

Work In Progress!

# Linguistic Tools for Managing Grammatical Domains

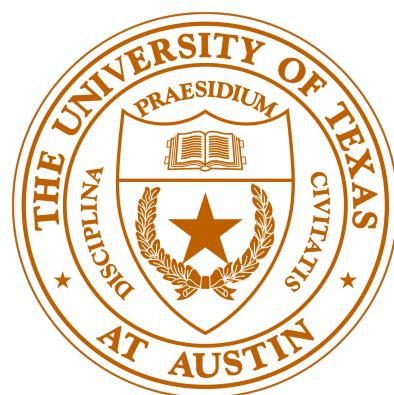
Anders Miltner

Devon Loehr

Arnold Mong

Kathleen Fisher

David Walker



# Let's Talk About Dates



```
Date my_date = new Date("05/26/22");
```

# Bad Dates

[https://en.wikipedia.org/wiki/Time\\_formatting\\_and\\_storage\\_bugs](https://en.wikipedia.org/wiki/Time_formatting_and_storage_bugs)

The image shows a screenshot of the Wikipedia article titled "Time formatting and storage bugs". The page layout includes a left sidebar with navigation links, a main content area with an introductory paragraph, and a table of contents.

**WIKIPEDIA**  
The Free Encyclopedia

Main page  
Contents  
Current events  
Random article  
About Wikipedia  
Contact us  
Donate



Contribute

Help  
Learn to edit  
Community portal  
Recent changes  
Upload file

Tools

What links here  
Related changes  
Special pages  
Permanent link  
Page information  
Cite this page  
Wikidata item

Print/export  
Download as PDF  
Printable version

Languages   
עברית  
Українська 

## Time formatting and storage bugs

From Wikipedia, the free encyclopedia

In computer science, **time formatting and storage bugs** are a class of **software bugs** that may cause **time and date** calculation or display to be improperly handled. These are most commonly manifestations of **arithmetic overflow**, but can also be the result of other issues. The most well-known consequence of bugs of this type is the **Y2K problem**, but many other milestone dates or times exist that have caused or will cause problems depending on various programming deficiencies.

<b>Contents</b> <span>(hide)</span>
1 Year 1975
2 Year 1978
3 Year 1989
4 Year 1997
5 Year 1999
5.1 First GPS rollover
5.2 9/9/99
6 Year 2000
6.1 Two-digit year representations
7 Year 2001
8 Year 2010
9 Year 2011
10 Year 2013
11 Year 2015
12 Year 2019
12.1 Second GPS rollover
12.2 Japanese calendar transition
13 Year 2020
13.1 Classic Mac OS
14 Year 2021
15 Year 2022
16 Year 2025
17 Year 2028
18 Year 2031
19 Year 2032
20 Year 2036
21 Year 2038
21.1 Unix time rollover
21.2 Windows C runtime library
21.3 DVB rollover
21.4 Third GPS rollover
22 Year 2040
23 Year 2042
24 Year 2048
25 Year 2051
26 Year 2061
27 Year 2079
27.1 Days 32,768 and 65,536
28 Year 2080
29 Year 2100
30 Year 2106
31 Year 2108
32 Year 2137
33 Year 2262
34 Years 4000 and 8000
35 Year 4294



**Don't forget to attend LangSec on 26/05/22!  
(26 May 2022)**



**Don't forget to attend LangSec on 26/05/22!  
(2026 May 22)**

# This can be addressed by creating a formal parser!

But there's so many possible parsers we might want to create!

05/26/22  
05/26/2022  
05-26-2022  
5-26-22

05/26/22  
05/26/2022  
5/26/2022  
05-26-2022

# Grammar Induction

Learn a Grammar from some form of specification

- L\* Algorithm [Angluin, 1987]
  - Seminal, but exponential in worst case
- Can we do better?
  - Not unless we break RSA!  
[Angluin and Kharitonov, 1995]
  - SOTA algorithms work via biasing to grammars they think occur in practice

# Our Approach

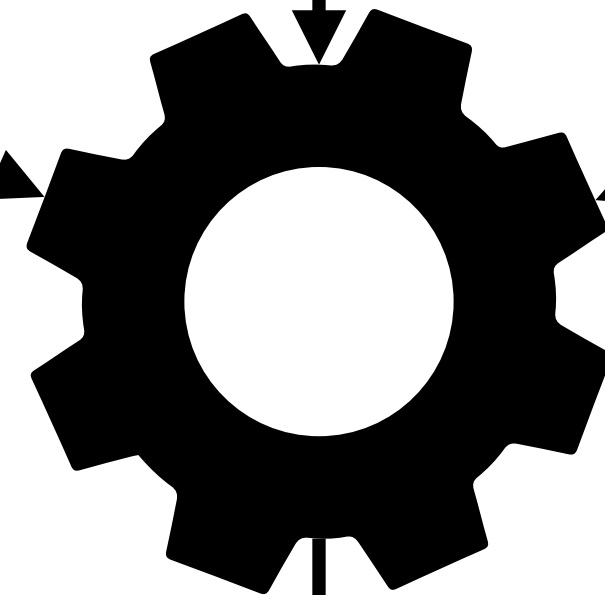
Let users encode the set of possible grammars  
and the biases in grammar generation

# Workflow for Grammar Induction

Metagrammar describing  
the grammatical domain

Set of Positive Examples

Set of Negative Examples



Grammar within the  
grammatical domain



# Benefits

Few Examples Required

Explicitly Encoded Biases

Generated Grammar Guarantees

# Metagrammar by Example

```
1 Sep -> ? "," named COMMA
2       ? "/" named SLASH
3       ? "-" named DASH.
4 constraint(|Production(Sep)| = 1)

6 Digit -> ["0"-"9"].

8 Year -> ? Digit Digit
9         ? Digit Digit Digit Digit.
10 constraint(|Productions(Year)| = 1)

12 Month -> ? Digit
13           ? "0" Digit
14           | "10" | "11" | "12".
15 constraint(|Productions(Month)| = 2)

17 Day -> ? ["1" - "9"]
18         ? "0" ["1" - "9"]
19         | ["1" - "2"] Digit | "30" | "31".
20 constraint(|Productions(Day)| = 2).

22 Date -> ? Day Sep Month Sep Year
23         ? Month Sep Day Sep Year
24         ? Year Sep Month Sep Day
25         ? Year Sep Day Sep Month.
26 constraint(|Productions(Date)| = 1).

28 preference prefer SLASH 2.0.
29 preference prefer DASH 1.0.

31 start Date
```

# Work-In-Progress

- Improving efficiency of grammar induction algorithm
- Improving speed of parsing
- Improving surface language with better syntactic sugar

# Questions

- By focusing on individual domains, grammar induction becomes easier
- Metagrammar files encode possible grammars and grammatical biases for selection
- Future work in algorithmic improvements and improvements to the surface language