Bear with me

In defence of ambiguity in Invisible XML



Dr. Bethan Tovey-Walsh

bytheway@linguacelta.com

Preliminaries

Preliminaries

- I do not consent to having my words, voice, or likeness provided to any application or system which may process them using generative AI or an LLM, for any purpose;
  I do not consent to having my words, voice, or likeness used to train AI models of any kind.
- Download slides at:

linguacelta.com/papers/DA2026.pdf

1. Bear with me:

What is ambiguity?

1. Bear with me

bear

1. noun

A member of the family Ursidae.
A large mammal with thick fur,
a small tail, and flat feet.



1. Bear with me

bear

1. noun

A member of the family Ursidae.
A large mammal with thick fur,
a small tail, and flat feet.

2. verb

To carry, support, or endure.
In the phrase “bear with”: to be patient.



1. Bear with me

- *Some strings appear to be inherently ambiguous*

1. Bear with me

- *Some strings appear to be inherently ambiguous*

12/11/2026

1. Bear with me

- *Some strings appear to be inherently ambiguous*

12/11/2026



1. Bear with me

- *Some strings appear to be inherently ambiguous*

12/11/2026

```
date: euro | us.
euro: d, "/", m, "/", y.
us: m, "/", d, "/", y.
d: "0", ["1"-"9"] |
   ["012"], ["0"-"9"] | "3", ["01"].
m: "0", ["1"-"9"] | "1", ["012"].
y: "20", ["012"], ["0"-"9"].
```

```
<date>
   <euro>
      <d>12</d>/
      <m>11</m>/
      <y>2026</y>
   </euro>
</date>
```

```
<date>
   <us>
      <m>12</m>/
      <d>11</d>/
      <y>2026</y>
   </us>
</date>
```

1. Bear with me

- *Some strings appear to be inherently ambiguous*
- *An unambiguous grammar could parse the same input*

```
date: euro | us | euro_us.
euro: d, "/", m, "/", y.
us: m, "/", d, "/", y.
euro_us: m, "/", m, "/", y.
d: "1", ["3"-"9"] | "2", ["0"-"9"] | "3", ["01"].
m: "0", ["1"-"9"] | "1", ["012"].
y: "20", ["012"], ["0"-"9"].
```

1. Bear with me

- *Some strings appear to be inherently ambiguous*
- *An unambiguous grammar could parse the same input*
- *Yet the ambiguous result may feel more semantically faithful*

```
<date>
   <euro>
      <d>12</d>/
      <m>11</m>/
      <y>2026</y>
   </euro>
</date>
```

12/11/2026

```
<date>
   <us>
      <m>12</m>/
      <d>11</d>/
      <y>2026</y>
   </us>
</date>
```

```
<date>
   <euro_us>
      <m>12</m>/
      <m>11</m>/
      <y>2026</y>
   </euro_us>
</date>
```

1. Bear with me

- *Some strings appear to be inherently ambiguous*
- *An unambiguous grammar could parse the same input*
- *Yet the ambiguous result may feel more semantically faithful*
- *Formally, the grammar needn't be ambiguous*

1. Bear with me

- *Some strings appear to be inherently ambiguous*
- *An unambiguous grammar could parse the same input*
- *Yet the ambiguous result may feel more semantically faithful*
- *Formally, the grammar needn't be ambiguous*
- *Practically, ambiguity may be our best choice*

2. Formally Speaking:

Grammars, Languages, & Sentences

2. Formally speaking

Let's talk about three things:

grammars

languages

sentences

2. Formally speaking

- *A* **grammar** *is an instruction manual for making strings*



2. Formally speaking

- *A* **grammar** *is an instruction manual for making strings*
- *A* **language** *is the set of* **all** *strings that can be made using the grammar*





2. Formally speaking

- *A* **grammar** *is an instruction manual for making strings*
- *A* **language** *is the set of* **all** *strings that can be made using the grammar*
- *A* **sentence** *is any* **one** *of the strings belonging to the language*





2. Formally speaking

- *A* **grammar** *is an instruction manual for making strings*
- *A* **language** *is the set of* **all** *strings that can be made using the grammar*
- *A* **sentence** *is any* **one** *of the strings belonging to the language*

```
date: euro | us.
euro: d, "/", m, "/", y.
us: m, "/", d, "/", y.
d: "0", ["1"-"9"] |
   ["012"], ["0"-"9"] | "3", ["01"].
m: "0", ["1"-"9"] | "1", ["012"].
y: "20", ["012"], ["0"-"9"].
```

grammar

sentence ->

01/01/2000
12/11/2026
06/09/2011
26/02/2016
05/29/2003
17/12/2024
[...]

language

2. Formally speaking

- *An IXML processor accepts an input, and functions like:*

    - *a* **recognizer**

    - *a* **parser**

    - *a* **transducer**

2. Formally speaking

- *A* **recognizer** *asks whether the input belongs to the language defined by the grammar*



2. Formally speaking

- *A* **recognizer** *asks whether the input belongs to the language defined by the grammar*

12/11/2026

01/01/2000
12/11/2026
06/09/2011
26/02/2016
05/29/2003
17/12/2024
[...]



2. Formally speaking

- *A* **recognizer** *asks whether the input belongs to the language defined by the grammar*

51/11/2026

01/01/2000
12/11/2026
06/09/2011
26/02/2016
05/29/2003
17/12/2024
[...]



2. Formally speaking

- *A* **recognizer** *asks whether the input belongs to the language defined by the grammar*
- *A* **parser** *asks how the input is constructed from the rules in the grammar*



2. Formally speaking

- *A* **recognizer** *asks whether the input belongs to the language defined by the grammar*
- *A* **parser** *asks how the input is constructed from the rules in the grammar*

13/11/2026




```
<date>
   <euro>
      <d>13</d>/
      <m>11</m>/
      <y>2026</y>
   </euro>
</date>
```

2. Formally speaking

- *A* **recognizer** *asks whether the input belongs to the language defined by the grammar*
- *A* **parser** *asks how the input is constructed from the rules in the grammar*
- *A* **transducer** *asks how to translate the input into the output*



2. Formally speaking

- *A* **recognizer** *asks whether the input belongs to the language defined by the grammar*
- *A* **parser** *asks how the input is constructed from the rules in the grammar*
- *A* **transducer** *asks how to translate the input into the output*

```
sentence: person, " ", feeling, " ", thing.
person: -"Ik", +"I".
feeling: verb, " ", preposition.
verb: -"hou", +"love".
preposition: -"van".
thing: -"katten", +"cats" |
       -"konijntjes", +"bunnies" |
       -"beren", +"bears".
```



2. Formally speaking

- *A* **recognizer** *asks whether the input belongs to the language defined by the grammar*
- *A* **parser** *asks how the input is constructed from the rules in the grammar*
- *A* **transducer** *asks how to translate the input into the output*

```
sentence: person, " ", feeling, " ", thing.
person: -"Ik", +"I".
feeling: verb, " ", preposition.
verb: -"hou", +"love".
preposition: -"van".
thing: -"katten", +"cats" |
       -"konijntjes", +"bunnies" |
```

Ik hou van katten
Ik hou van konijntjes
Ik hou van beren



I love cats
I love bunnies
I love bears

2. Formally speaking



2. Formally speaking

Ambiguity occurs when there are multiple correct answers to this question.



2. Formally speaking



Making a bunny rabbit.

Method 1



Make and stuff the body.



Add the floppy ears.



Assemble the tummy and feet...



...and attach them to the body.



Attach the paws.



Finally, add the eyes and nose

Method 2



Make and stuff the body.



Add the floppy ears.



Attach the tummy.



Assemble the feet and paws...



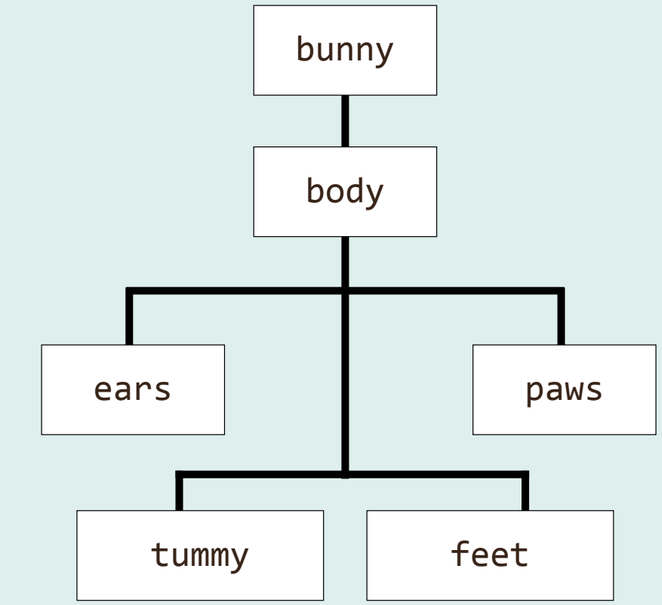
...and attach them to the body.

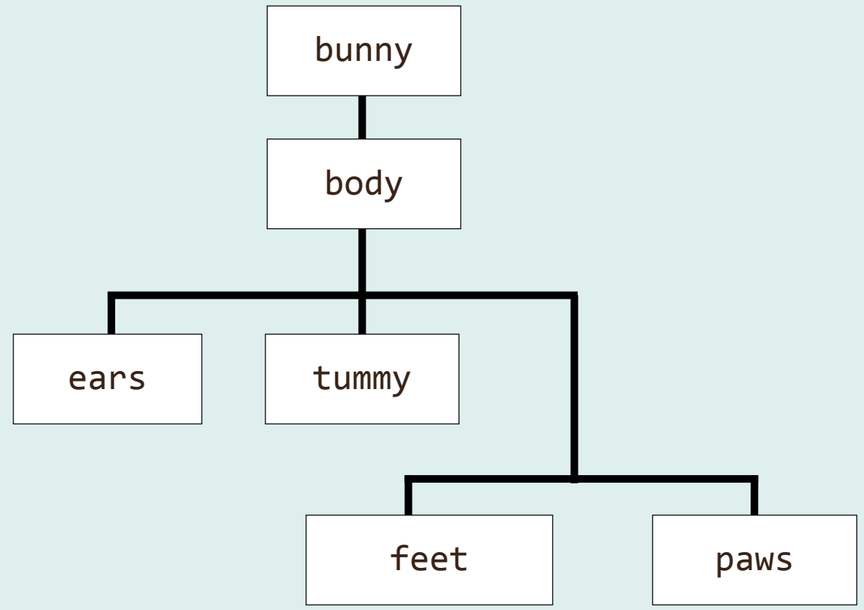


Finally, add the eyes and nose

2. Formally speaking







2. Formally speaking

- **Any** *language can be defined by an* **ambiguous** *grammar*

2. Formally speaking

- **Any** *language can be defined by an* **ambiguous** *grammar*



2. Formally speaking

- **Any** *language can be defined by an* **ambiguous** *grammar*

```
sentence: "bear!".
```



2. Formally speaking

- **Any** *language can be defined by an* **ambiguous** *grammar*

```
sentence: "bear!".
```

```
sentence: noun, "!" | verb.
noun: "bear"? | noun.
verb: "bear", "!".
```



2. Formally speaking

- *Any language can be defined by an* **ambiguous** *grammar*

```
sentence: noun, "!" | verb.
noun: "bear"? | noun.
verb: "bear", "!".
```

```
<sentence>
   <noun>
      <noun>bear</noun>
   </noun>!
</sentence>
```

```
<sentence>
   <verb>bear!</verb>
</sentence>
```

```
<sentence>
   <noun>bear</noun>!
</sentence>
```

2. Formally speaking

- **Any** *language can be defined by an* **ambiguous** *grammar*
- **Not** *every language can be defined by an* **unambiguous** *grammar*

2. Formally speaking

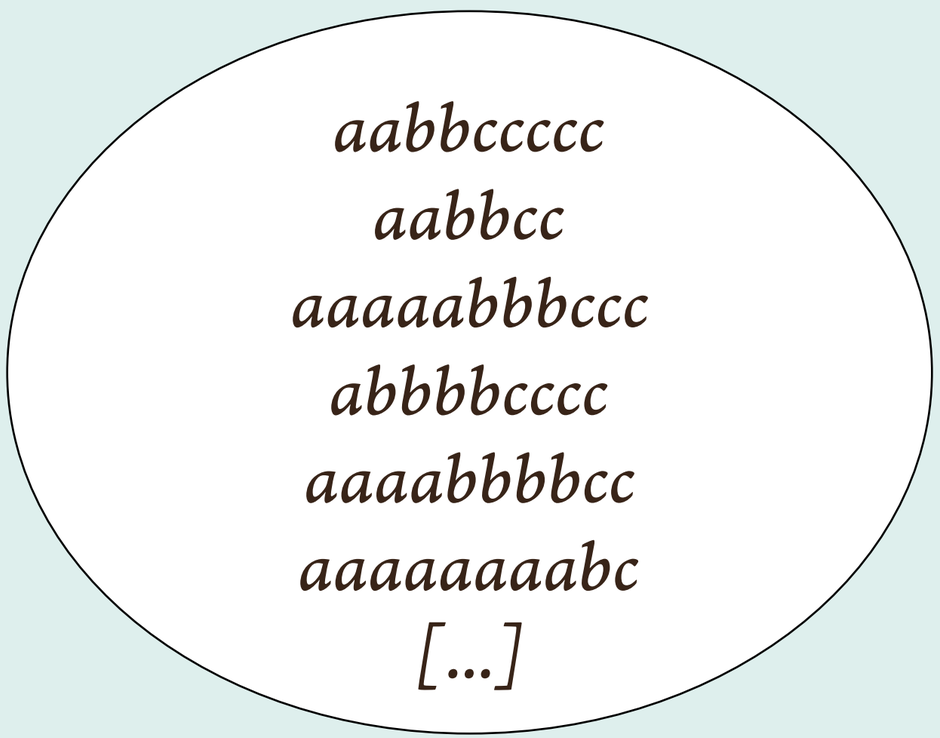
- **Any** *language can be defined by an* **ambiguous** *grammar*
- **Not** *every language can be defined by an* **unambiguous** *grammar*
  - **Inherently ambiguous** *languages have no unambiguous grammars*

2. Formally speaking

- **Any** *language can be defined by an* **ambiguous** *grammar*
- **Not** *every language can be defined by an* **unambiguous** *grammar*
  - **Inherently ambiguous** *languages have no unambiguous grammars*
  - *What does an inherently ambiguous language look like?*

2. Formally speaking

A canonical inherently ambiguous language



2. Formally speaking

A canonical inherently ambiguous language

- *Two alternatives:*
- *An equal number of “a” and “b”, followed by any number of “c”*
- *Any number of “a”, followed by an equal number of “b” and “c”*

```
ABC:
   A_and_B, "c"+ |
   "a"+, B_and_C.
A_and_B: "a", A_and_B?, "b".
B_and_C: "b", B_and_C?, "c".
```

aabbccccc
aabbcc
aaaaabbbccc
abbbbcccc
aaaabbbbcc
aaaaaaaabc
[…]

2. Formally speaking

A canonical inherently ambiguous language

- *Two alternatives:*
- *An equal number of “a” and “b”, followed by any number of “c”*
- *Any number of “a”, followed by an equal number of “b” and “c”*
- *The grammar is ambiguous for strings with the same number of all three items*

```
ABC:
   A_and_B, "c"+ |
   "a"+, B_and_C.
A_and_B: "a", A_and_B?, "b".
B_and_C: "b", B_and_C?, "c".
```

abc
aabbcc
aaabbbccc
aaaabbbbcccc
aaaaabbbbbccccc
[...]

2. Formally speaking

A canonical inherently ambiguous language

- *Two alternatives:*
- *An equal number of "a" and "b", followed by any number of "c"*
- *Any number of "a", followed by an equal number of "b" and "c"*
- *The grammar is ambiguous for strings with the same number of all three items*
- *There is no way to write a (context-free) grammar for just these strings*

abc
aabbcc
aaabbbccc
aaaabbbbcccc
aaaaabbbbbccccc
[...]

2. Formally speaking

A canonical inherently ambiguous language

- *Two alternatives:*
- *An equal number of "a" and "b", followed by any number of "c"*
- *Any number of "a", followed by an equal number of "b" and "c"*
- *The grammar is ambiguous for strings with the same number of all three items*
- *There is no way to write a (context-free) grammar for just these strings*
- *So we can't rewrite the grammar to remove the ambiguity and also preserve the language*

aabbccccc
aabbcc
aaaaabbbccc
abbbbcccc
aaaabbbbcc
aaaaaaaabc
[...]

2. Formally speaking

- *Any* **ambiguous** *grammar may give* **unambiguous** *results for* **some** *sentences*

2. Formally speaking

- *Any* **ambiguous** *grammar may give* **unambiguous** *results for* **some** *sentences*

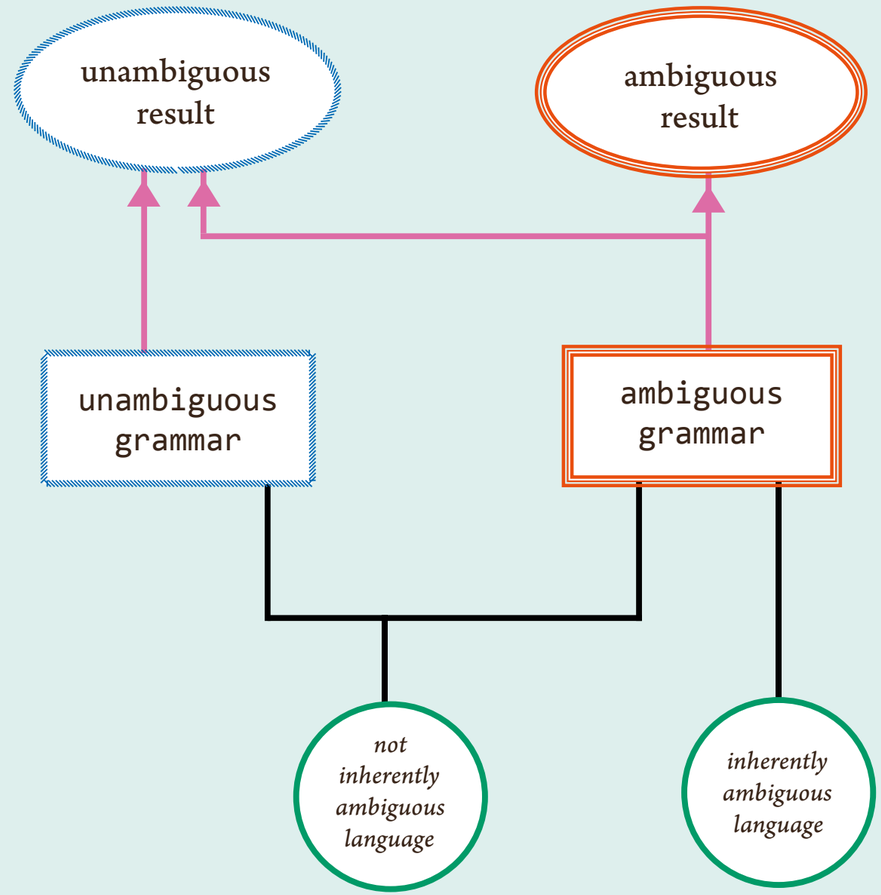
```
ABC:
   A_and_B, "c"+ |
   "a"+, B_and_C.
A_and_B: "a", A_and_B?, "b".
B_and_C: "b", B_and_C?, "c".
```

aabbbccc

```
<ABC>
   aa
   <B_and_C>b
      <B_and_C>b
         <B_and_C>bc</B_and_C>
      c</B_and_C>
   c</B_and_C>
</ABC>
```

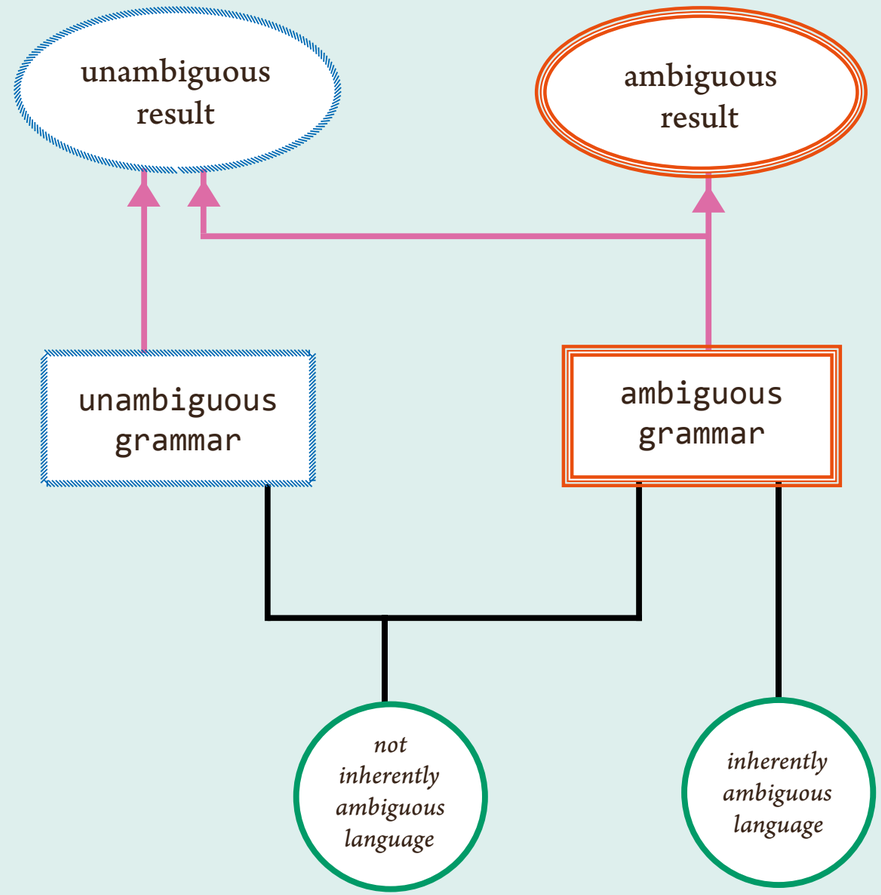
2. Formally speaking

- An ambiguous grammar can be written for either type of language



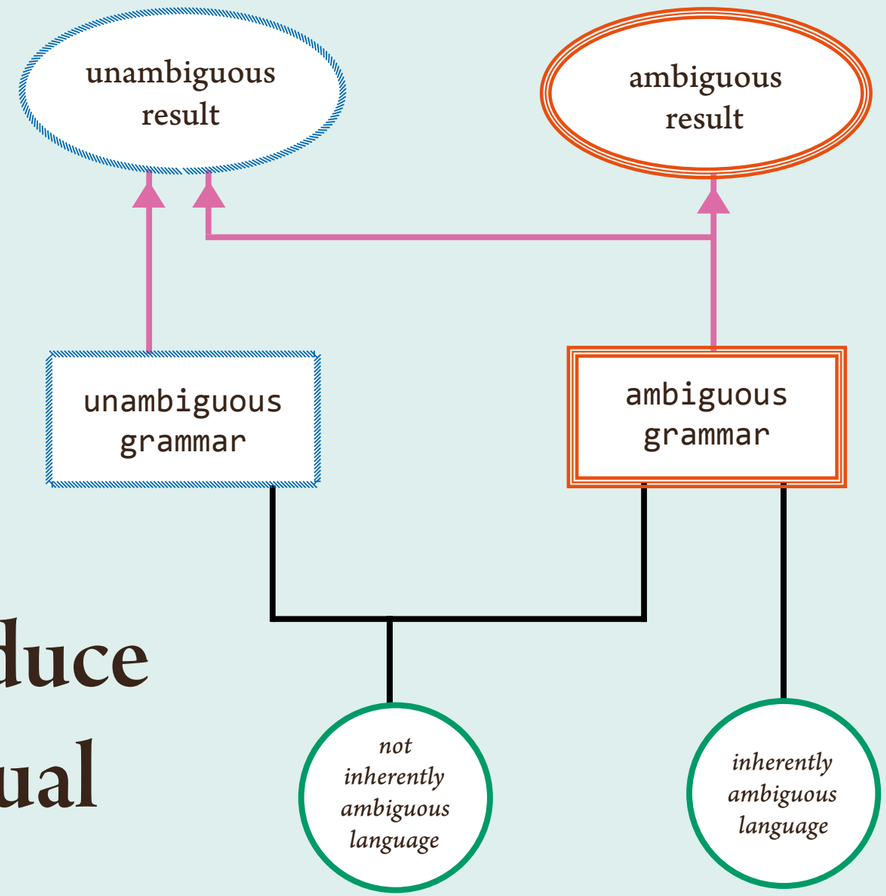
2. Formally speaking

- An ambiguous grammar can be written for either type of language
- An unambiguous result can come from either type of grammar



2. Formally speaking

- An ambiguous grammar can be written for either type of language
- An unambiguous result can come from either type of grammar
- Even an ambiguous grammar for an inherently ambiguous language can produce an unambiguous result for some individual inputs



2. Formally speaking

Formally speaking, an inherently ambiguous language is the only factor which entirely prevents us from writing an unambiguous grammar.

3. Practically Speaking:

The building blocks of an IXML grammar

3. Practically Speaking

We write IXML grammars to do things with data.

3. Practically Speaking

We write IXML grammars to do things with data.

There are two facets to an ideal IXML dataset:

3. Practically Speaking

We write IXML grammars to do things with data.

There are two facets to an ideal IXML dataset:

- a set of strings
- some shared (implicit) structure for those strings

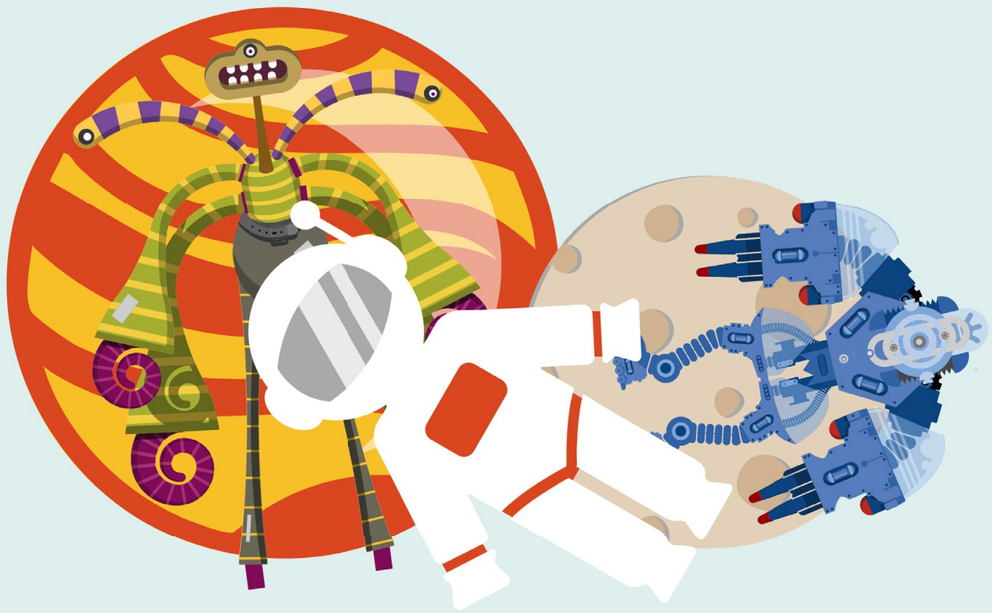
3. Practically Speaking

- *A dataset is like a collection of items that we want to organize*



3. Practically Speaking

- *A dataset is like a collection of items that we want to organize*
- *We look for features which some of the items have in common*





3. Practically Speaking

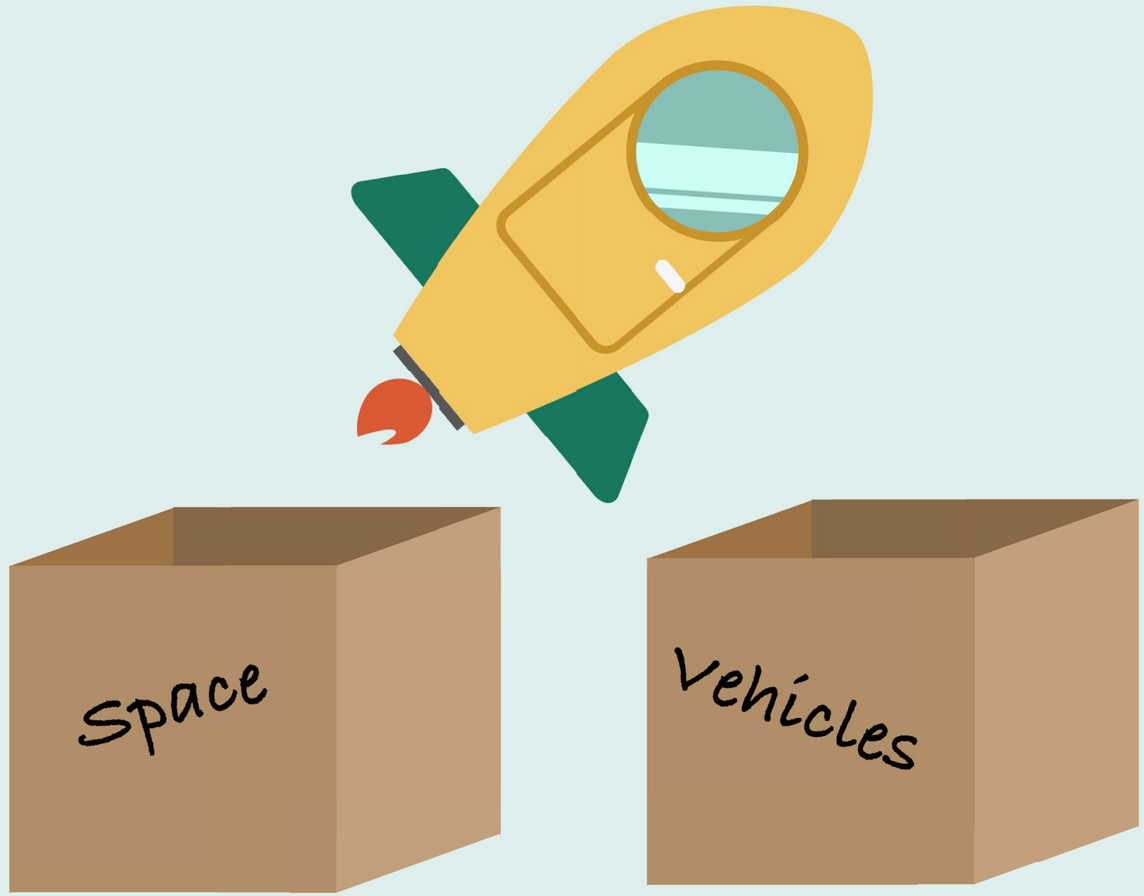
- *A dataset is like a collection of items that we want to organize*
- *We look for features which some of the items have in common*
- *We organize the similar items into boxes*





3. Practically Speaking

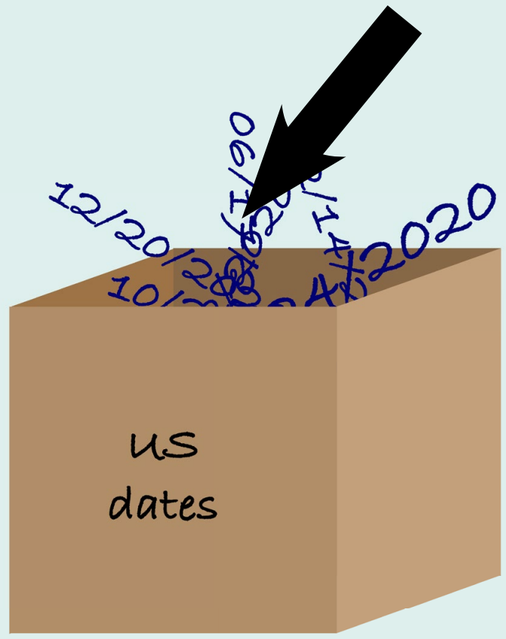
- *There may be items which could fit into more than one box*

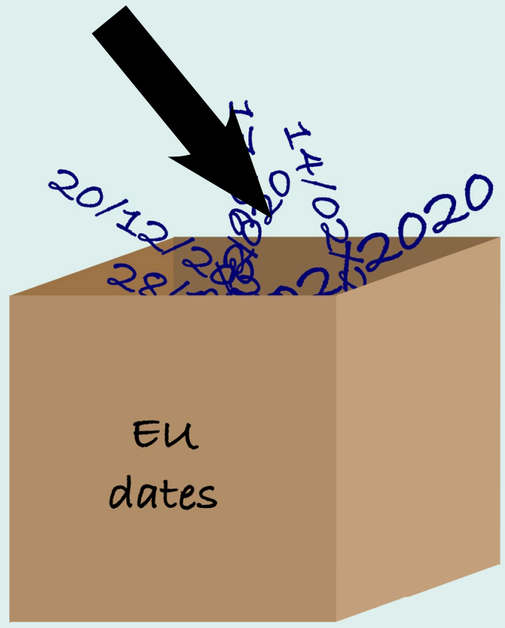


3. Practically Speaking

- *There may be items which could fit into more than one box*

06/06/2026

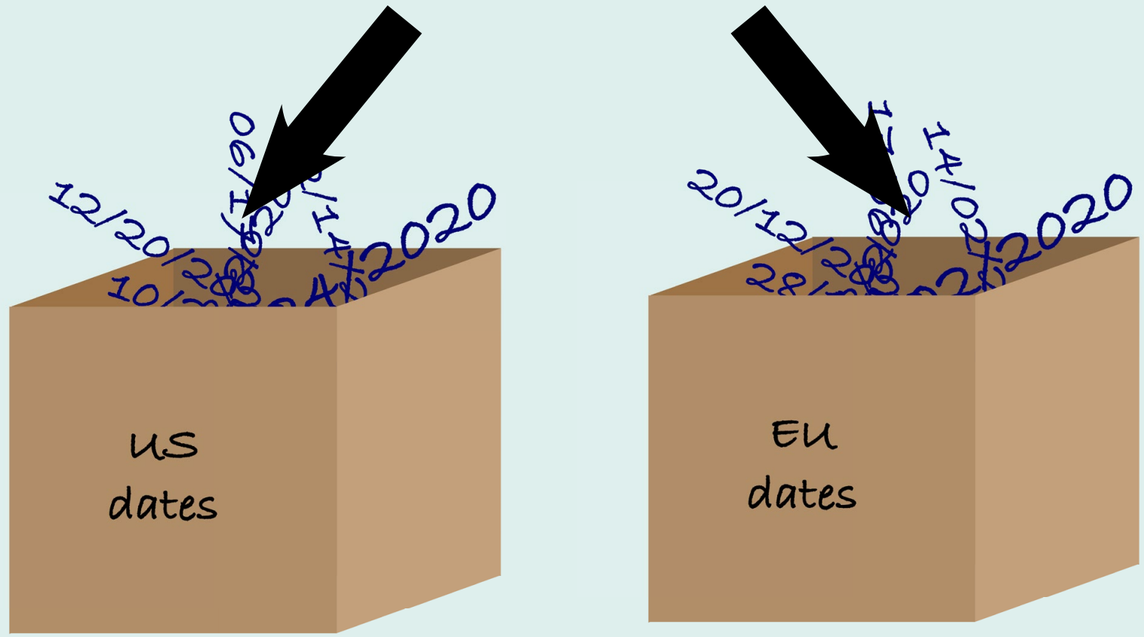




3. Practically Speaking

- *There may be items which could fit into more than one box*
- *Replicating this implicit structure results in an ambiguous grammar*

06/06/2026

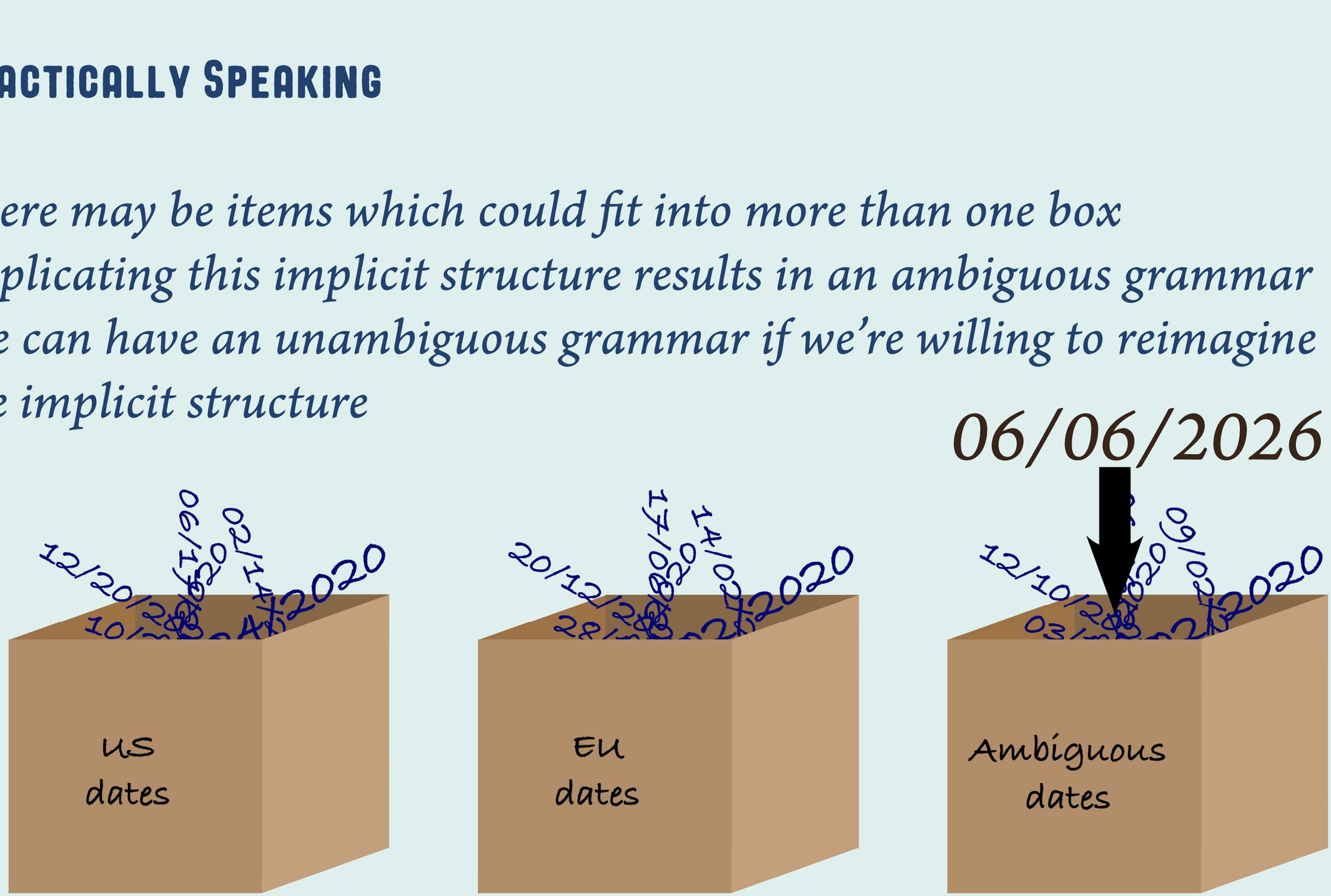


3. Practically Speaking

- *There may be items which could fit into more than one box*
- *Replicating this implicit structure results in an ambiguous grammar*
- *We can have an unambiguous grammar if we're willing to reimagine the implicit structure*

3. Practically Speaking

- *There may be items which could fit into more than one box*
- *Replicating this implicit structure results in an ambiguous grammar*
- *We can have an unambiguous grammar if we're willing to reimagine the implicit structure*



3. Practically Speaking

Our IXML grammar must be able to recognize our dataset

3. Practically Speaking

Our IXML grammar must be able to recognize our dataset

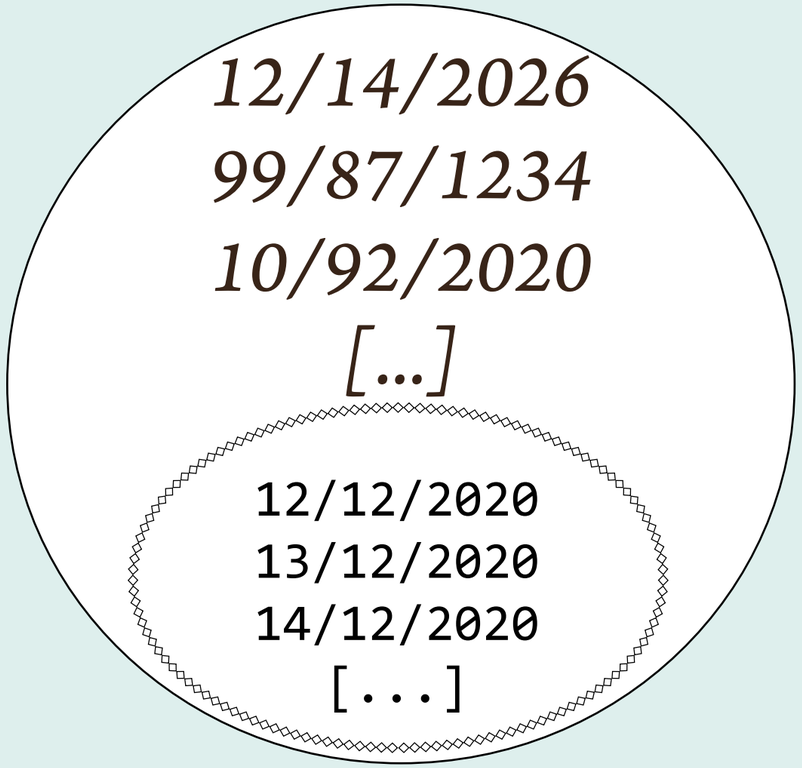
- *the language defined by the grammar* **must** *include the whole dataset*



3. Practically Speaking

Our IXML grammar must be able to recognize our dataset

- *the language defined by the grammar* **must** *include the whole dataset*
- *the language of a permissive grammar* **may** *be larger than the dataset*




```
date: d, "/", m, "/", y.
d: ["0"-"9"], ["0"-"9"].
m: ["0"-"9"], ["0"-"9"].
y: ["0"-"9"], ["0"-"9"], ["0"-"9"],
   ["0"-"9"].
```

3. Practically Speaking

Our IXML grammar must be able to recognize our dataset

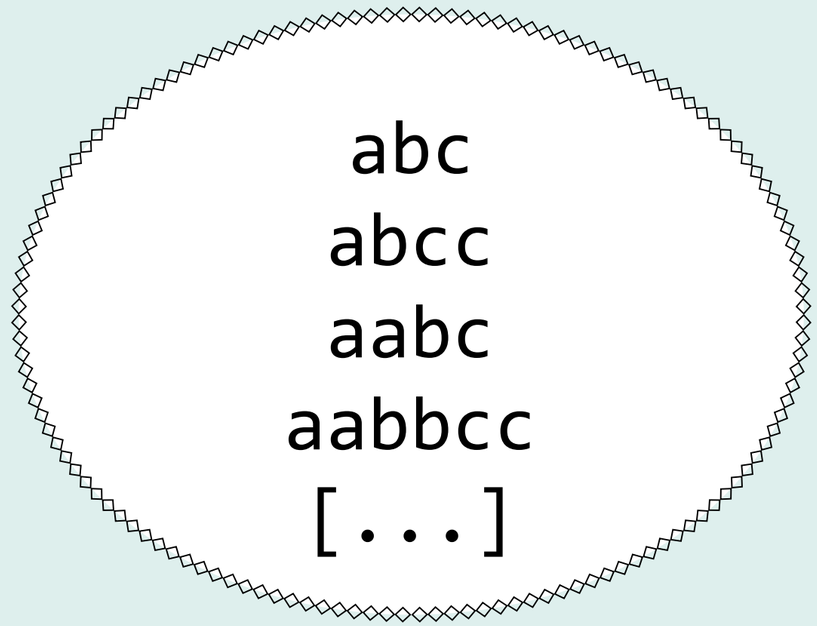
- *the language defined by the grammar* **must** *include the whole dataset*
- *the language of a permissive grammar* **may** *be larger than the dataset*
- *if the language of the dataset is inherently ambiguous, a permissive grammar might still let us avoid ambiguity*

3. Practically Speaking

Our IXML grammar must be able to recognize our dataset

- *the language defined by the grammar* **must** *include the whole dataset*
- *the language of a permissive grammar* **may** *be larger than the dataset*
- *if the language of the dataset is inherently ambiguous, a permissive grammar might still let us avoid ambiguity*

```
ABC:
   A_and_B, "c"+ |
   "a"+, B_and_C.
A_and_B: "a", A_and_B?, "b".
B_and_C: "b", B_and_C?, "c".
```

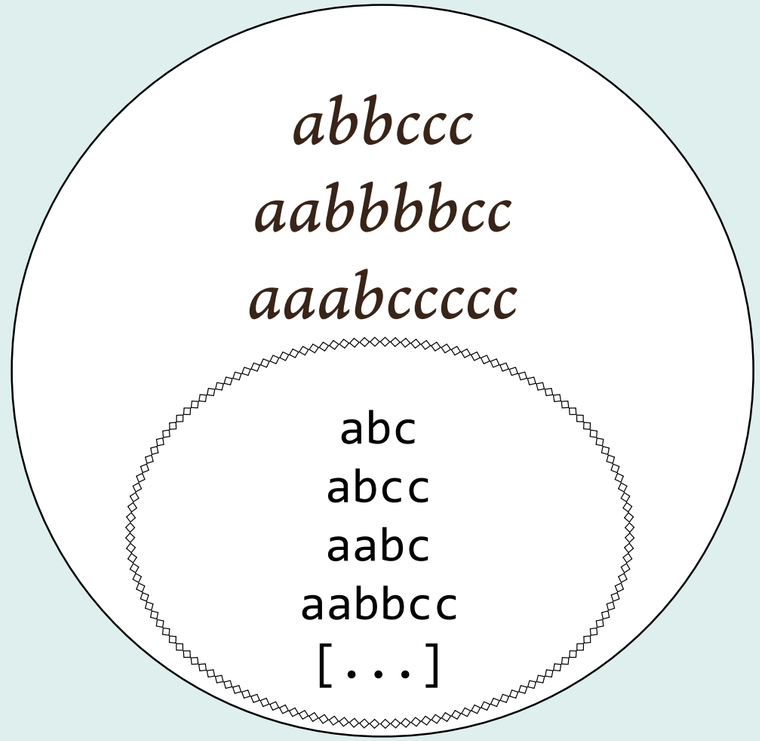


3. Practically Speaking

Our IXML grammar must be able to recognize our dataset

- *the language defined by the grammar* **must** *include the whole dataset*
- *the language of a permissive grammar* **may** *be larger than the dataset*
- *if the language of the dataset is inherently ambiguous, a permissive grammar might still let us avoid ambiguity*

```
ABC: "a"+, "b"+, "c"+.
```



3. Practically Speaking

Our IXML grammar must be able to recognize our dataset

- *the language defined by the grammar* **must** *include the whole dataset*
- *the language of a permissive grammar* **may** *be larger than the dataset*
- *if the language of the dataset is inherently ambiguous, a permissive grammar might still let us avoid ambiguity*
- *but unambiguity means our grammar no longer validates our data*

```
ABC: "a"+, "b"+, "c"+.
```

aaabccccc



3. Practically Speaking

- When we write an IXML grammar, we should consider the relative importance of three things for our use case:

3. Practically Speaking

- When we write an IXML grammar, we should consider the relative importance of three things for our use case:
  - *permissiveness: how precisely does the grammar's language match the dataset?*

3. Practically Speaking

- When we write an IXML grammar, we should consider the relative importance of three things for our use case:
  - *permissiveness: how precisely does the grammar's language match the dataset?*
  - *semantic fidelity: how well does the grammar's structure replicate the dataset's implicit structure?*

3. Practically Speaking

- When we write an IXML grammar, we should consider the relative importance of three things for our use case:
  - *permissiveness: how precisely does the grammar's language match the dataset?*
  - *semantic fidelity: how well does the grammar's structure replicate the dataset's implicit structure?*
  - *unambiguity: does the grammar have a single way to derive all of its sentences?*

3. Practically Speaking

- When we write an IXML grammar, we should consider the relative importance of three things for our use case:
  - *permissiveness: how precisely does the grammar's language match the dataset?*
  - *semantic fidelity: how well does the grammar's structure replicate the dataset's implicit structure?*
  - *unambiguity: does the grammar have a single way to derive all of its sentences?*
- We can't always prioritize all three.

3. Practically Speaking

Practically speaking, an ambiguous grammar
may be the best choice for a particular dataset and use case.

4. Fun with ambiguity:

From party tricks to next-generation markup

4. Fun with ambiguity

Stop! Demo time...



5. Conclusions

5. Conclusions

Why might we want to avoid ambiguity?

5. Conclusions

Why might we want to avoid ambiguity?

- *Unambiguous grammars generally parse more efficiently*

5. Conclusions

Why might we want to avoid ambiguity?

- *Unambiguous grammars generally parse more efficiently*
- *If we prefer or require unambiguous results for further processing*

5. Conclusions

Why might we want to avoid ambiguity?

- *Unambiguous grammars generally parse more efficiently*
- *If we prefer or require unambiguous results for further processing*
- *If we have an unambiguous implicit structure in mind; ambiguity in the grammar means we didn't write what we intended to*

5. Conclusions

Why might we want to avoid ambiguity?

- *Unambiguous grammars generally parse more efficiently*
- *If we prefer or require unambiguous results for further processing*
- *If we have an unambiguous implicit structure in mind; ambiguity in the grammar means we didn't write what we intended to*

Why might we be willing to accept ambiguity?

5. Conclusions

Why might we want to avoid ambiguity?

- *Unambiguous grammars generally parse more efficiently*
- *If we prefer or require unambiguous results for further processing*
- *If we have an unambiguous implicit structure in mind; ambiguity in the grammar means we didn't write what we intended to*

Why might we be willing to accept ambiguity?

- *If we are more concerned about semantic fidelity to the implicit structure*

5. Conclusions

Why might we want to avoid ambiguity?

- *Unambiguous grammars generally parse more efficiently*
- *If we prefer or require unambiguous results for further processing*
- *If we have an unambiguous implicit structure in mind; ambiguity in the grammar means we didn't write what we intended to*

Why might we be willing to accept ambiguity?

- *If we are more concerned about semantic fidelity to the implicit structure*
- *If we are more concerned about validating the input data*

5. Conclusions

Why might we want to avoid ambiguity?

- *Unambiguous grammars generally parse more efficiently*
- *If we prefer or require unambiguous results for further processing*
- *If we have an unambiguous implicit structure in mind; ambiguity in the grammar means we didn't write what we intended to*

Why might we be willing to accept ambiguity?

- *If we are more concerned about semantic fidelity to the implicit structure*
- *If we are more concerned about validating the input data*
- *If we know that the processor will make all outputs identical*

5. Conclusions

Why might we want to avoid ambiguity?

- *Unambiguous grammars generally parse more efficiently*
- *If we prefer or require unambiguous results for further processing*
- *If we have an unambiguous implicit structure in mind; ambiguity in the grammar means we didn't write what we intended to*

Why might we be willing to accept ambiguity?

- *If we are more concerned about semantic fidelity to the implicit structure*
- *If we are more concerned about validating the input data*
- *If we know that the processor will make all outputs identical*

Why might we *want* ambiguity?

5. Conclusions

Why might we want to avoid ambiguity?

- *Unambiguous grammars generally parse more efficiently*
- *If we prefer or require unambiguous results for further processing*
- *If we have an unambiguous implicit structure in mind; ambiguity in the grammar means we didn't write what we intended to*

Why might we be willing to accept ambiguity?

- *If we are more concerned about semantic fidelity to the implicit structure*
- *If we are more concerned about validating the input data*
- *If we know that the processor will make all outputs identical*

Why might we *want* ambiguity?

- *If our data has multiple valid interpretations*

5. Conclusions

Why might we want to avoid ambiguity?

- *Unambiguous grammars generally parse more efficiently*
- *If we prefer or require unambiguous results for further processing*
- *If we have an unambiguous implicit structure in mind; ambiguity in the grammar means we didn't write what we intended to*

Why might we be willing to accept ambiguity?

- *If we are more concerned about semantic fidelity to the implicit structure*
- *If we are more concerned about validating the input data*
- *If we know that the processor will make all outputs identical*

Why might we *want* ambiguity?

- *If our data has multiple valid interpretations*
- *If we want to commit crimes against context-free grammars!*



Be ambiguous.
Do crimes.