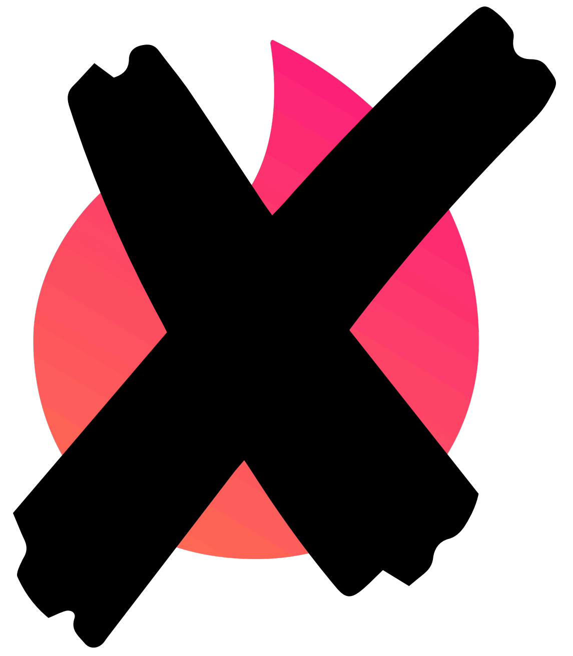
Work In Progress!

Linguistic Tools for Managing Grammatical Domains

Anders Miltner
Devon Loehr
Arnold Mong
Kathleen Fisher
David Walker

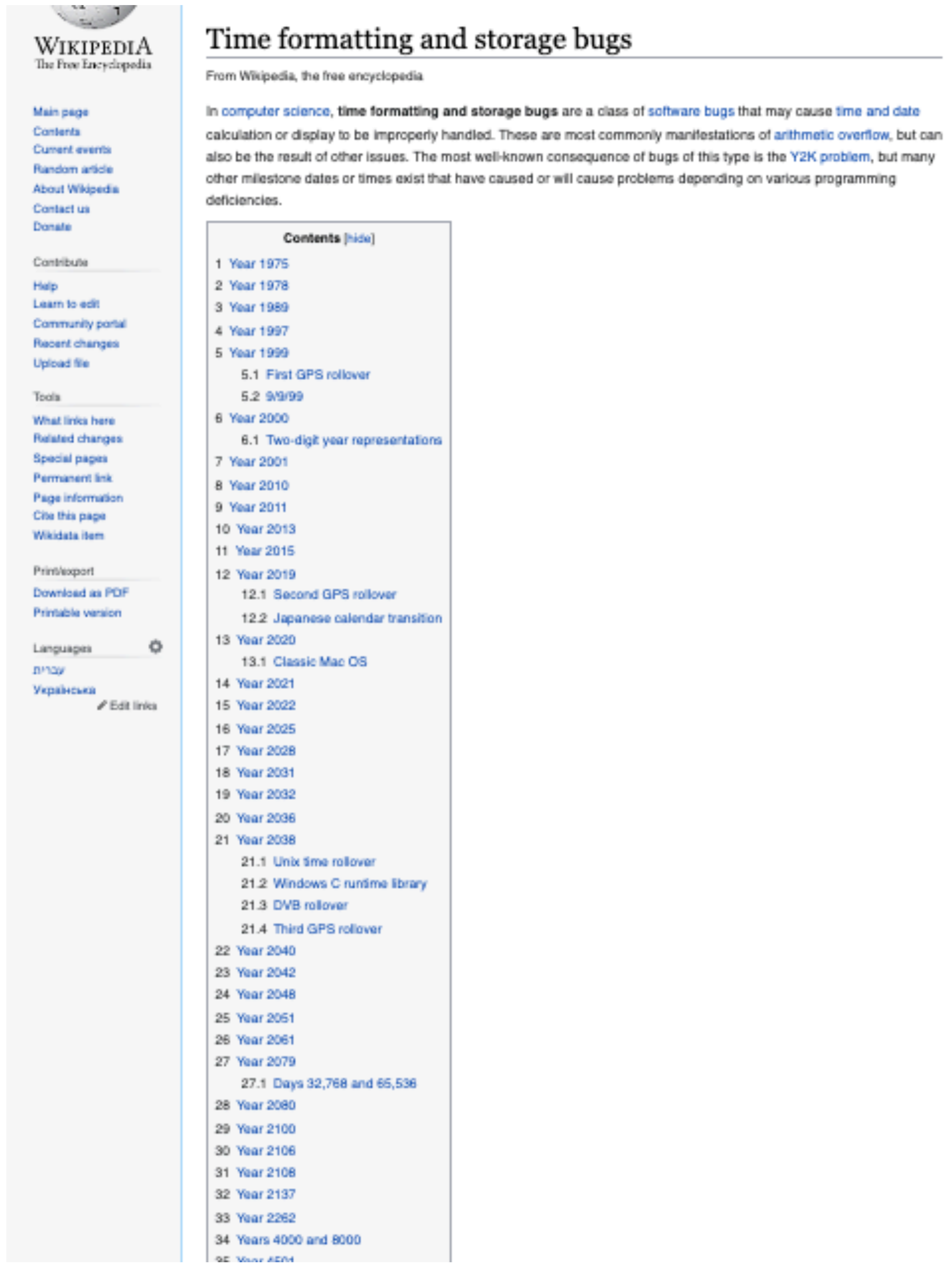
Let’s Talk About Dates



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

Bad Dates

https://en.wikipedia.org/wiki/Time_formatting_and_storage_bugs



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
Edit links

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.

Contents [hide]

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

Grammar Induction

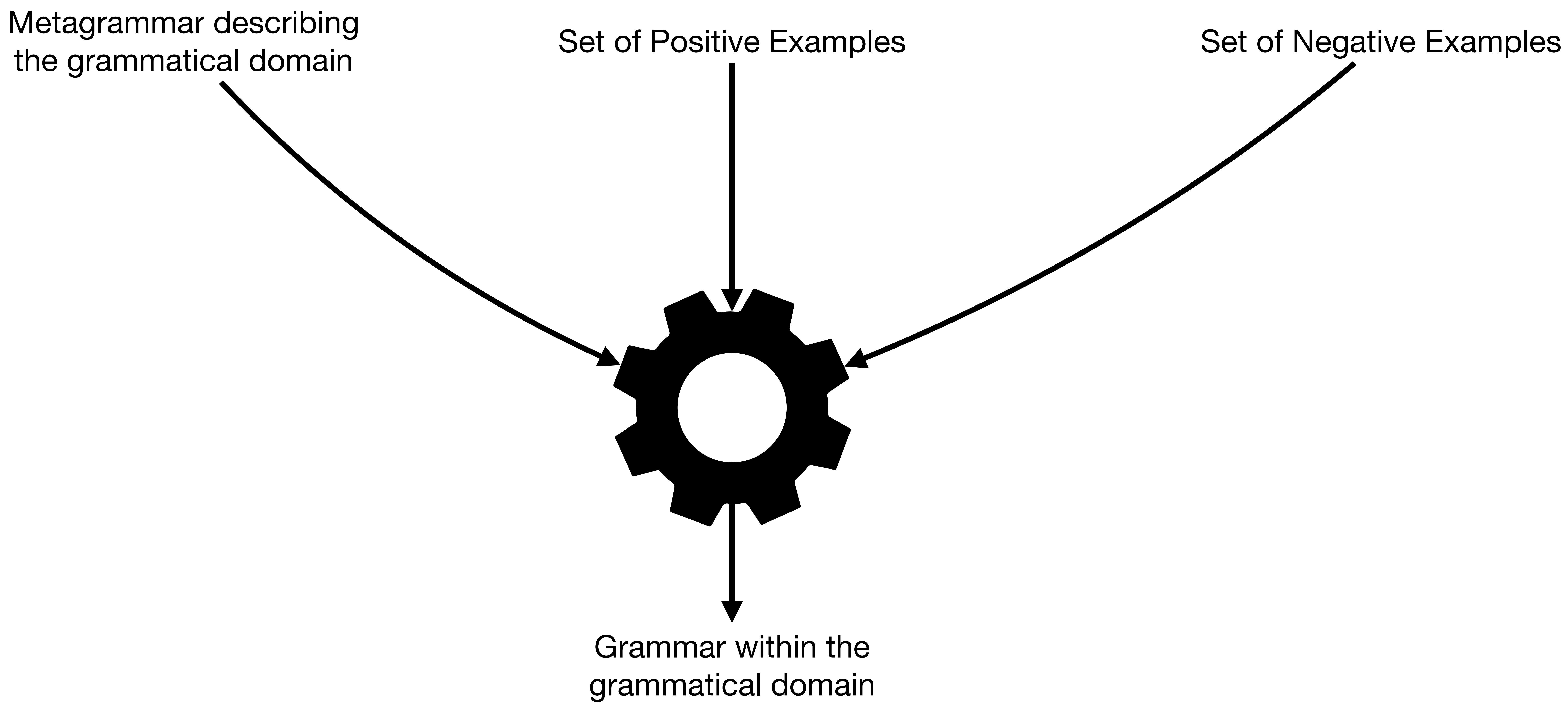
Learn a Grammar from some form of specification

- 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

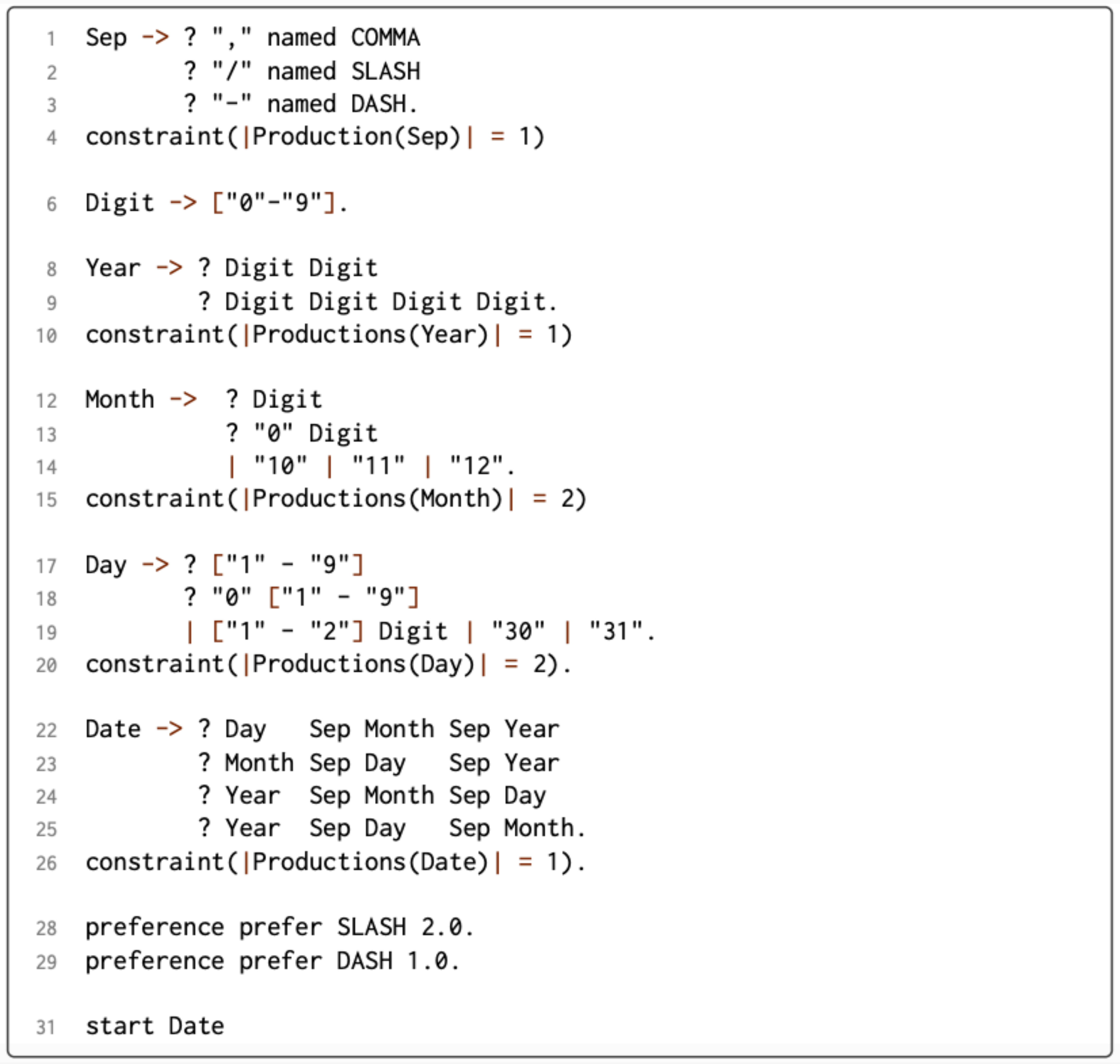
Benefits

Few Examples Required

Explicitly Encoded Biases

Generated Grammar Guarantees

Metagrammar by Example



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

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

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

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

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

Date -> ? Day   Sep Month Sep Year
        ? Month Sep Day   Sep Year
        ? Year  Sep Month Sep Day
        ? Year  Sep Day   Sep Month.
constraint(|Productions(Date)| = 1).

preference prefer SLASH 2.0.
preference prefer DASH 1.0.

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