Language Research in the KWARC group

Jan Frederik Schaefer

FAU Erlangen-Nürnberg/KWARC

Workshop: Approaches to the Logic and Syntax of Mathematical Texts
Erlangen
Dec. 6, 2022

Corpus work with arxiv at KWARC

arxiv.org:

- Open-access pre-print server
- $> 2,000,000$ scientific articles
- Fields: physics, mathematics, computer science, ...
- LaTeX sources
- $\rightsquigarrow$ a great corpus

Corpus work with arxiv at KWARC

Problem:

This: *“The average is $\frac{A+B}{2}$.”*

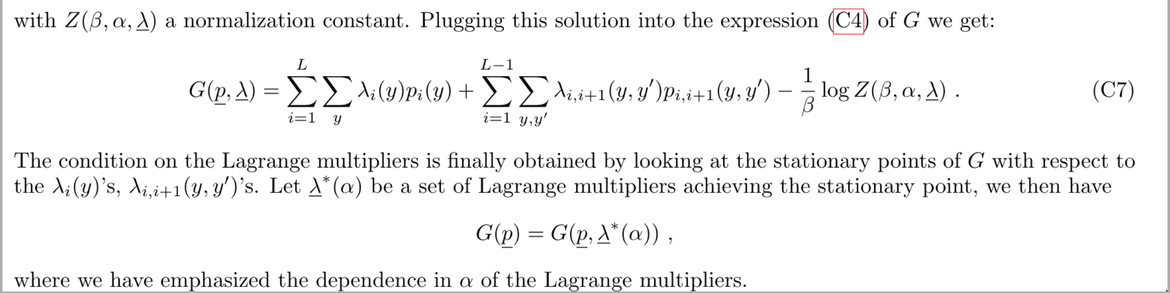
Could be written like this:

```
The average is $\frac{A+B}{2}$.
```

Or like this:

```
\def\avg#1#2{\ensuremath{\frac{#1+#2}2}}
% ...
The average is \avg AB.
```

Corpus work with arxiv at KWARC



with $Z(\beta, \alpha, \underline{\lambda})$ a normalization constant. Plugging this solution into the expression (C4) of G we get:

$$G(\underline{p}, \underline{\lambda}) = \sum_{i=1}^{L} \sum_{y} \lambda_i(y) p_i(y) + \sum_{i=1}^{L-1} \sum_{y,y'} \lambda_{i,i+1}(y, y') p_{i,i+1}(y, y') - \frac{1}{\beta} \log Z(\beta, \alpha, \underline{\lambda}) \ . \tag{C7}$$

The condition on the Lagrange multipliers is finally obtained by looking at the stationary points of G with respect to the $\lambda_i(y)$'s, $\lambda_{i,i+1}(y, y')$'s. Let $\underline{\lambda}^*(\alpha)$ be a set of Lagrange multipliers achieving the stationary point, we then have

$$G(\underline{p}) = G(\underline{p}, \underline{\lambda}^*(\alpha)) \ ,$$

where we have emphasized the dependence in α of the Lagrange multipliers.

Example from [BP22]

Corpus work with arxiv at KWARC

Problem:

This: *“The average is $\frac{A+B}{2}$.”*

Could be written like this:

```
The average is $\frac{A+B}{2}$.
```

Or like this:

```
\def\avg#1#2{\ensuremath{\frac{#1+#2}2}}
% ...
The average is \avg AB.
```

Solution: Convert to more managable format: HTML with MathML.

HTML, MathML

“The average is $\frac{A+B}{2}$.”

```
<p>The average is <math>...</math>.</p>
```

Presentation MathML

```
<mfrac>
    <mrow>
        <mi>A</mi>
        <mo>+</mo>
        <mi>B</mi>
    </mrow>
    <mn>2</mn>
</mfrac>
```

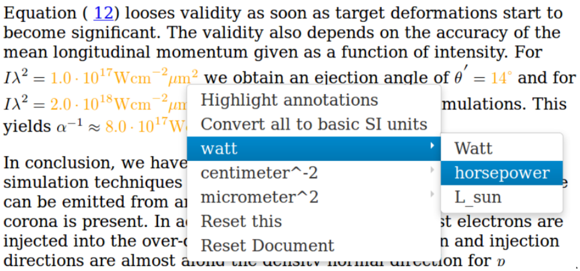
Content MathML

```
<apply>
    <divide/>
    <apply>
        <plus/>
        <ci>A</ci>
        <ci>B</ci>
    </apply>
    <cn type="integer">2</cn>
</apply>
```

ar5iv corpus

- Use LaTeXML to convert arxiv to HTML+MathML *Done by Deyan Ginev*
- $\rightsquigarrow$ ar5iv corpus $\approx 2 \cdot 10^6$ *documents*
- Goal: Extract semantic information and provide services

search, interactive documents, . . .



Equation (12) looses validity as soon as target deformations start to become significant. The validity also depends on the accuracy of the mean longitudinal momentum given as a function of intensity. For $I\lambda^2 = 1.0 \cdot 10^{17}\mathrm{Wcm}^{-2}\mu\mathrm{m}^2$ we obtain an ejection angle of $\theta^{'} = 14^{\circ}$ and for
$I\lambda^2 = 2.0 \cdot 10^{18}\mathrm{Wcm}^{-2}\mu\mathrm{m}$ mulations. This
yields $\alpha^{-1} \approx 8.0 \cdot 10^{17}\mathrm{W}$

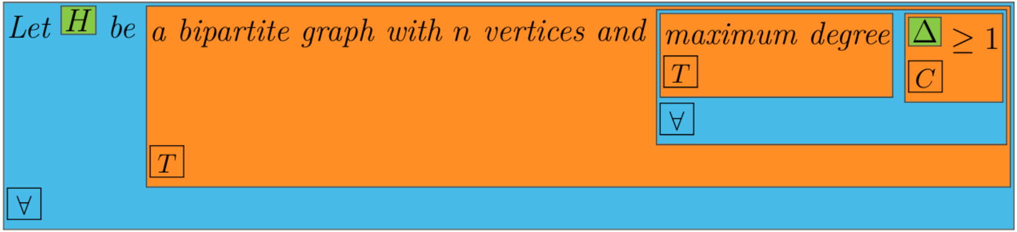
In conclusion, we have
simulation techniques e
can be emitted from a
corona is present. In a st electrons are
injected into the over- n and injection
directions are almost on for p

Screenshot from [MK]

ar5iv corpus

- Use LaTeXML to convert arxiv to HTML+MathML *Done by Deyan Ginev*
- $\rightsquigarrow$ ar5iv corpus $\approx 2 \cdot 10^6$ *documents*
- Goal: Extract semantic information and provide services

  search, interactive documents, . . .



Let H *be a bipartite graph with* n *vertices and maximum degree* $\Delta \geq 1$

Screenshot from [Sch16]

—And Now for Something Completely Different—

GLIF: A tool for prototyping natural language semantics

Natural Language Semantics (Symbolic)

For me:

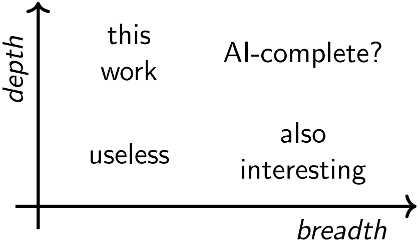
Translating natural language into a formal semantic representation (logic).

Example:

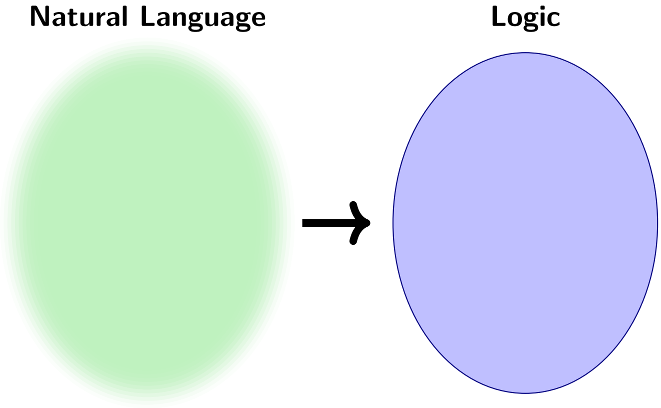
"Every student paints and is quiet." $\rightsquigarrow$ $\forall x.s(x) \Rightarrow (p(x) \wedge q(x))$

Rule-based (no ML):

Parsing $\rightsquigarrow$ semantics construction $\rightsquigarrow$ inference.



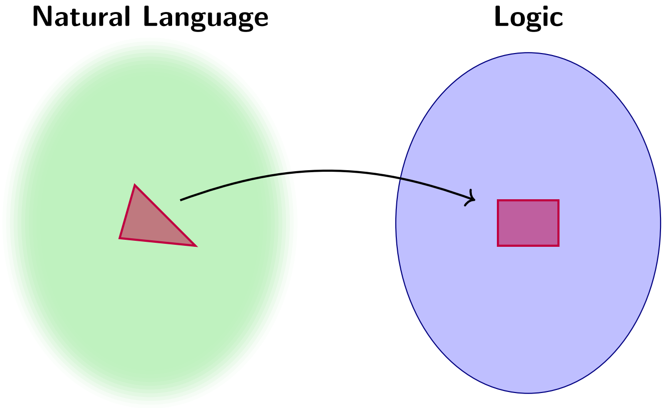
Method of Fragments



How do we get from messy language to formal logic?

Montague [Mon70]: Look at a “nice” subset and map into logic.

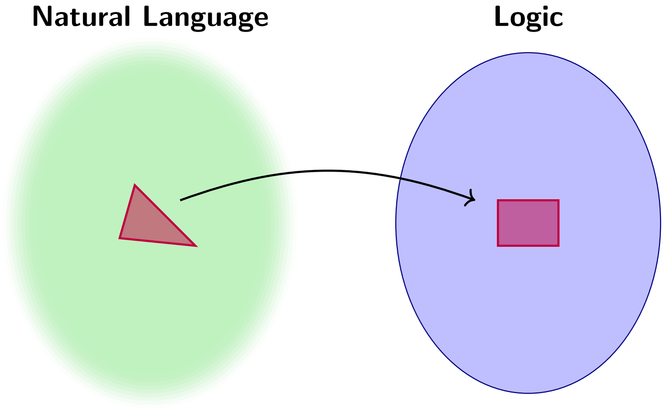
Method of Fragments



How do we get from messy language to formal logic?

Montague [Mon70]: Look at a “nice” subset and map into logic.

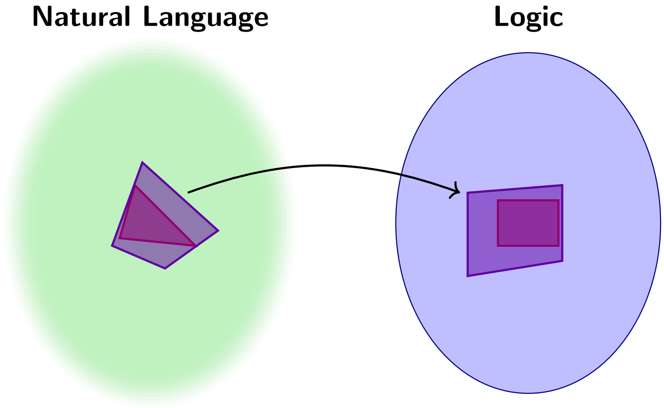
Method of Fragments



“Ahmed paints and Berta is quiet.” $p(a) \wedge q(b)$

“Ahmed doesn’t paint.” $\neg p(a)$

Method of Fragments



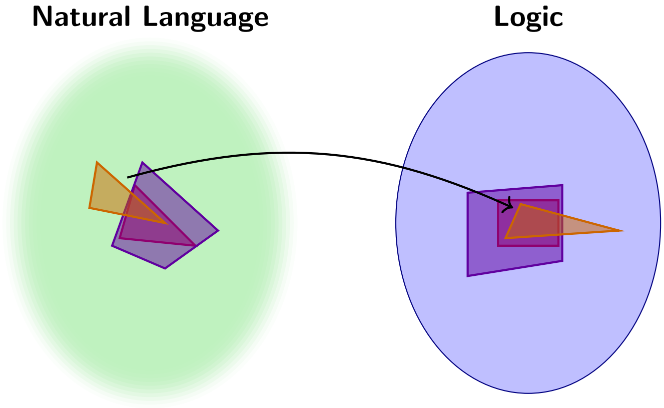
"Every student paints and is quiet."

$\forall x.s(x) \Rightarrow (p(x) \wedge q(x))$

"Nobody paints."

$\neg\exists x.p(x)$

Method of Fragments



"Ahmed isn't allowed to paint." $\neg\Diamond p(a)$

"Ahmed and Berta must paint." $(\Box p(a)) \wedge \Box p(b)$

Method of Fragments

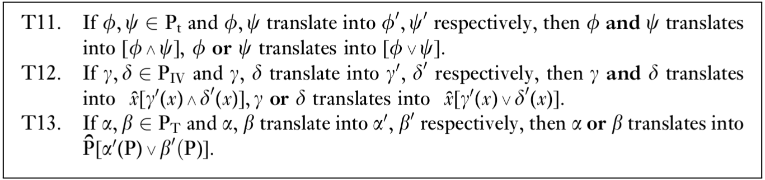
Hand-waving is problematic:

"Ahmed paints. He is quiet." $\overset{?}{\rightsquigarrow}$ $p(a) \wedge q(a)$

Montague: Specify

- grammar, *fixes NL subset*
- target logic,
- semantics construction. *maps parse trees to logic*

Example from [Mon74]



T11. If $\phi, \psi \in \mathrm{P_t}$ and ϕ, ψ translate into ϕ', ψ' respectively, then ϕ **and** ψ translates into $[\phi \wedge \psi]$, ϕ **or** ψ translates into $[\phi \vee \psi]$.

T12. If $\gamma, \delta \in \mathrm{P_{IV}}$ and γ, δ translate into γ', δ' respectively, then γ **and** δ translates into $\hat{x}[\gamma'(x) \wedge \delta'(x)], \gamma$ **or** δ translates into $\hat{x}[\gamma'(x) \vee \delta'(x)]$.

T13. If $\alpha, \beta \in \mathrm{P_T}$ and α, β translate into α', β' respectively, then α **or** β translates into $\hat{\mathrm{P}}[\alpha'(\mathrm{P}) \vee \beta'(\mathrm{P})]$.

Claim: That doesn't scale well ⇝ **We need prototyping!**

NLU Prototyping

```
> translate "Every student paints and is quiet."
```

$$\forall x.s(x) \Rightarrow (p(x) \wedge q(x))$$

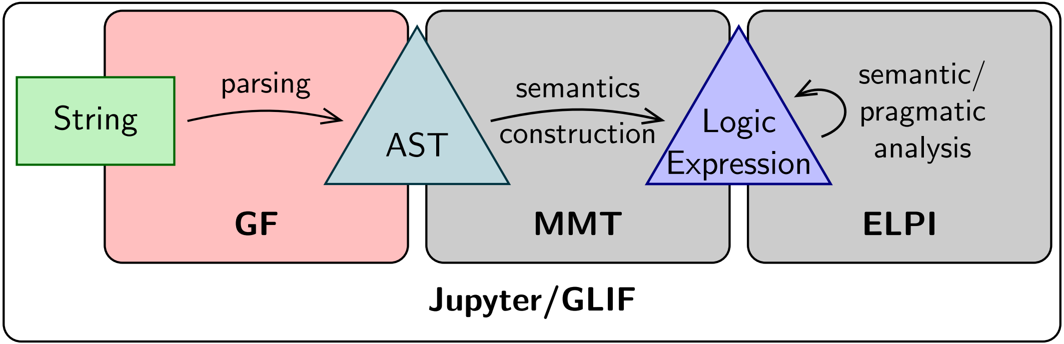
```
> answer "Every student is quiet.  John is a student.  Is John quiet?"
```

$$\forall x.s(x) \Rightarrow q(x), s(j) \vdash^{?} q(j)$$

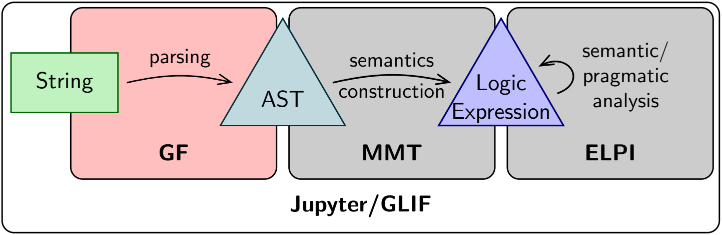
```
yes
```

- Traditionally done in Prolog/Haskell
  - $\longrightarrow$ requires a lot of work
- A dedicated framework might be better
  - $\longrightarrow$ only partial solutions exist
- Can we combine existing partial solutions?
  - $\rightsquigarrow$ GLIF

Components of GLIF: GF



Components of GLIF: GF

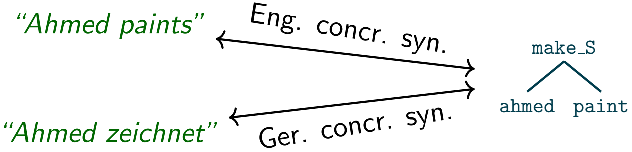


Components of GLIF: Grammatical Framework [GF]

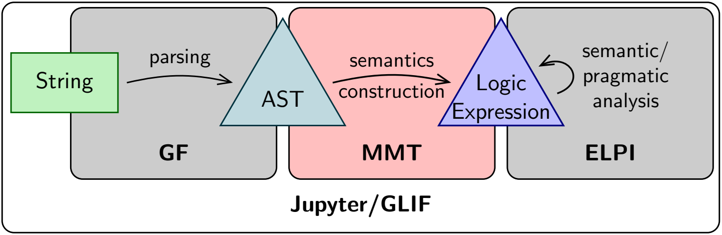
- Specialized for developing natural language grammars
- Separates abstract and concrete syntax

```
make_S : NP -> VP -> S;                  abstract
make_S np vp = np.s ++ vp.s!np.n;        concrete
```

- Abstract syntax based on LF
- Comes with large library $\geq$ 36 *languages*



Components of GLIF: MMT



Components of GLIF: MMT

- Modular logic development and knowledge repr.
- Not specialized in one logical framework *we use LF*
- We will use MMT to:
  1. represent abstract syntax
  2. specify target logic and discourse domain theory
  3. specify semantics construction

Components of GLIF: MMT

- Modular logic development and knowledge repr.
- Not specialized in one logical framework *we use LF*
- We will use MMT to:
  1. **represent abstract syntax**
  2. specify target logic and discourse domain theory
  3. specify semantics construction

GF

```
cat
  NP; VP; S;
fun
  make_S :
    NP -> VP -> S;
```

$\longmapsto$

MMT

```
NP : type
VP : type
S  : type
make_S :
  NP → VP → S
```

Components of GLIF: MMT

- Modular logic development and knowledge repr.
- Not specialized in one logical framework *we use LF*
- We will use MMT to:
  1. represent abstract syntax
  2. **specify target logic and discourse domain theory**
  3. specify semantics construction

Logic Syntax

```
o  :  type  //propositions
¬  :  o  →  o
∧  :  o  →  o  →  o
∨  :  o  →  o  →  o

ι  :  type  //individuals
∀  :  (ι  →  o)  →  o
∃  :  (ι  →  o)  →  o
```

Discourse Domain

```
paint  :  ι  →  o
quiet  :  ι  →  o
ahmed  :  ι
berta  :  ι
```

idea: $\forall f$ or $\forall \lambda x.f(x)$
instead of $\forall x.f(x)$

Components of GLIF: MMT

- Modular logic development and knowledge repr.
- Not specialized in one logical framework *we use LF*
- We will use MMT to:
  1. represent abstract syntax
  2. specify target logic and discourse domain theory
  3. **specify semantics construction**

Semantics Construction

map symbols in abstract syntax to terms in logic/domain theory

Simple setting

```
S        ↦ o
NP       ↦ ι
VP       ↦ ι → o
make_S   ↦ λn.λv.v n
ahmed    ↦ ahmed
```

More advanced

```
NP        ↦ (ι→o)→o
sentence  ↦ λn.λv.n v
everyone  ↦ λp.∀λx.p x
berta     ↦ λp.p berta
```

Example: Parsing + Semantics Construction

"Ahmed and Berta paint"

↓parsing

make_S (andNP ahmed berta) paint

↓semantics construction

`(λn.λv.n v) ((λa.λb.λp.a p ∧ b p) (λp.p ahmed) (λp.p berta)) pain`

↓β-reduction

`paint ahmed ∧ paint berta`

Example: Student Project [Int]

```
parse "John has not always run" | construct
```

```
¬H (run john)
```

```
parse "John has to have been allowed to always run" | construct
```

```
□P ◇(H (run john)∧G (run john))
```

```
parse "John probably will never run" | construct
```

```
Prob G ¬(run john)
```

```
parse "it has to be possible that John runs" | construct
```

```
□◇(run john)
```

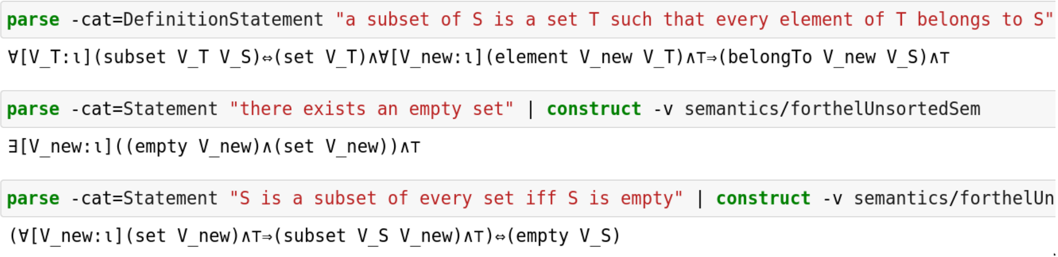
```
parse "Mary saw that John would kill the dog" | construct
```

```
P 👁mary👁F (kill john dog)
```

```
parse "Mary runs and John sees it" | construct
```

```
(run mary)∧👁john👁(run mary)
```

Example: ForTheL

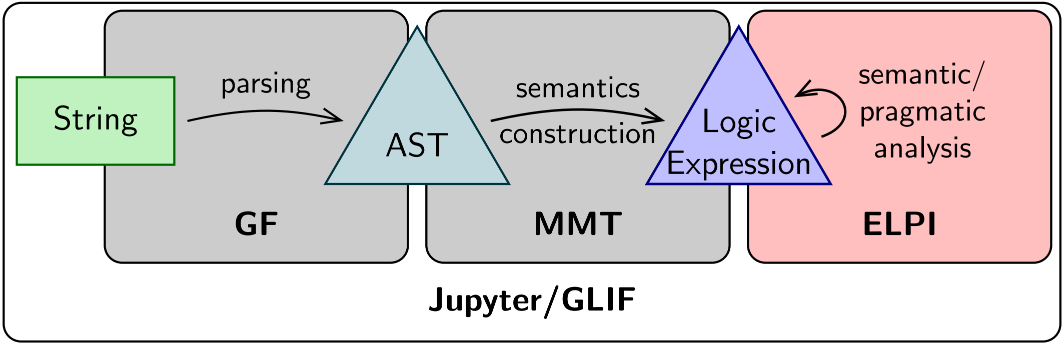


```
parse -cat=DefinitionStatement "a subset of S is a set T such that every element of T belongs to S"
∀[V_T:ι](subset V_T V_S)⇔(set V_T)∧∀[V_new:ι](element V_new V_T)∧⊤⇒(belongTo V_new V_S)∧⊤

parse -cat=Statement "there exists an empty set" | construct -v semantics/forthelUnsortedSem
∃[V_new:ι]((empty V_new)∧(set V_new))∧⊤

parse -cat=Statement "S is a subset of every set iff S is empty" | construct -v semantics/forthelUn
(∀[V_new:ι](set V_new)∧⊤⇒(subset V_S V_new)∧⊤)⇔(empty V_S)
```

Components of GLIF: ELPI



Components of GLIF: ELPI

- Implementation and extension of λProlog ≈ *Prolog + HOAS*
- MMT can generate logic signatures
- First experiments with prover generation
- Generic inference/reasoning step after semantics construction

MMT

```
o : type //propositions
¬ : o → o
∧ : o → o → o
∨ : o → o → o

ι : type //individuals
∀ : (ι → o) → o
∃ : (ι → o) → o
```

ELPI

```
kind o type.
not : o -> o.
and : o -> o -> o.
or  : o -> o -> o.

kind i type.
type forall (i -> o) -> o.
type exists (i -> o) -> o.
```

Example: Discard wrong readings in controlled natural language

“the ball has a mass of 5kg” $\longrightarrow$ AST $\xrightarrow{\qquad}$ $\mathrm{mass}(\mathrm{theball}, \mathrm{quant}(5, \mathrm{kilo\ gram}))$

Example: Discard wrong readings in controlled natural language

$$\begin{tikzcd}
\textit{``the ball has a mass of 5kg''} \arrow[r] & \mathrm{AST} \arrow[r] & \mathrm{mass}(\mathrm{theball}, \mathrm{quant}(5, \mathrm{kilo\ gram}))
\end{tikzcd}$$

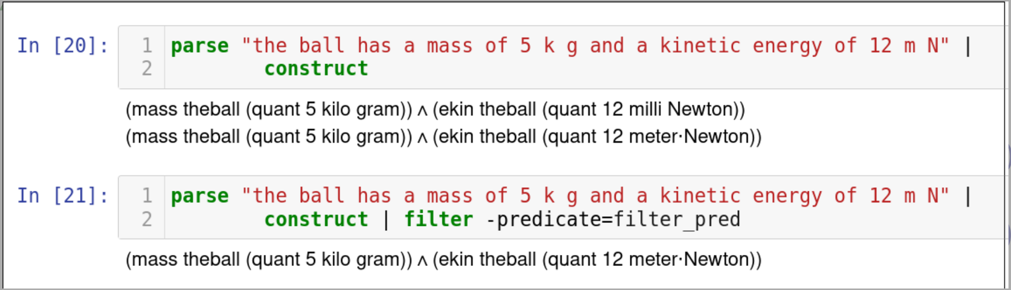
$$\begin{tikzcd}
 & \mathrm{AST}_1 \arrow[r] & \lambda x.E_{\mathrm{kin}}(x, \mathrm{quant}(2, \mathbf{milli\ Newton})) \\
\textit{``a kinetic energy of 12mN''} \arrow[ur] \arrow[dr] & & \\
 & \mathrm{AST}_2 \arrow[r] & \lambda x.E_{\mathrm{kin}}(x, \mathrm{quant}(2, \mathbf{meter}\cdot\mathbf{Newton}))
\end{tikzcd}$$

Example: Discard wrong readings in controlled natural language

$$\begin{tikzcd}
\textit{``the ball has a mass of 5kg''} \arrow[r] & \text{AST} \arrow[r] & \text{mass}(\text{theball}, \text{quant}(5, \text{kilo gram}))
\end{tikzcd}$$

$$\begin{tikzcd}
 & \text{AST}_1 \arrow[r] & \lambda x.E_{\text{kin}}\xcancel{(x, \text{quant}(2, \textbf{milli Newton}}))) \\
\textit{``a kinetic energy of 12mN''} \arrow[ur] \arrow[dr] & & \\
 & \text{AST}_2 \arrow[r] & \lambda x.E_{\text{kin}}(x, \text{quant}(2, \textbf{meter}\cdot\textbf{Newton}))
\end{tikzcd}$$

Example: Discard wrong readings in controlled natural language



In [20]:

```
parse "the ball has a mass of 5 k g and a kinetic energy of 12 m N" |
        construct
```

(mass theball (quant 5 kilo gram)) ∧ (ekin theball (quant 12 milli Newton))

(mass theball (quant 5 kilo gram)) ∧ (ekin theball (quant 12 meter·Newton))

In [21]:

```
parse "the ball has a mass of 5 k g and a kinetic energy of 12 m N" |
        construct | filter -predicate=filter_pred
```

(mass theball (quant 5 kilo gram)) ∧ (ekin theball (quant 12 meter·Newton))

Example: Epistemic Q&A

John knows that Mary or Eve knows that Ping has a dog. (S_1)
Mary doesn't know if Ping has a dog. (S_2)
Does Eve know if Ping has a dog? (Q)

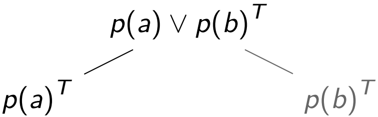
$$\begin{aligned} S_1 &= \Box_{john}(\Box_{mary} hd(ping) \vee \Box_{eve} hd(ping)) \\ S_2 &= \neg(\Box_{mary} hd(ping) \vee \Box_{mary} \neg hd(ping)) \\ Q &= \Box_{eve} hd(ping) \vee \Box_{eve} \neg hd(ping) \end{aligned}$$

$$\begin{array}{lcl} S_1, S_2 \vdash_{S5_n} Q & \leadsto & \text{yes} \\ S_1, S_2 \vdash_{S5_n} \neg Q & \leadsto & \text{no} \\ \text{else} & \leadsto & \text{maybe} \end{array}$$

Example: Tableaux Machine [KK03]

- Can use tableaux for model generation
- Tableau machine: pick “best” branch as model and continue there with next sentence *like a human?*

“Ahmed or Berta paints”



$p(a) \vee p(b)^T$

$p(a)^T$

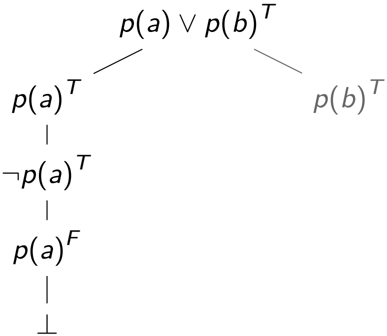
$p(b)^T$

Example: Tableaux Machine [KK03]

- Can use tableaux for model generation
- Tableau machine: pick "best" branch as model and continue there with next sentence *like a human?*

"Ahmed or Berta paints"

"Ahmed doesn't paint"



$p(a) \vee p(b)^T$

$p(a)^T$

$p(b)^T$

$\neg p(a)^T$

$p(a)^F$

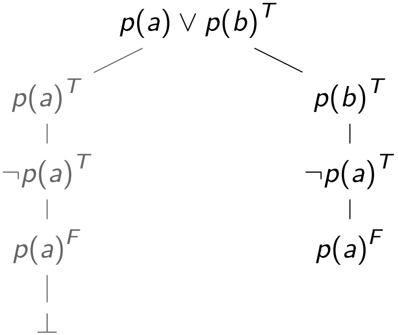
$\bot$

Example: Tableaux Machine [KK03]

- Can use tableaux for model generation
- Tableau machine: pick "best" branch as model and continue there with next sentence *like a human?*

"Ahmed or Berta paints"

"Ahmed doesn't paint"



Example: Tableaux Machine

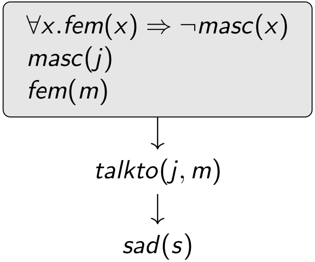
Background Knowledge

"John talks to Mary."

$talkto(j, m)$

"Sasha is sad."

$sad(s)$



$\forall x.fem(x) \Rightarrow \neg masc(x)$
$masc(j)$
$fem(m)$

$talkto(j, m)$

$sad(s)$

Example: Tableaux Machine

Background Knowledge

“John talks to Mary.”

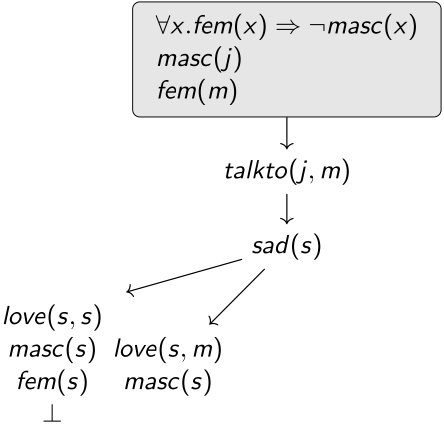
$talkto(j, m)$

“Sasha is sad.”

$sad(s)$

“He loves her.”

$\exists X.masc(X) \wedge$
$\quad \exists Y.fem(Y) \wedge love(X, Y)$



$\forall x.fem(x) \Rightarrow \neg masc(x)$
$masc(j)$
$fem(m)$

$talkto(j, m)$

$sad(s)$

$love(s, s)$
$masc(s)$
$fem(s)$
$\perp$

$love(s, m)$
$masc(s)$

Example: Tableaux Machine

Background Knowledge

"John talks to Mary."

$talkto(j, m)$

"Sasha is sad."

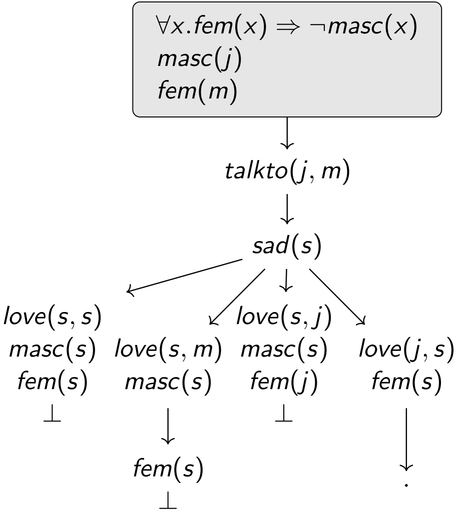
$sad(s)$

"He loves her."

$\exists X.masc(X) \wedge$
$\quad \exists Y.fem(Y) \wedge love(X, Y)$

"Sasha is a woman."

$fem(s)$



$\forall x.fem(x) \Rightarrow \neg masc(x)$
$masc(j)$
$fem(m)$

$talkto(j, m)$

$sad(s)$

$love(s, s)$, $masc(s)$, $fem(s)$, $\perp$

$love(s, m)$, $masc(s)$, $fem(s)$, $\perp$

$love(s, j)$, $masc(s)$, $fem(j)$, $\perp$

$love(j, s)$, $fem(s)$, $.$

Example: Tableaux Machine

Background Knowledge

“John talks to Mary.”

$talkto(j, m)$

“Sasha is sad.”

$sad(s)$

“He loves her.”

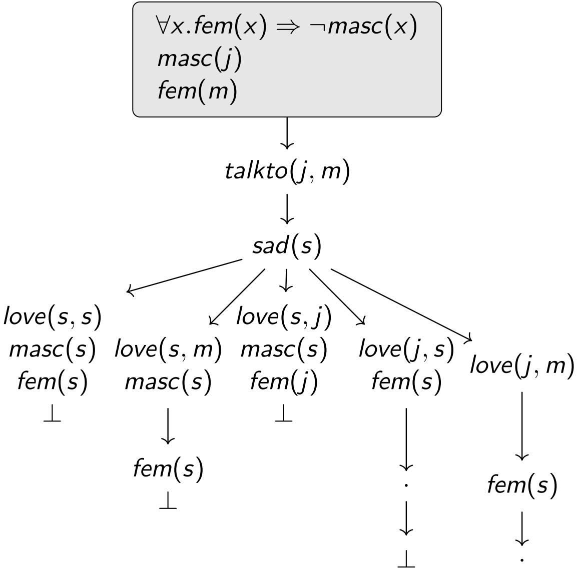
$\exists X.masc(X) \wedge$
$\quad \exists Y.fem(Y) \wedge love(X, Y)$

“Sasha is a woman.”

$fem(s)$

“John doesn’t love Sasha.”

$\neg love(j, s)$



$\forall x.fem(x) \Rightarrow \neg masc(x)$
$masc(j)$
$fem(m)$

$talkto(j, m)$

$sad(s)$

$love(s, s)$
$masc(s)$
$fem(s)$
$\perp$

$love(s, m)$
$masc(s)$
$fem(s)$
$\perp$

$love(s, j)$
$masc(s)$
$fem(j)$
$\perp$

$love(j, s)$
$fem(s)$
$\cdot$
$\perp$

$love(j, m)$
$fem(s)$
$\cdot$

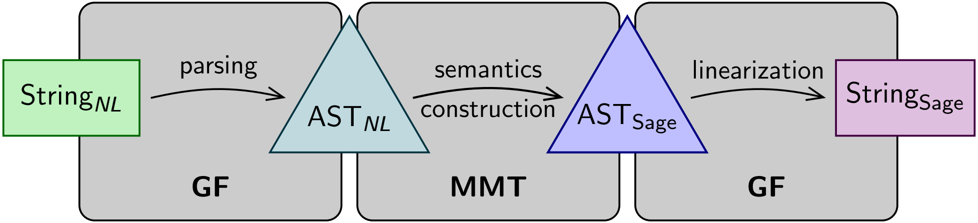
Example: Input Language for SageMath

- Can we make a natural input language for SageMath? *WolframAlpha-like*

```
sage: g = AlternatingGroup(5)
sage: g.cardinality()
60
```

"Let G be the alternating group on 5 symbols. What is the cardinality of G?"

Example: Input Language for SageMath



Example: Input Language for SageMath

```
> Let G be the alternating group on 5 symbols.
# G = AlternatingGroup(5)

> Let |H| be a notation for the cardinality of H.
# def bars(H): return H.cardinality()

> What is |G|?
# print(bars(G))
60

> Let A_N be a notation for the alternating group on N symbols.
# def A(N): return AlternatingGroup(N)

> What are the cardinalities of A_4 and A_5?
# print(A(4).cardinality()); print(A(5).cardinality())
12
60
```

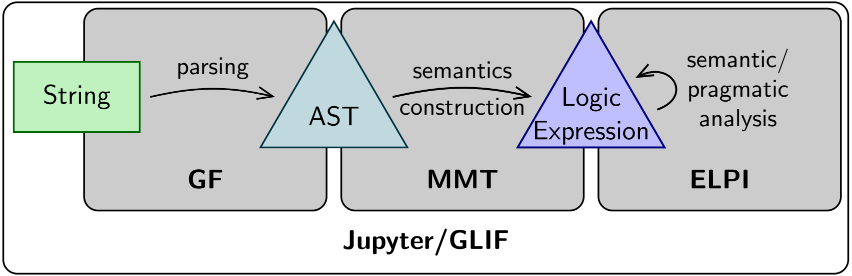
Conclusion

Summary:

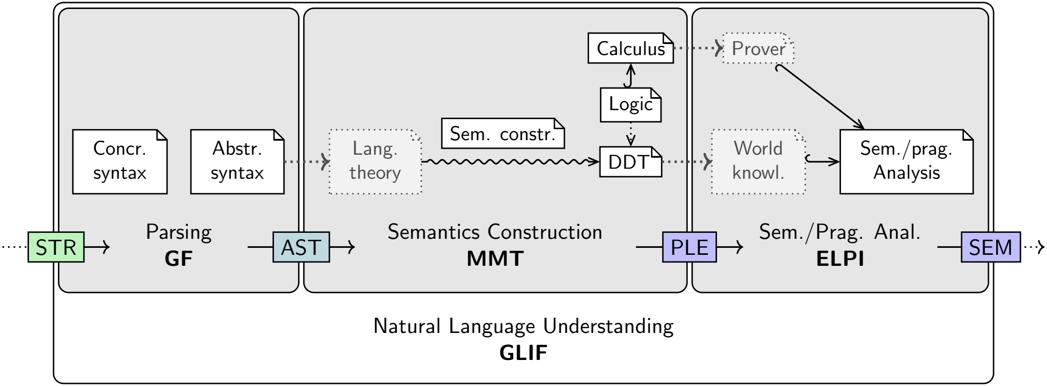
- GLIF = GF + MMT + ELPI
- Prototyping natural language semantics
- We use it for teaching

Examples:

1. *"a kinetic energy of 12mN"*
2. *"He loves her"* (tableaux machine)
3. *"John knows that Eve has a dog"*
4. *"What is the cardinality of G?"*



Pipeline Specification



References I

[BP22] Louise Budzynski and Andrea Pagnani. *Small Coupling Expansion for Multiple Sequence Alignment*. 2022. DOI: 10.48550/ARXIV.2210.03463. URL: https://arxiv.org/abs/2210.03463.

[GF] *GF - Grammatical Framework*. URL: http://www.grammaticalframework.org (visited on 09/27/2017).

[Int] *Case study of implementing intensional logic in GLIF*. URL: https://github.com/us77ipis/glif-intensional-logic (visited on 07/19/2022).

[KK03] Michael Kohlhase and Alexander Koller. "Resource-Adaptive Model Generation as a Performance Model". In: *Logic Journal of the IGPL* 11.4 (2003), pp. 435–456. URL: http://jigpal.oxfordjournals.org/cgi/content/abstract/11/4/435.

References II

[MK] Tom Wiesing Michael Kohlhase. *Report on OpenDreamKit deliverable D4.9, in-place computation in active documents (context/computation)*. URL: https://raw.githubusercontent.com/OpenDreamKit/OpenDreamKit/master/WP4/D4.9/report-final.pdf (visited on 12/02/2022).

[Mon70] R. Montague. "English as a Formal Language". In: Reprinted in [Tho74], 188–221. Edizioni di Communita, Milan, 1970, pp. 189–224.

[Mon74] Richard Montague. "The Proper Treatment of Quantification in Ordinary English". In: *Formal Philosophy. Selected Papers*. Ed. by R. Thomason. New Haven: Yale University Press, 1974.

[Sch16] Jan Frederik Schaefer. "Declaration Spotting in Mathematical Documents". B. Sc. Thesis. Jacobs University Bremen, 2016. URL: https://gl.kwarc.info/supervision/BSc-archive/blob/master/2016/schaefer-frederick.pdf.

References III

[Tho74] R. Thomason, ed. *Formal Philosophy: selected Papers of Richard Montague*. Yale University Press, New Haven, CT, 1974.