Prototyping Controlled Mathematical Languages in Jupyter Notebooks

Jan Frederik Schaefer Kai Amann Michael Kohlhase

FAU Erlangen-Nürnberg

ICMS 2020
remotely from Erlangen, Germany

Introduction

- Using math software requires learning input language
- Wouldn't it be nice to just use English? *really hard!*

→ **Controlled mathematical languages** *= CNL for maths*

  - Are formal languages for mathematics
  - Have fixed semantics
  - Imitate natural language

"`forall x \ int(x) => even(x)`"

↓

$\forall x.\mathsf{int}(x) \Rightarrow \mathsf{even}(X)$

Introduction

- Using math software requires learning input language
- Wouldn't it be nice to just use English? *really hard!*

→ **Controlled mathematical languages** *= CNL for maths*

  - Are formal languages for mathematics
  - Have fixed semantics
  - Imitate natural language

"*Every integer is even.*"

↓

$$\forall x.\mathsf{int}(x) \Rightarrow \mathsf{even}(X)$$

Introduction

- Using math software requires learning input language
- Wouldn't it be nice to just use English? *really hard!*

→ **Controlled mathematical languages** *= CNL for maths*

  - Are formal languages for mathematics
  - Have fixed semantics
  - Imitate natural language

- Designing such languages is hard!
- Tool for prototyping: **GLF** *Grammatical Logical Framework*
- Contribution: Jupyter interface

"Every integer is even."

$$\downarrow$$

$$\forall x.\mathsf{int}(x) \Rightarrow \mathsf{even}(X)$$

Introduction

- Using math software requires learning input language
- Wouldn't it be nice to just use English? *really hard!*

→ **Controlled mathematical languages** *= CNL for maths*

  - Are formal languages for mathematics
  - Have fixed semantics
  - Imitate natural language

- Designing such languages is hard!
- Tool for prototyping: **GLF** *Grammatical Logical Framework*
- Contribution: Jupyter interface

"*What is the cardinality of the alternating group on 5 symbols?*"

↓

compute(cardinality(alternating_group(int_term(5))))

Introduction

- Using math software requires learning input language
- Wouldn't it be nice to just use English? *really hard!*

→ **Controlled mathematical languages** *= CNL for maths*

  - Are formal languages for mathematics
  - Have fixed semantics
  - Imitate natural language

- Designing such languages is hard!
- Tool for prototyping: **GLF** *Grammatical Logical Framework*
- Contribution: Jupyter interface

"*What is the cardinality of the alternating group on 5 symbols?*"

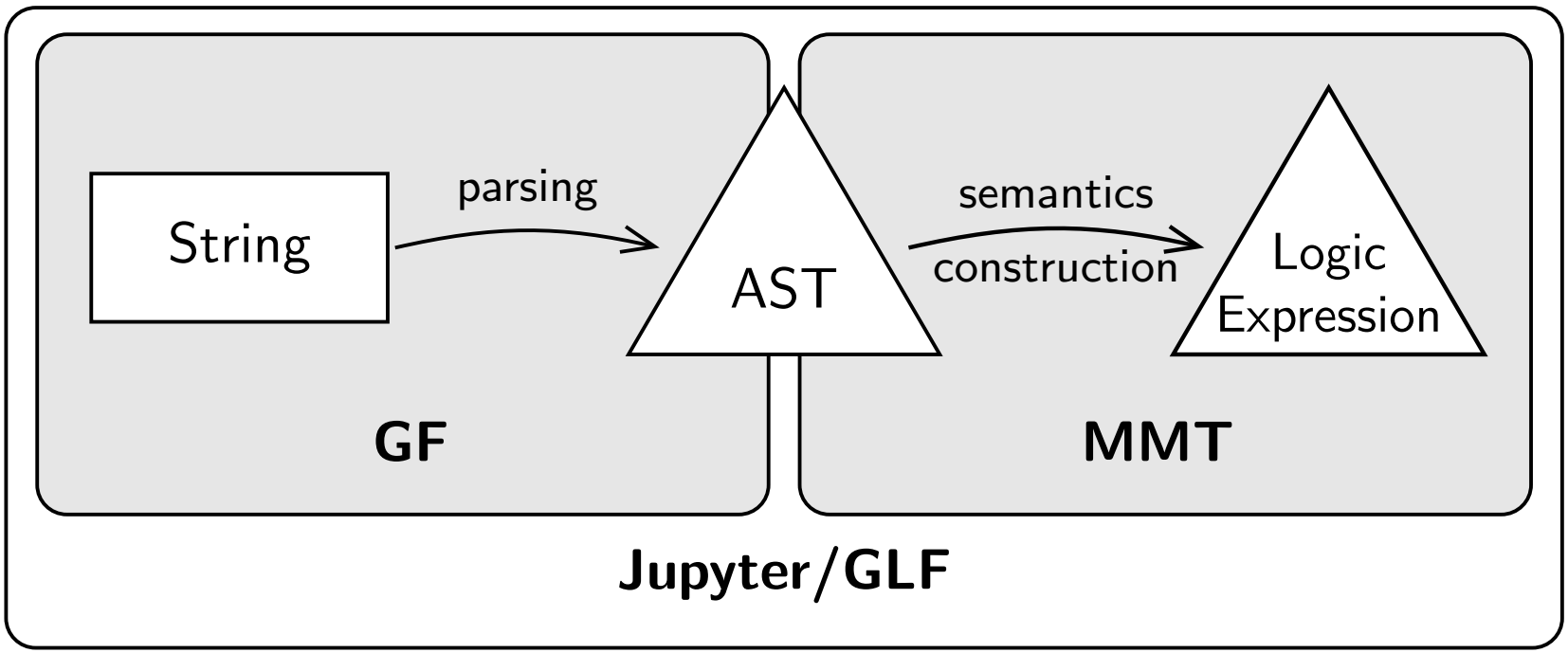
↓

```
print(AlternatingGroup(5).cardinality())
```

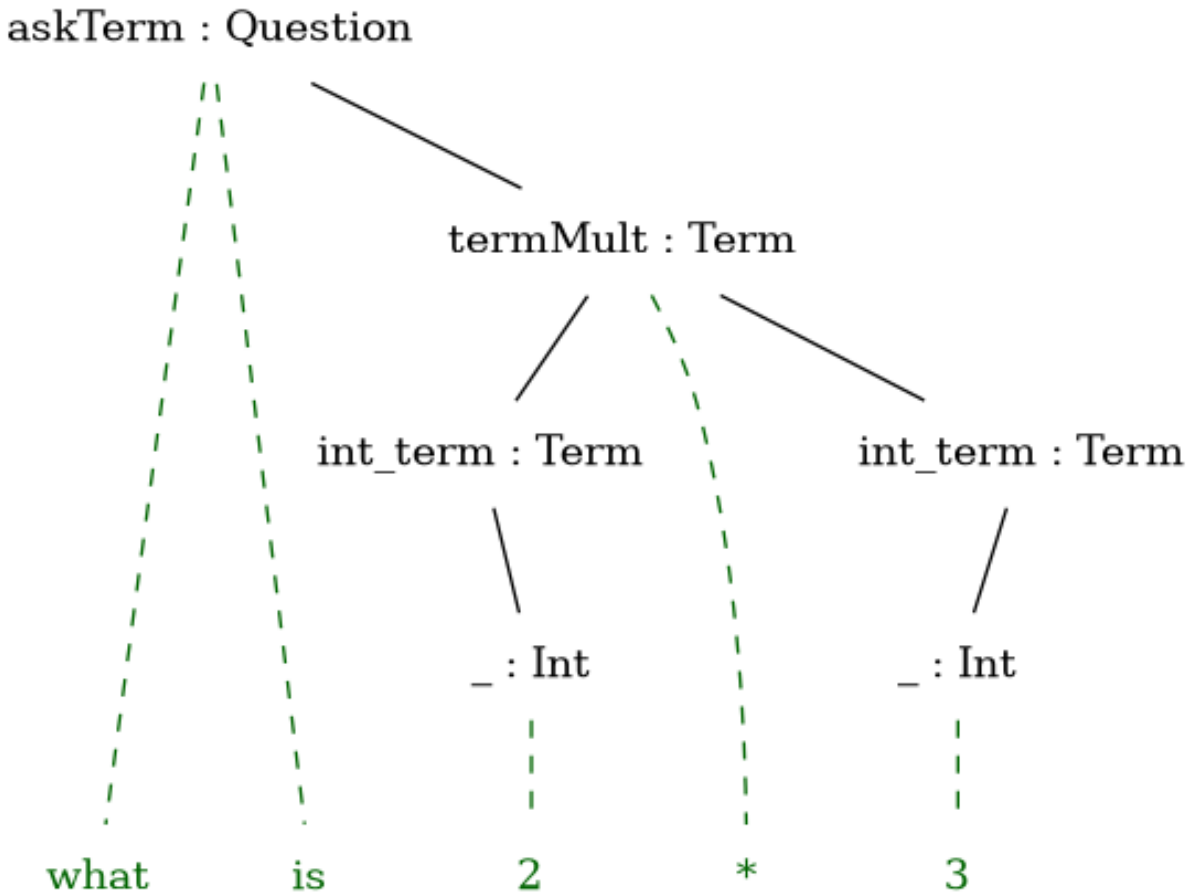
Grammatical Logical Framework (GLF)

Combine two existing frameworks:

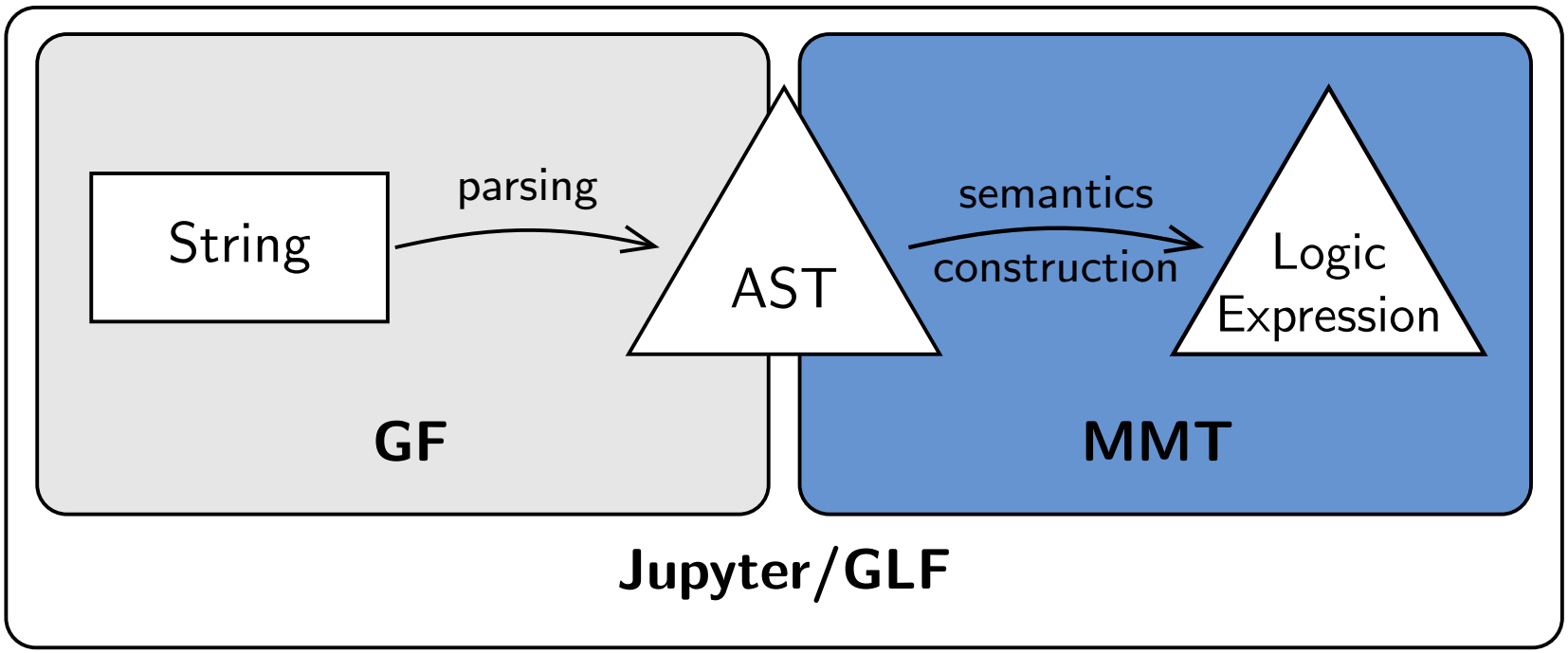
- **GF** (*Grammatical Framework*) for grammar development
- **MMT** for logic development/semantics construction



GF in Jupyter: Grammar Development

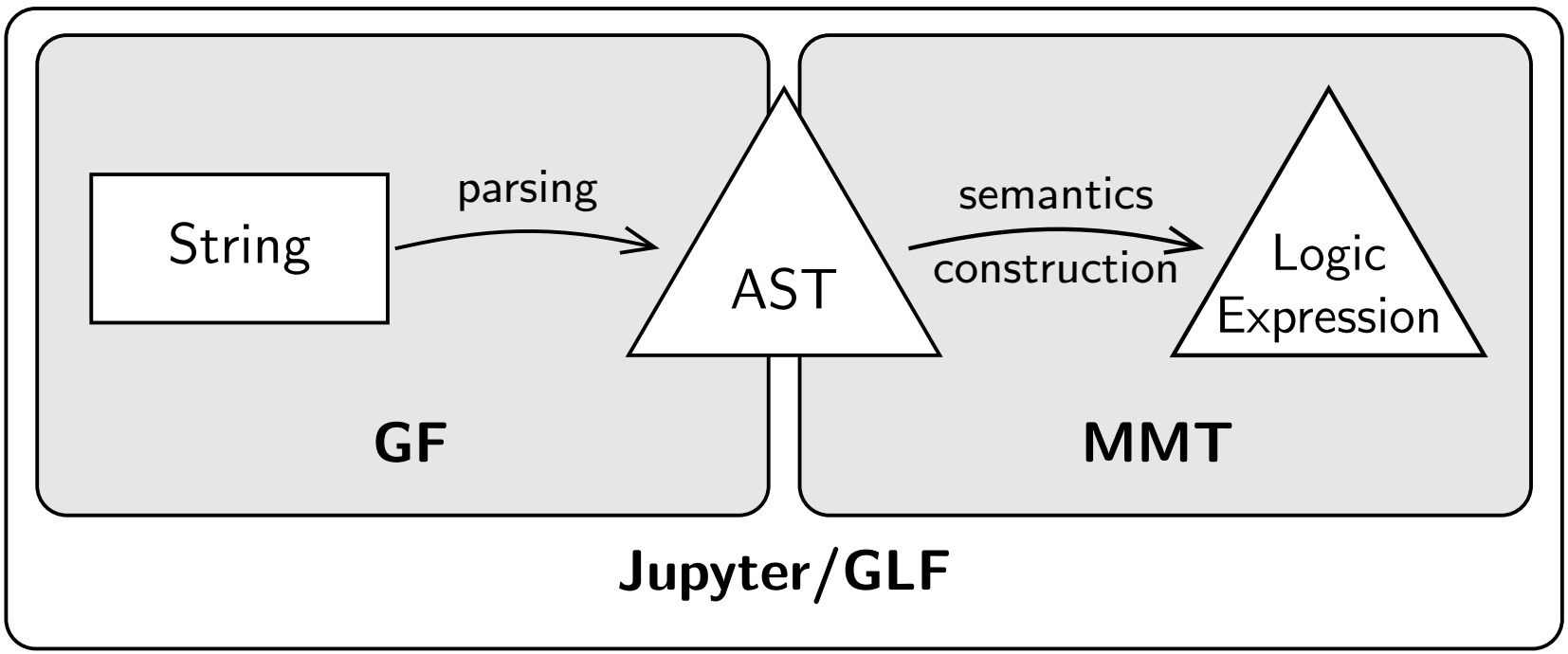


Grammatical Logical Framework (GLF)



Jupyter/GLF

- Run GF and MMT in background
- Identify content using pattern matching
- Tab completion for stub generation



Jupyter/GLF for Larger Projects

- Implement grammar/logic/semantics construction externally
- Use Jupyter notebooks for
  - Experimenting with specific challenges
  - Testing
  - Demos
- Case study: GLForTheL *re-implement ForTheL in GLF*

"a subset of S is a set T such that every element of T belongs to S"

$$\downarrow$$

$$\forall T.T \subseteq S \quad \Leftrightarrow \quad set(T) \wedge \forall x.x \in T \Rightarrow belongto(x, S)$$

Jupyter/GLF For Teaching

- Used in 1-semester course on logic-based language processing
- Homework assignments:
  - Provide partial implementations + explanations
  - Easier to set up
  - Was preferred by most students
- Presentation in Classroom
  - Interactive development with students
  - Easy to share after lecture

```
concrete AnimalGrammarGer of AnimalGrammar = {
    param
        -- TODO: you will need some parameter types
    lincat
        S = Str;
        NP = Str;
        VP = Str;
        -- TODO: Add `Adj` and `N` (remember record types and table types)
    lin
        -- TODO
}
```

Recent Development: GLIF

- We need inference *e.g. for ambiguity resolution*
- We added ELPI (an extension of λProlog) to the pipeline
- Signatures can be generated from MMT
- First experiments with prover generation

```
parse "the ball has a kinetic energy of 12 m N" | construct -e
```

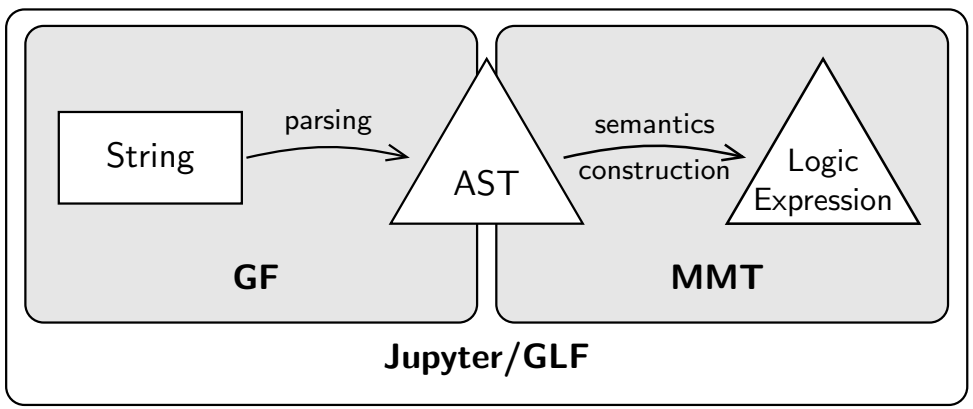
```
(ekin ball (quant 12 (milli newton)))
(ekin ball (quant 12 (mult meter newton)))
```

```
parse "the ball has a kinetic energy of 12 m N" | construct -e | elpi filter dimChec
```

```
ekin ball (quant 12 (mult meter newton))
```

Summary

- We presented a Jupyter kernel for GLF *now GLIF*
- Kernel distinguishes content types with pattern matching:
  - GF grammar modules
  - MMT content
  - Commands *handled by kernel/passed to GF*
- Used for: teaching, prototyping, sharing results/demos, . . .



`https://github.com/KWARC/GLIF`