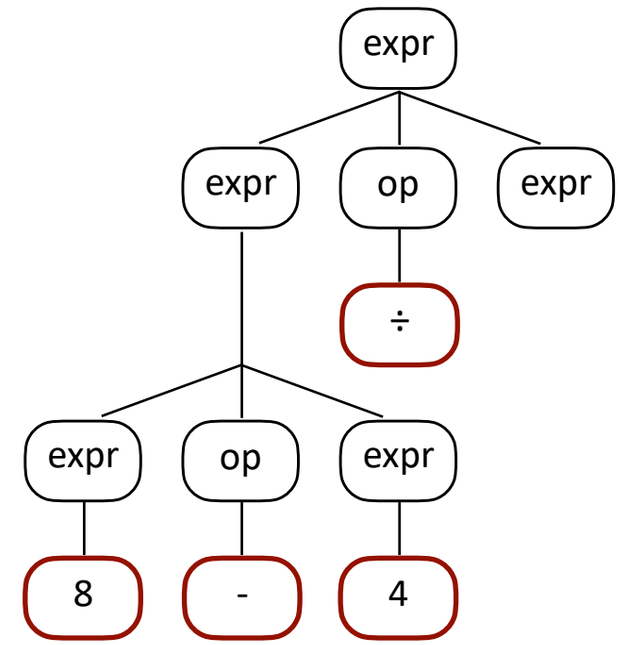
A less cooperative grammar

```
1.  <expr> ::== <expr> <op> <expr>
2.         ::== num
3.  <op>   ::== ÷
4.         ::== -
```

input tokens: `8 - 4 ÷ 2`

Left-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
[num,8]<op> <expr> <op> <expr>
[num,8]  -  <expr> <op> <expr>
[num,8]  -  [num,4]<op> <expr>
[num,8]  -  [num,4]  ÷  <expr>
```



Ambiguous Context-Free Grammars

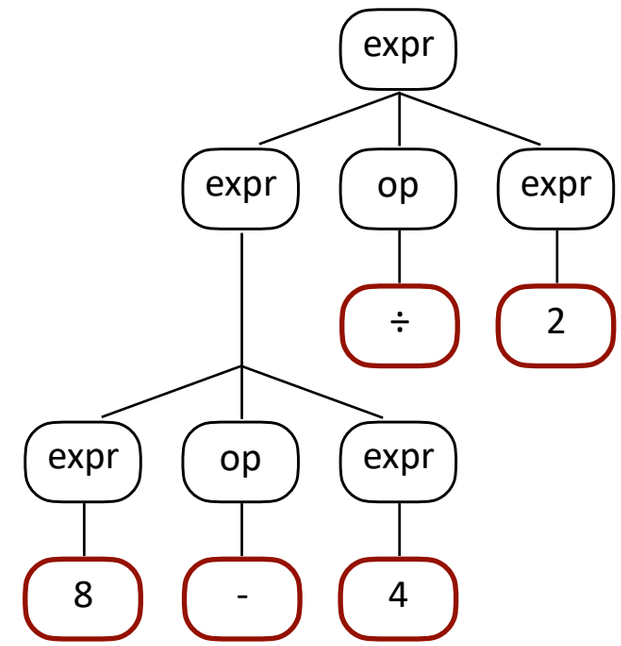
A less cooperative grammar

```
1.  <expr> ::== <expr> <op> <expr>
2.         ::== num
3.  <op>   ::== ÷
4.         ::== -
```

input tokens: `8 - 4 ÷ 2`

Left-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
[num,8]<op> <expr> <op> <expr>
[num,8]  -  <expr> <op> <expr>
[num,8]  -  [num,4]<op> <expr>
[num,8]  -  [num,4]  ÷  <expr>
[num,8]  -  [num,4]  ÷  [num,2]
```



Ambiguous Context-Free Grammars

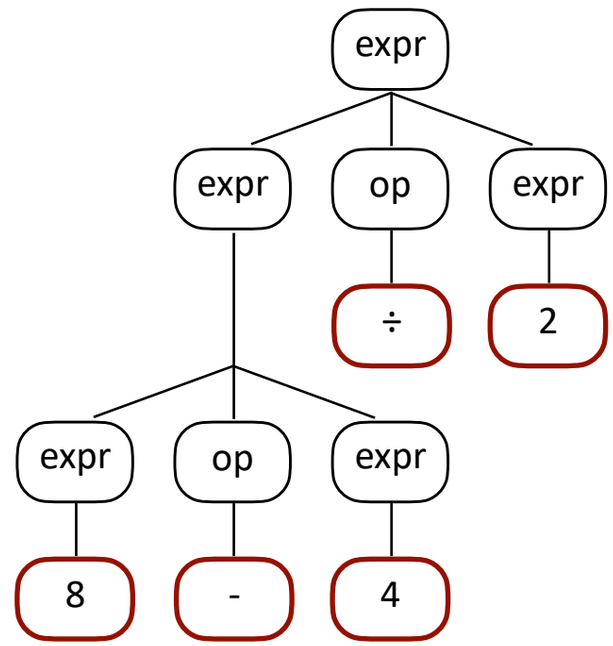
A less cooperative grammar

```
1. <expr> ::== <expr> <op> <expr>
2.        ::== num
3. <op>   ::== ÷
4.        ::== -
```

input tokens: `8 - 4 ÷ 2`

Left-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
[num,8]<op> <expr> <op> <expr>
[num,8]  -  <expr> <op> <expr>
[num,8]  -  [num,4]<op> <expr>
[num,8]  -  [num,4]  ÷  <expr>
[num,8]  -  [num,4]  ÷  [num,2]
```



A depth-first in-order traversal of a CST where we print the leaf nodes as we go will yield 8 - 4 ÷ 2.

The shape of the CST determines the grouping.

Ambiguous Context-Free Grammars

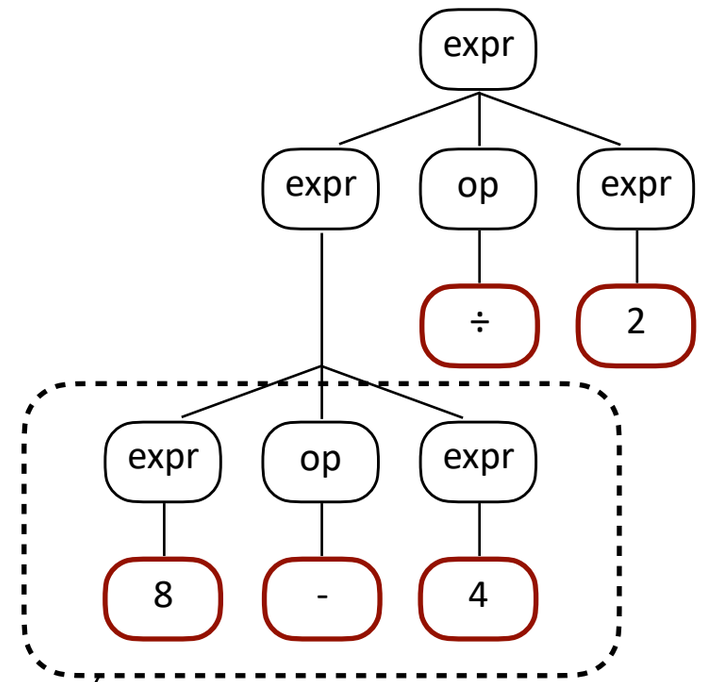
A less cooperative grammar

```
1.  <expr> ::== <expr> <op> <expr>
2.         ::== num
3.  <op>   ::== ÷
4.         ::== -
```

input tokens: `8 - 4 ÷ 2`

Left-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
[num,8]<op> <expr> <op> <expr>
[num,8]  -  <expr> <op> <expr>
[num,8]  -  [num,4]<op> <expr>
[num,8]  -  [num,4]  ÷  <expr>
[num,8]  -  [num,4]  ÷  [num,2]
```



Because of the shape of the tree: (8 - 4) ÷ 2

Ambiguous Context-Free Grammars

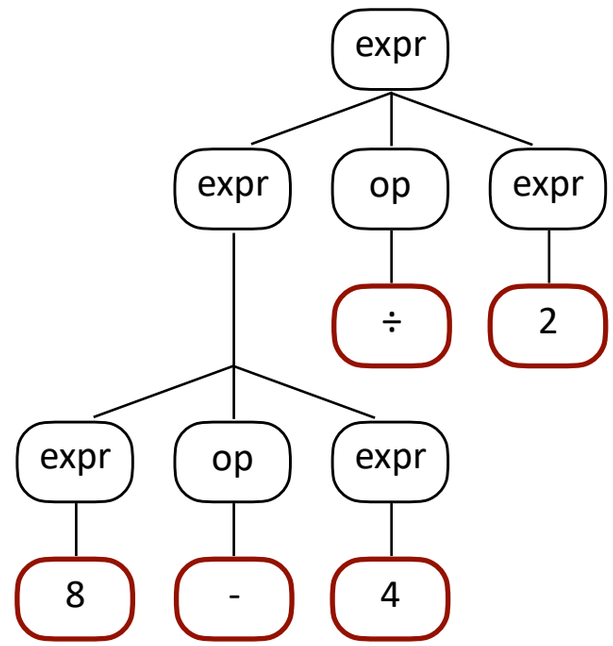
A less cooperative grammar

```
1. <expr> ::== <expr> <op> <expr>
2.        ::== num
3. <op>   ::== ÷
4.        ::== -
```

input tokens: `8 - 4 ÷ 2`

Left-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
[num,8]<op> <expr> <op> <expr>
[num,8]  -  <expr> <op> <expr>
[num,8]  -  [num,4]<op> <expr>
[num,8]  -  [num,4]  ÷  <expr>
[num,8]  -  [num,4]  ÷  [num,2]
```



Because of the shape of the tree: (8 - 4) ÷ 2 = 2, which is wrong.
We broke math.

Ambiguous Context-Free Grammars

A less cooperative grammar

```
1.  <expr>  ::== <expr> <op> <expr>
2.          ::== num
3.  <op>    ::== ÷
4.          ::== -
```

Right-most derivation

<expr>

Ambiguous Context-Free Grammars

A less cooperative grammar

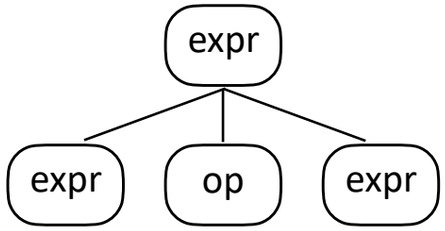
```
1.  <expr>  ::== <expr> <op> <expr>
2.          ::== num
3.  <op>    ::== ÷
4.          ::== -
```

input tokens: `8 - 4 ÷ 2`

Right-most derivation

<expr>

<expr> <op> <expr>



Ambiguous Context-Free Grammars

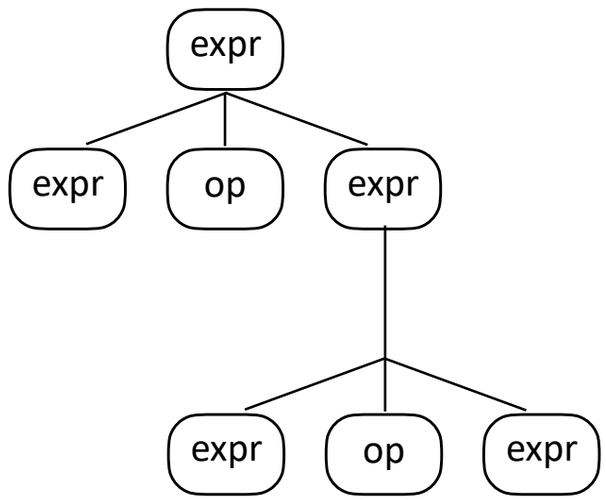
A less cooperative grammar

```
1.  <expr> ::== <expr> <op> <expr>
2.         ::== num
3.  <op>   ::== ÷
4.         ::== -
```

input tokens: `8 - 4 ÷ 2`

Right-most derivation

<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>



Ambiguous Context-Free Grammars

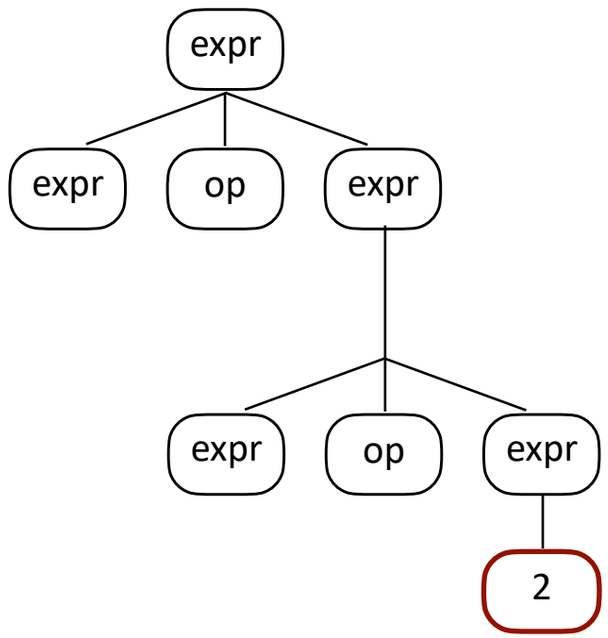
A less cooperative grammar

```
1.  <expr>  ::== <expr> <op> <expr>
2.          ::== num
3.  <op>    ::== ÷
4.          ::== -
```

input tokens: `8 - 4 ÷ 2`

Right-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
<expr> <op> <expr> <op> [num,2]
```



Ambiguous Context-Free Grammars

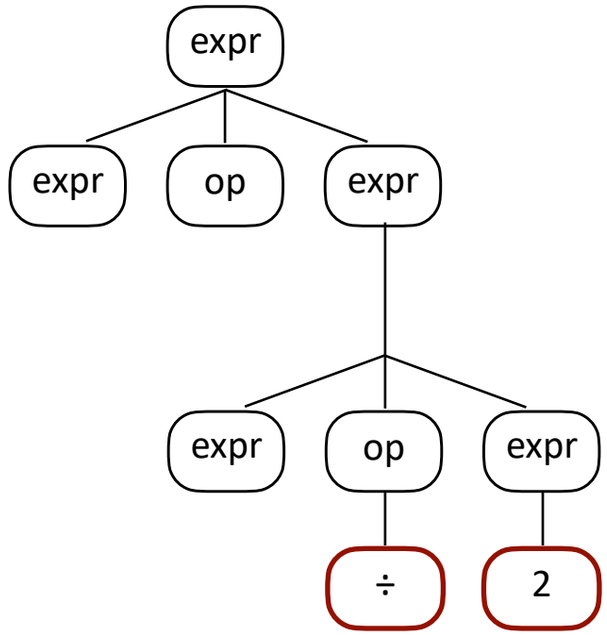
A less cooperative grammar

```
1.  <expr>  ::== <expr> <op> <expr>
2.          ::== num
3.  <op>    ::== ÷
4.          ::== -
```

input tokens: `8 - 4 ÷ 2`

Right-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
<expr> <op> <expr> <op> [num,2]
<expr> <op> <expr>   ÷   [num,2]
```



Ambiguous Context-Free Grammars

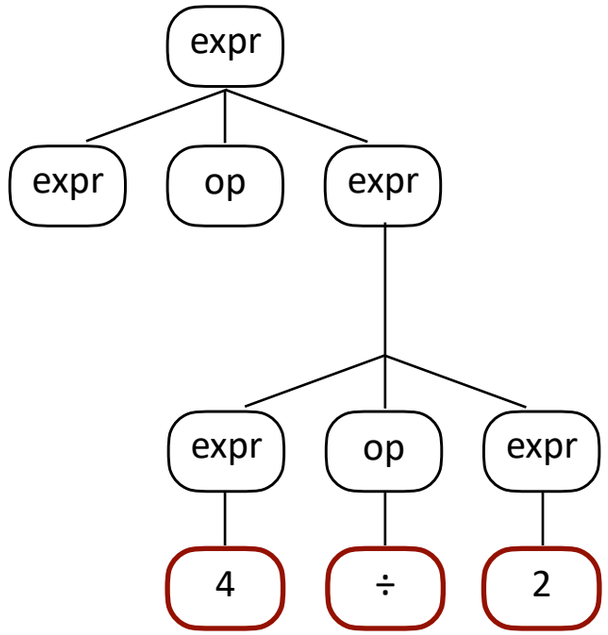
A less cooperative grammar

```
1.  <expr>  ::== <expr> <op> <expr>
2.          ::== num
3.  <op>    ::== ÷
4.          ::== -
```

input tokens: `8 - 4 ÷ 2`

Right-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
<expr> <op> <expr> <op> [num,2]
<expr> <op> <expr>   ÷   [num,2]
<expr> <op> [num,4]  ÷   [num,2]
```



Ambiguous Context-Free Grammars

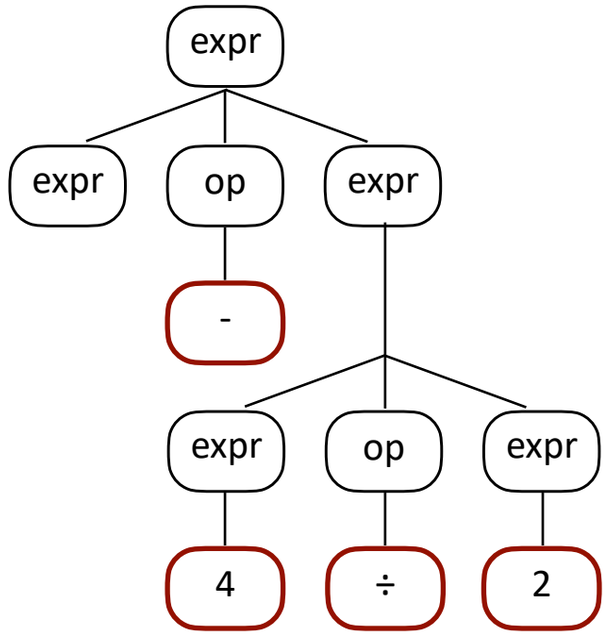
A less cooperative grammar

```
1.  <expr>  ::== <expr> <op> <expr>
2.          ::== num
3.  <op>    ::== ÷
4.          ::== -
```

input tokens: `8 - 4 ÷ 2`

Right-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
<expr> <op> <expr> <op> [num,2]
<expr> <op> <expr>   ÷   [num,2]
<expr> <op> [num,4]  ÷   [num,2]
<expr>   -  [num,4]  ÷   [num,2]
```



Ambiguous Context-Free Grammars

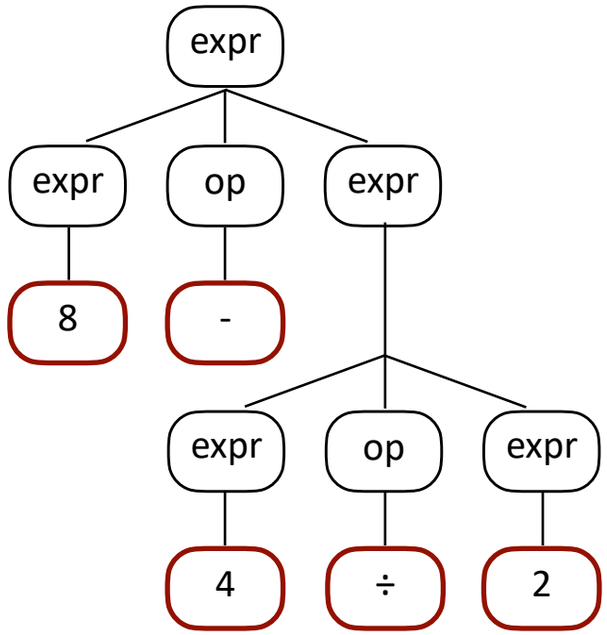
A less cooperative grammar

```
1.  <expr>  ::== <expr> <op> <expr>
2.          ::== num
3.  <op>    ::== ÷
4.          ::== -
```

input tokens: `8 - 4 ÷ 2`

Right-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
<expr> <op> <expr> <op> [num,2]
<expr> <op> <expr>   ÷   [num,2]
<expr> <op> [num,4]  ÷   [num,2]
<expr>   -  [num,4]  ÷   [num,2]
[num,8]  -  [num,4]  ÷   [num,2]
```



Ambiguous Context-Free Grammars

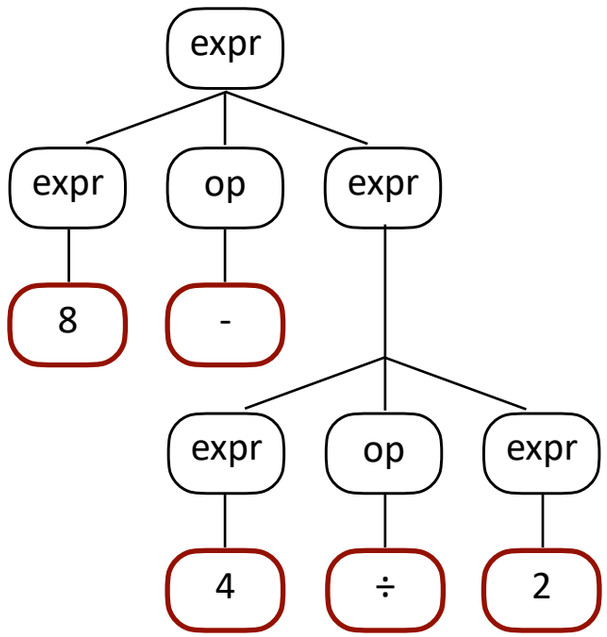
A less cooperative grammar

```
1. <expr> ::== <expr> <op> <expr>
2.        ::== num
3. <op>   ::== ÷
4.        ::== -
```

input tokens: `8 - 4 ÷ 2`

Right-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
<expr> <op> <expr> <op> [num,2]
<expr> <op> <expr>   ÷   [num,2]
<expr> <op> [num,4]  ÷   [num,2]
<expr>   -  [num,4]  ÷   [num,2]
[num,8]  -  [num,4]  ÷   [num,2]
```



A depth-first in-order traversal of a CST where we print the leaf nodes as we go will yield 8 - 4 ÷ 2.

The shape of the CST determines the grouping.

Ambiguous Context-Free Grammars

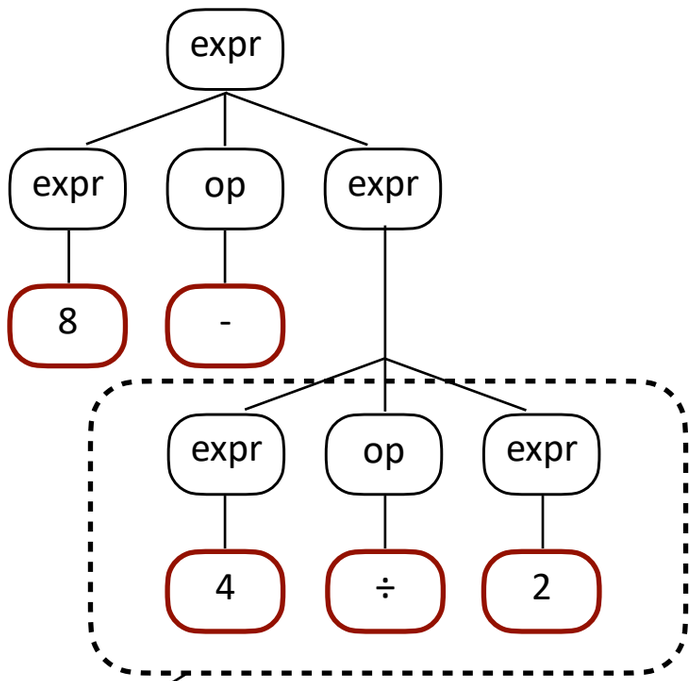
A less cooperative grammar

```
1.  <expr>  ::== <expr> <op> <expr>
2.          ::== num
3.  <op>    ::== ÷
4.          ::== -
```

input tokens: `8 - 4 ÷ 2`

Right-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
<expr> <op> <expr> <op> [num,2]
<expr> <op> <expr>   ÷   [num,2]
<expr> <op> [num,4]  ÷   [num,2]
<expr>   -  [num,4]  ÷   [num,2]
[num,8]  -  [num,4]  ÷   [num,2]
```



Because of the shape of the tree: 8 - (4 ÷ 2)

Ambiguous Context-Free Grammars

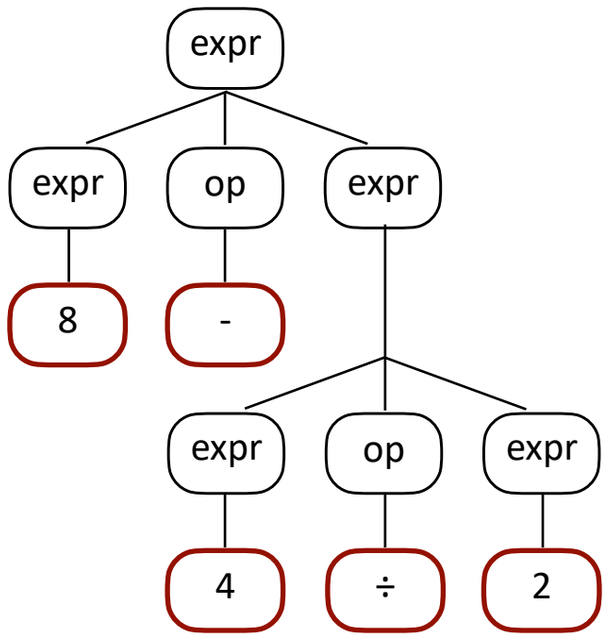
A less cooperative grammar

```
1. <expr> ::== <expr> <op> <expr>
2.        ::== num
3. <op>   ::== ÷
4.        ::== -
```

input tokens: `8 - 4 ÷ 2`

Right-most derivation

```
<expr>
<expr> <op> <expr>
<expr> <op> <expr> <op> <expr>
<expr> <op> <expr> <op> [num,2]
<expr> <op> <expr>   ÷   [num,2]
<expr> <op> [num,4]  ÷   [num,2]
<expr>   -  [num,4]  ÷   [num,2]
[num,8]  -  [num,4]  ÷   [num,2]
```



Because of the shape of the tree: 8 - (4 ÷ 2) = 6, which is correct. Yay, math!

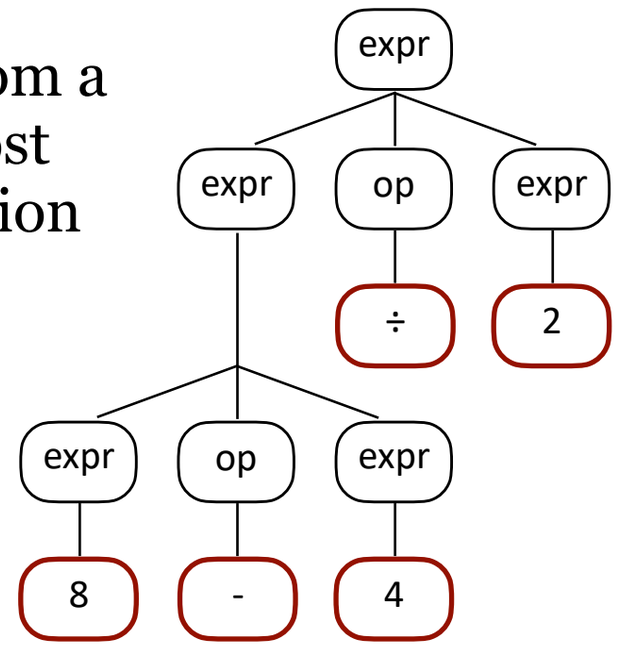
Ambiguous Context-Free Grammars

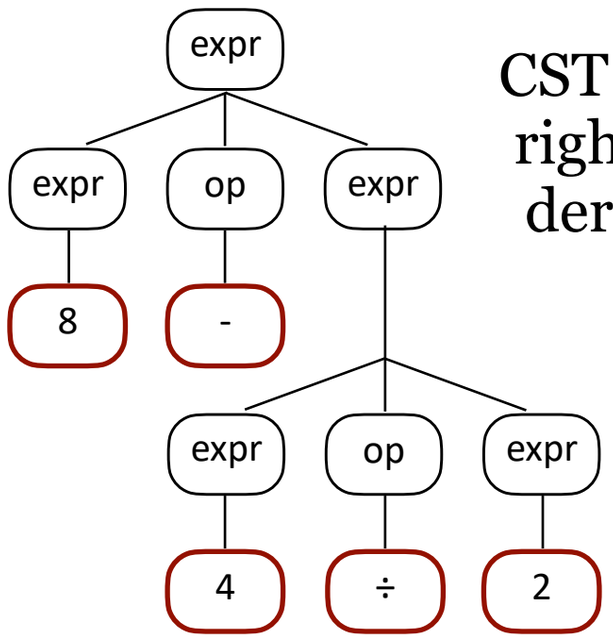
A less cooperative grammar

```
1. <expr> ::== <expr> <op> <expr>
2.        ::== num
3. <op>   ::== ÷
4.        ::== -
```

input tokens: `8 - 4 ÷ 2`

CST from a left-most derivation





CST from a right-most derivation

This is what happens with ambiguous grammars. Different derivations can lead to different parse trees, which leads to different meanings of the program, which leads to suffering.

~~Ambiguous~~ Context-Free Grammars

A more cooperative grammar

```
1. <expr> ::== <expr> - <term>
2.        ::== <term>
3. <term> ::== <term> ÷ <num>
4.        ::== <num>
5. <num>  ::== 0|1|2 ··· |9
```

Context-Free Grammars

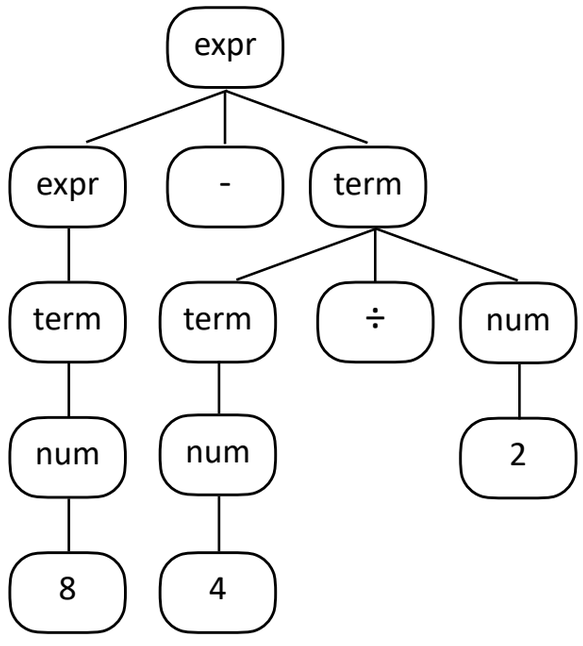
A more cooperative grammar

```
1. <expr> ::== <expr> - <term>
2.        ::== <term>
3. <term> ::== <term> ÷ <num>
4.        ::== <num>
5. <num>  ::== 0|1|2 ··· |9
```

input tokens: `8 - 4 ÷ 2`

Left-most derivation

```
<expr>
<expr> - <term>
<term> - <term>
<num>  - <term>
  8    - <term>
  8    - <term> ÷ <num>
  8    - <num>  ÷ <num>
  8    -   4    ÷ <num>
  8    -   4    ÷   2
```



Context-Free Grammars

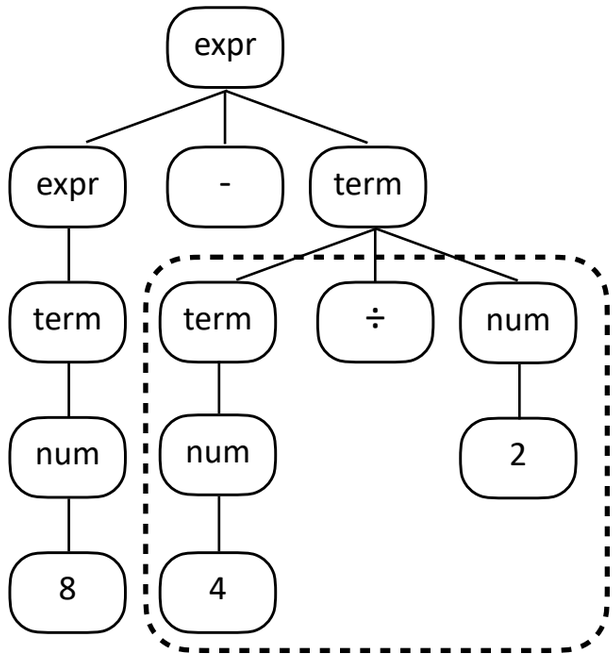
A more cooperative grammar

```
1. <expr> ::== <expr> - <term>
2.        ::== <term>
3. <term> ::== <term> ÷ <num>
4.        ::== <num>
5. <num>  ::== 0|1|2 ··· |9
```

input tokens: `8 - 4 ÷ 2`

Left-most derivation

```
<expr>
<expr> - <term>
<term> - <term>
<num>  - <term>
  8    - <term>
  8    - <term> ÷ <num>
  8    - <num>  ÷ <num>
  8    -   4    ÷ <num>
  8    -   4    ÷   2
```



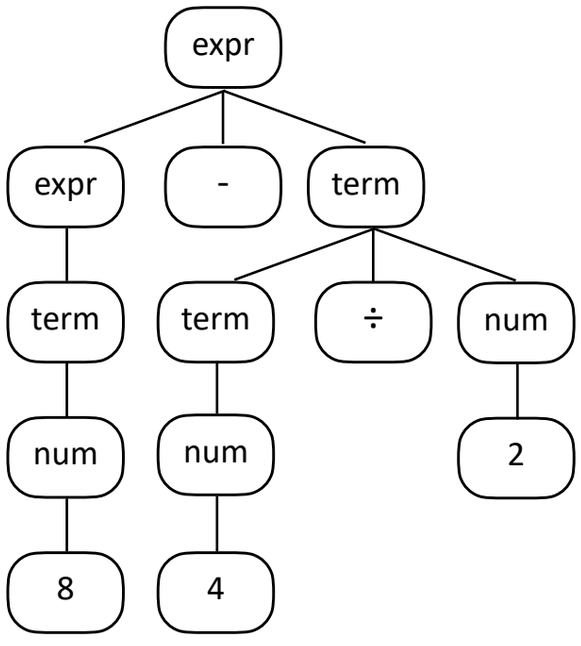
output: `8 - (4 ÷ 2)`

Context-Free Grammars

A more cooperative grammar

```
1. <expr> ::== <expr> - <term>
2.        ::== <term>
3. <term> ::== <term> ÷ <num>
4.        ::== <num>
5. <num>  ::== 0|1|2 ··· |9
```

input tokens: `8 - 4 ÷ 2`



Right-most derivation

```
<expr>
<expr> - <term>
<expr> - <term> ÷ <num>
<expr> - <term> ÷   2
<expr> - <term> ÷   2
<expr> - <num>  ÷   2
<expr> -   4    ÷   2
<expr> -   4    ÷   2
<term> -   4    ÷   2
<num>  -   4    ÷   2
  8    -   4    ÷   2
```

Context-Free Grammars

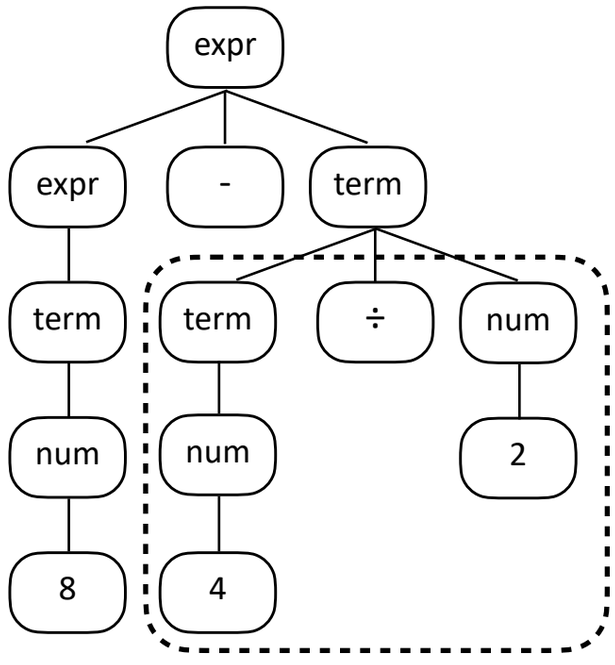
A more cooperative grammar

```
1. <expr> ::== <expr> - <term>
2.        ::== <term>
3. <term> ::== <term> ÷ <num>
4.        ::== <num>
5. <num>  ::== 0|1|2 ··· |9
```

input tokens: `8 - 4 ÷ 2`

Right-most derivation

```
<expr>
<expr> - <term>
<expr> - <term> ÷ <num>
<expr> - <term> ÷   2
<expr> - <term> ÷   2
<expr> - <num>  ÷   2
<expr> -   4    ÷   2
<expr> -   4    ÷   2
<term> -   4    ÷   2
<num>  -   4    ÷   2
  8    -   4    ÷   2
```



output: `8 - (4 ÷ 2)`

Context-Free Grammars

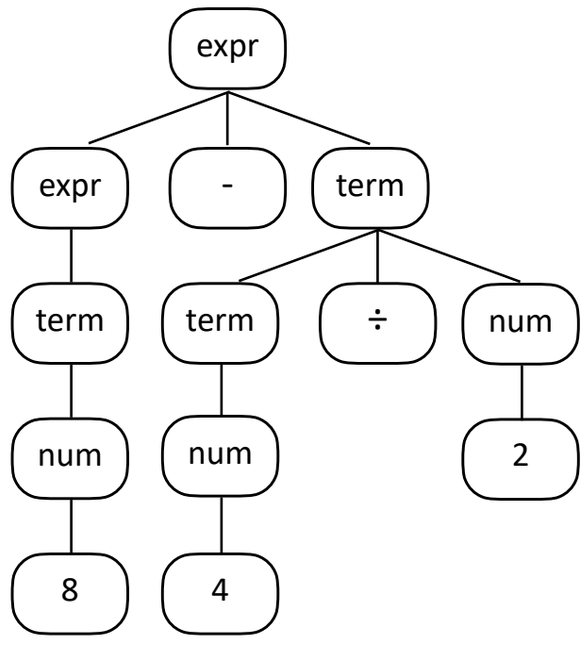
A very cooperative grammar

```
1. <expr> ::== <expr> - <term>
2.        ::== <term>
3. <term> ::== <term> ÷ <num>
4.        ::== <num>
5. <num>  ::== 0|1|2 ··· |9
```

input tokens: `8 - 4 ÷ 2`

Left-most derivation

```
<expr>
<expr> - <term>
<term> - <term>
<num>  - <term>
  8    - <term>
  8    - <term> ÷ <num>
  8    - <num>  ÷ <num>
  8    -   4    ÷ <num>
  8    -   4    ÷   2
```



Right-most derivation

```
<expr>
<expr> - <term>
<expr> - <term> ÷ <num>
<expr> - <term> ÷   2
<expr> - <term> ÷   2
<expr> - <num>  ÷   2
<expr> -   4    ÷   2
<expr> -   4    ÷   2
<term> -   4    ÷   2
<num>  -   4    ÷   2
  8    -   4    ÷   2
```

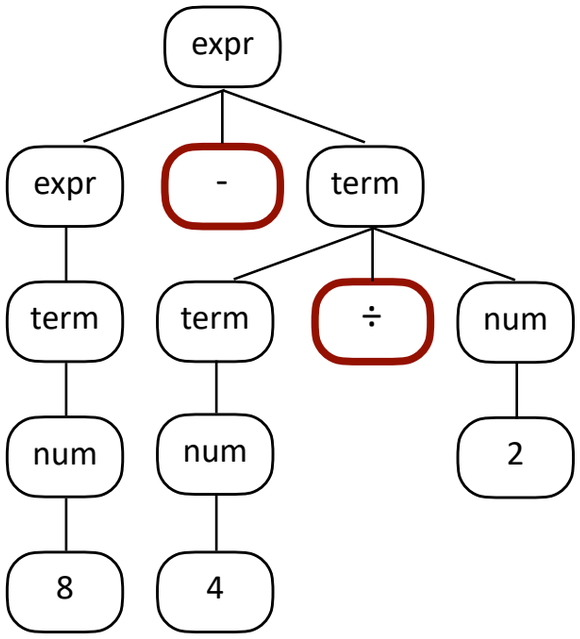
Same CST because
the grammar is unambiguous.

Context-Free Grammars

A note about operator precedence.

```
1.  <expr> ::== <expr> - <term>
2.         ::== <term>
3.  <term> ::== <term> ÷ <num>
4.         ::== <num>
5.  <num>  ::== 0|1|2 ··· |9
```

input tokens: `8 - 4 ÷ 2`



Note:
The **higher** the operator precedence, the **lower** it goes in the syntax tree.

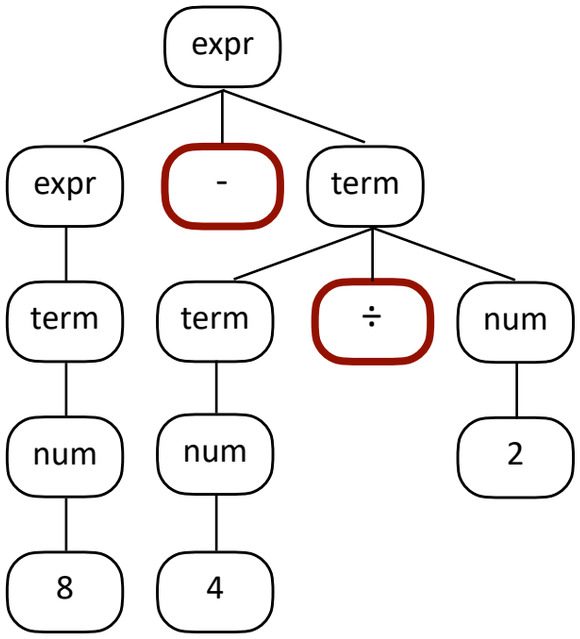
Why is this?

Context-Free Grammars

A note about operator precedence.

```
1.  <expr> ::== <expr> - <term>
2.         ::== <term>
3.  <term> ::== <term> ÷ <num>
4.         ::== <num>
5.  <num>  ::== 0|1|2 ··· |9
```

input tokens: `8 - 4 ÷ 2`



Note:
The **higher** the operator precedence, the **lower** it goes in the syntax tree.

This is because of the grammar.

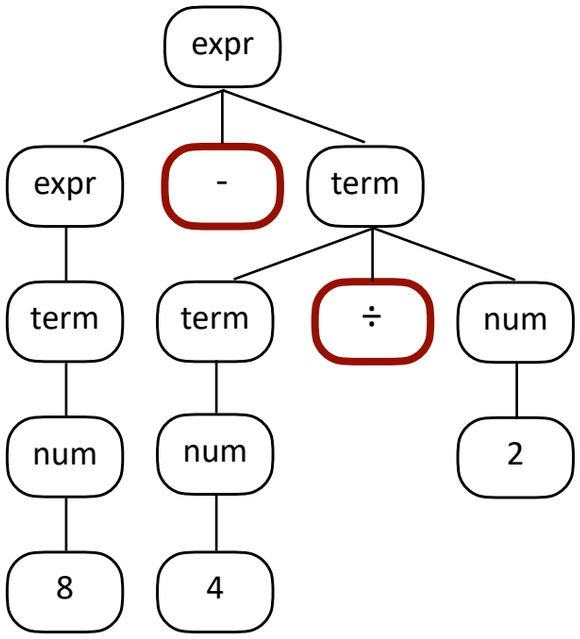
In this example, if there is subtraction, it has to be processed before division because we can only divide **<term>**s so we'll have to go through production #1 to get there, thus placing minus higher in the derivation — and therefore higher in the syntax tree — than division.

Context-Free Grammars

A note about operator precedence.

```
1. <expr> ::== <expr> - <term>
2.        ::== <term>
3. <term> ::== <term> ÷ <num>
4.        ::== <num>
5. <num>  ::== 0|1|2 ··· |9
```

input tokens: `8 - 4 ÷ 2`



Note:
The **higher** the operator precedence, the **lower** it goes in the syntax tree.

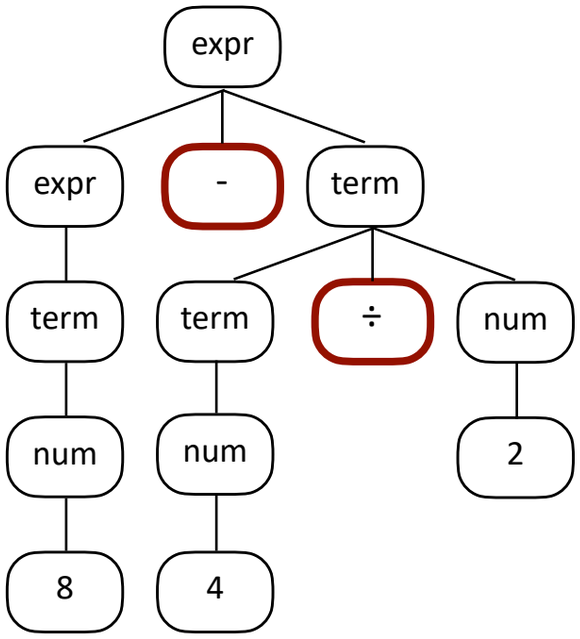
Why do we want this?

Context-Free Grammars

A note about operator precedence.

```
1. <expr> ::== <expr> - <term>
2.        ::== <term>
3. <term> ::== <term> ÷ <num>
4.        ::== <num>
5. <num>  ::== 0|1|2 ··· |9
```

input tokens: `8 - 4 ÷ 2`



Note:
The **higher** the operator precedence, the **lower** it goes in the syntax tree.

We want this because we're going to process our CSTs with depth-first in-order traversals. Since it's **depth-first**, we'll get to and process operators **lower** on the tree before the we get to and process operators higher up the tree.