High-Precision Semantics Extraction for Mathematics

Jan Frederik Schaefer

GF Summer School 2018
Stellenbosch, South Africa

December 11, 2018

- MSc Student, Computer Science
- FAU (**F**riedrich-**A**lexander-**U**niversität Erlangen-Nürnberg) (in Southern Germany)
- KWARC research group
  - Led by Michael Kohlhase
  - Knowledge representation and reasoning techniques
  - Focus on mathematical content

- A **S***emantic,* **M***ultilingual* **Glo***ssary of* **M***athematics*
- Definitions of mathematical terms
- Semantic information about dependencies

1714 words defined, the language coverage is:

English	94.0%
German	69.8%
Chinese (simplified)	8.5%
Romanian	3.5%
...	
Afrikaans	0.0%

. . . let's use GF!

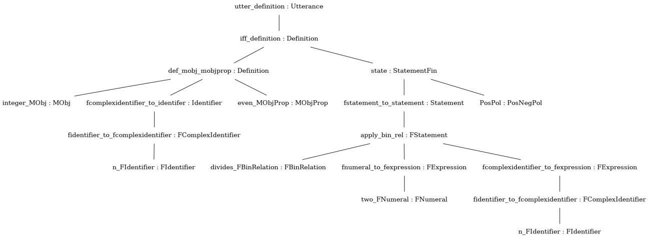
Example adapted from [GIJ+16]:

"A non-empty graph G is said to be **connected***, if any two of its nodes are linked by a path in G."*

"Ein nicht-leerer Graph G heißt **zusammenhängend***, wenn je zwei seiner Knoten durch einen Weg in G verbunden sind."*

"An integer n is called even iff $2|n$*."*

$\Downarrow$ **parse**

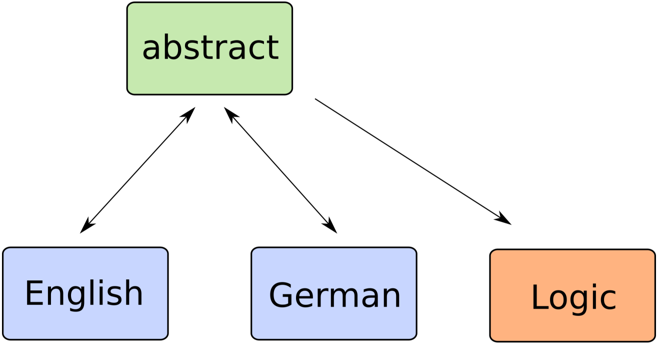


$\Downarrow$ **linearize**

"Eine ganze Zahl n heißt gerade genau dann, wenn $2|n$*."*

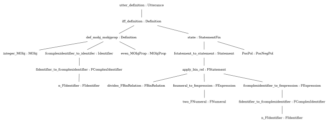
- Let's try to formalize sentences
- Example: $\forall n.\mathbf{int}(n) \Rightarrow (\mathbf{even}(n) \Leftrightarrow \mathbf{divides}(2, n))$
- I will present two different approaches

The formal representation is just another language:



"An integer n is called even iff $2|n$*."*

$\Downarrow$ **parse**



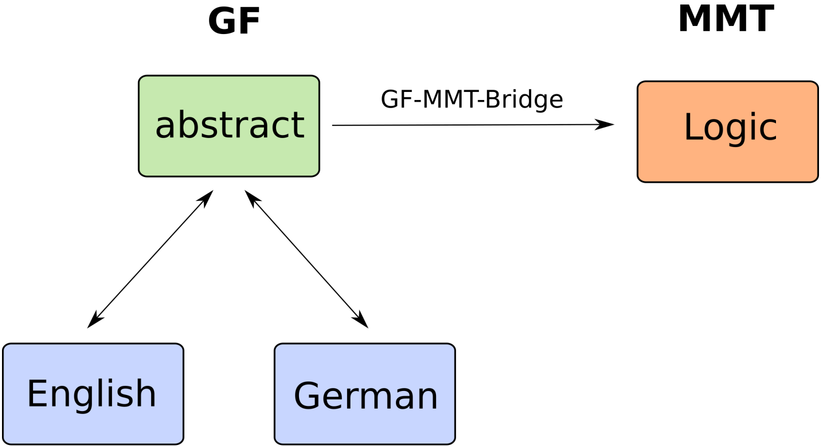
$\Downarrow$ **linearize**

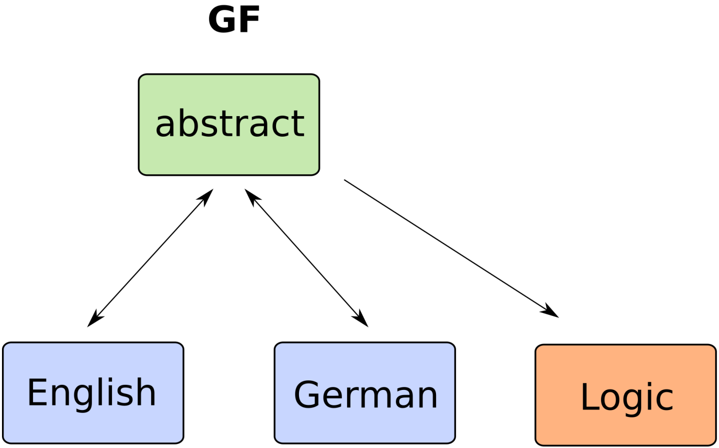
$$(\forall n.((\lambda x.\mathbf{int}(x))n) \Rightarrow ((\lambda x.\mathbf{even}(x))n \Leftrightarrow \mathbf{divides}(2, n)))$$

$\Downarrow$ external simplifier

$$\forall n.\mathbf{int}(n) \Rightarrow (\mathbf{even}(n) \Leftrightarrow \mathbf{divides}(2, n))$$

Use an external system (MMT) for the logic:




GF
abstract
English
German
Logic

There are formulas in the text:

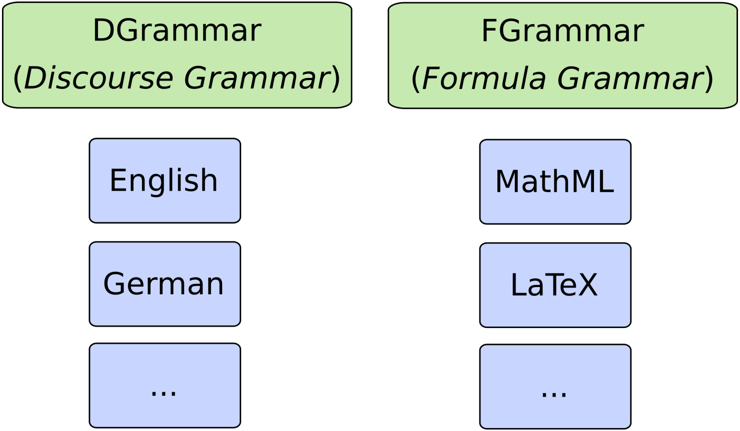
"A sequence $x = \{x_n\}_{n=1}^{\infty} \in l^{\infty}(V)$ *is called quasi-almost convergent to* $v \in V$ *if* $\forall L \in \Pi$, $L(x - \widetilde{v}) = 0$.*"* [5]

There are formulas in the text:

"A sequence $x = \{x_n\}_{n=1}^{\infty} \in l^{\infty}(V)$ *is called quasi-almost convergent to* $v \in V$ *if* $\forall L \in \Pi$, $L(x - \widetilde{v}) = 0$*."* [5]

There is text in the formulas:

"$H(P) = \{\alpha \in \mathbb{N}_0 \mid \textit{there exists a rational function } f \textit{ on } C \textit{ with } (f)_\infty = \alpha P\}$" [2]



Depends on language

Depends on representation

Example:

"$x^2 + 1$ is greater than or equal to $\sqrt{x^2 + 1}$."

There is No Standard Formula Representation

Example:

"*$x^2 + 1$ is greater than or equal to $\sqrt{x^2 + 1}$.*"

First Idea:

```
x^2 + 1 is greater than or equal to SQRT(x^2 + 1).
```

Example:

"$x^2 + 1$ *is greater than or equal to* $\sqrt{x^2 + 1}$*."*

First Idea:

```
x^2 + 1 is greater than or equal to SQRT(x^2 + 1).
```

LaTeX:

```
$x^2 + 1$ is greater than or equal to $\sqrt{x^2 + 1}$.
```

Formula: $x^2 + 1$

Presentation MathML

```
<math>
    <mrow>
        <msup>
            <mi>x</mi>
            <mn>2</mn>
        </msup>
        <mo>+</mo>
        <mn>1</mn>
    </mrow>
</math>
```

Content MathML

```
<math>
  <apply>
    <plus />
    <apply>
      <power />
      <ci>x</ci>
      <cn>2</cn>
    </apply>
    <cn>1</cn>
  </apply>
</math>
```

Formula: $x^2 + 1$

Presentation MathML

```
<math>
    <mrow>
        <msup>
            <mi>x</mi>
            <mn>2</mn>
        </msup>
        <mo>+</mo>
        <mn>1</mn>
    </mrow>
</math>
```

Content MathML

```
<math>
  <apply>
    <plus />
    <apply>
      <power />
      <ci>x</ci>
      <cn>2</cn>
    </apply>
    <cn>1</cn>
  </apply>
</math>
```

$\LaTeX \xrightarrow{\text{LaTeXML}} \text{MathML}$ (see [Mil])

DGrammar
(*Discourse Grammar*)

English

German

...

Depends on language

FGrammar
(*Formula Grammar*)

MathML

LaTeX

...

Depends on representation

Definitions:

"An integer n is called **even** *iff $2|n$."*

Declarations:

"Let $\widetilde{H}$ be a numerical semigroup." [2]

Statements:

"For any graph G, $\alpha_1(G) \leq \frac{nb(G)}{4}$." [4]

Mathematical Objects:

"An integer n is called **even** *iff* $2|n$.*"*

"Let $\widetilde{H}$ *be a numerical semigroup."* [2]

"For any graph G, $\alpha_1(G) \leq \frac{nb(G)}{4}$.*"* [4]

Mathematical Properties:

"An integer n is called **even** *iff* $2|n$.*"*

"If $x = \{x_n\}_{n=1}^{\infty} \in l^{\infty}(V)$ *is strongly almost convergent in* V, *then [...]"* [5]

Remember: Grammar Architecture

DGrammar
(*Discourse Grammar*)

English

German

...

Depends on language

FGrammar
(*Formula Grammar*)

MathML

LaTeX

...

Depends on representation

Formula Statement:

"An integer n is called **even** *iff* $2|n$*."*

"Since $uv \notin A$*, this implies that* $uz \in A$ *and* $vz \in A$*"* [4]

Objects:

"$\bigcup_{i\in I} O_i$ *is open in* Y*."*

"The stabilizer in $G^{\mathbb{C}}$ *for* p *is an algebraic group."* [1]

(Restricted) Identifier:

"An integer n *is called* **even** *iff 2|n."*

"Let $r \geq 2$ *be an integer."*

Numerals: 2, 19, ...

Identifiers: n, φ, G, ...

Binary Operators: $+$, $\cap$, ...

Binary Relations: $\geq$, $\in$, ...

Example: $n^2 + 19 \geq n^2$

See also [Gin11]

Example: "$n > 1$"

```
-- abstract grammar:
greater_than : FExpression -> FExpression -> FStatement;

-- concrete grammar:
greater_than a b = a ++ ">" ++ b;
```

Parsing a Formula - Binary Relations

Example: "$n > 1$"

```
-- abstract grammar:
greater_than : FExpression -> FExpression -> FStatement;

-- concrete grammar:
greater_than a b = a ++ ">" ++ b;
```

Better way:

```
-- abstract grammar:
apply_bin_rel : FBinRelation -> FExpression ->
        FExpression -> FStatement;

-- concrete grammar could be:
apply_bin_rel rel a b = a ++ rel ++ b;
```

Parsing a Formula - Binary Relations

The first solution couldn't handle the following cases nicely:

- "$1 < m < n$"
- "$0 \leq r < 1$"
- "$0 = t_0 < t_1 < \ldots < t_n = 1$"

Parsing a Formula - Binary Relations

The first solution couldn't handle the following cases nicely:

- "$1 < m < n$"
- "$0 \leq r < 1$"
- "$0 = t_0 < t_1 < \ldots < t_n = 1$"

Cases like "$0 \leq r < 1$" are easy with better approach:

```
-- abstract grammar:
apply_tern_rel : FBinRelation -> FBinRelation->
        FExpression -> FExpression -> FExpresion ->
        FStatement;
```

```
-- concrete grammar:
apply_tern_rel rel1 rel2 a b c = a ++ rel1 ++ b ++
                                    rel2 ++ c;
```

Formula Statement:

"An integer n is called **even** *iff* $2|n$*."*

"Since $uv \notin A$*, this implies that* $uz \in A$ *and* $vz \in A$*"* [4]

Objects:

"$\bigcup_{i\in I} O_i$ *is open in* Y*."*

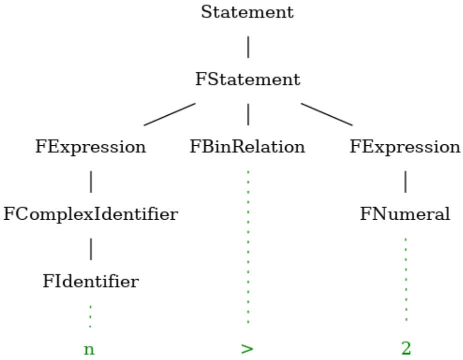
"The stabilizer in $G^{\mathbb{C}}$ *for* p *is an algebraic group."* [1]

(Restricted) Identifier:

"An integer n *is called* **even** *iff* $2|n$*."*

"Let $r \geq 2$ *be an integer."*

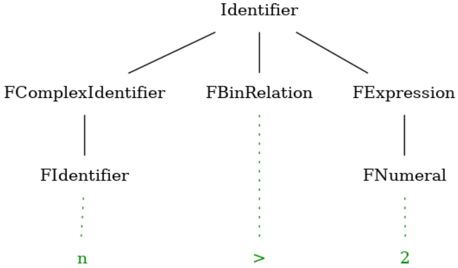
"we know that $n > 2$*"*



In logic: $\mathbf{greater}(n, 2)$

Example: "$n > 2$" as Identifier

"there is an integer $n > 2$ such that..."



$$(\exists n.(\lambda x.\mathbf{int}(x))n \wedge \mathbf{greater}(n, 2) \wedge (\ldots))$$

$\Downarrow$ external simplifier

$$\exists n.\mathbf{int}(n) \wedge \mathbf{greater}(n, 2) \wedge \ldots$$

We need to extract "n" from "$n > 2$":

```
-- Identifier record (simplified)
Identifier = {
    formula : Str;  -- the restriction, e.g. greater(n, 2)
    core :  Str;    -- the identifier, e.g. n
};
```

Example: “$n > 2$” as Identifier

We need to extract “n” from “$n > 2$”:

```
-- Identifier record (simplified)
Identifier = {
    formula : Str;   -- the restriction, e.g. greater(n, 2)
    core :  Str;     -- the identifier, e.g. n
};
```

```
-- example usage
fcid_fbinrel_fexpr_to_identifier a r1 b = {
    formula = r1 ++ "(" ++ a ++ "," ++ b ++ ")";
    core = a;
};
```

```
-- for statements like (∃n.(λx.int(x))n ∧ greater(n,2))
exists_statement obj id = inp("∃" ++ id.core ++ "."
        ++ lwrap(obj) ++ id.core ++ and_formula(id));
```

Examples:

- *"integer"*
- *"an even integer"*
- *"There is a bijective map from $(0,1)$ to $\mathbb{R}$"*
- *"There is a bijective map f from $(0,1)$ to $\mathbb{R}$"*
- *"Let C be a complete nonsingular irreducible curve over an algebraically closed field k of characteristic 0"* [2]

```
exists_suchthat :
    PosNegPol          -- is/isn't
    -> MObj            -- integer
    -> Identifier      -- n
    -> StatementFin    -- ...
    -> StatementFin;
```

"there is an integer n such that ..."

$\exists n.\mathbf{int}(n) \wedge \ldots$

```
exists_suchthat :
    PosNegPol               -- is/isn't
    -> MObj                 -- positive even integer
    -> Identifier           -- n
    -> StatementFin         -- ...
    -> StatementFin;
```

"there is a positive even integer n such that . . . "

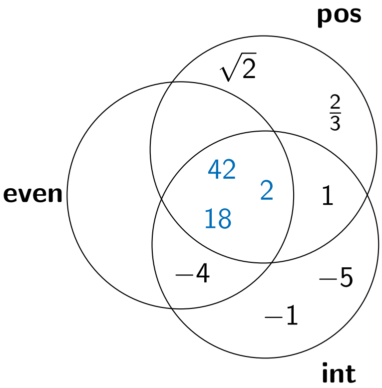
$\exists n.\mathbf{pos}(n) \wedge \mathbf{even}(n) \wedge \mathbf{int}(n) \wedge \ldots$

```
exists_suchthat :
    PosNegPol               -- is/isn't
    -> MObj                 -- positive even integer
    -> Identifier           -- n
    -> StatementFin         -- ...
    -> StatementFin;
```

"there is a positive even integer n such that . . . "

$\exists n.\mathbf{pos}(n) \wedge \mathbf{even}(n) \wedge \mathbf{int}(n) \wedge \ldots$

$\exists n.(\lambda x.\mathbf{pos}(x) \wedge \mathbf{even}(x) \wedge \mathbf{int}(x))n \wedge \ldots$



- *"Positive even integers"* are the intersection of integers, even things and positive things
- This doesn't work for every adjective!

"An integer n is called even iff $2|n$."

Idea: $\mathbf{even}(n) \Leftrightarrow \mathbf{divides}(2, n)$

"An integer n is called even iff $2|n$*."*

Idea: $\mathbf{even}(n) \Leftrightarrow \mathbf{divides}(2, n)$

With quantifier: $\forall n.\mathbf{int}(n) \Rightarrow (\mathbf{even}(n) \Leftrightarrow \mathbf{divides}(2, n))$

What's generated:
$(\forall n.((\lambda x.\mathbf{int}(x))n) \Rightarrow ((\lambda x.\mathbf{even}(x))n \Leftrightarrow \mathbf{divides}(2, n)))$

"A positive integer n is called prime, iff there is no integer $1 < m < n$ such that $m|n$"

Translation to (from) German:

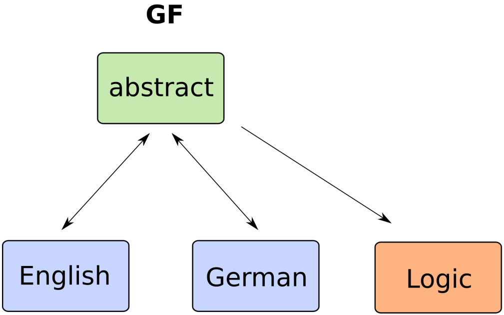
"Eine positive ganze Zahl n ist prim genau dann, wenn es keine ganze Zahl $1 < m < n$ gibt, sodass $m|n$"

Formalization:

$$(\forall n.((\lambda x.\mathbf{pos}(x) \wedge \mathbf{int}(x))n) \Rightarrow ((\lambda x.\mathbf{prime}(x))n \Leftrightarrow (\neg\exists m.(\lambda x.\mathbf{int}(x))m \wedge \mathbf{less}(1, m) \wedge \mathbf{less}(m, n) \wedge (\mathbf{divides}(m, n)))))$$

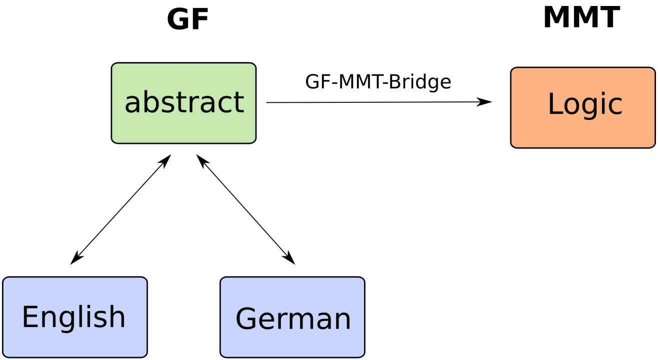
$\Downarrow$ external simplifier

$$\forall n.\mathbf{pos}(n) \wedge \mathbf{int}(n) \Rightarrow (\mathbf{prime}(n) \Leftrightarrow \neg\exists m.\mathbf{int}(m) \wedge \mathbf{divides}(m, n) \wedge \mathbf{less}(1, m) \wedge \mathbf{less}(m, n))$$



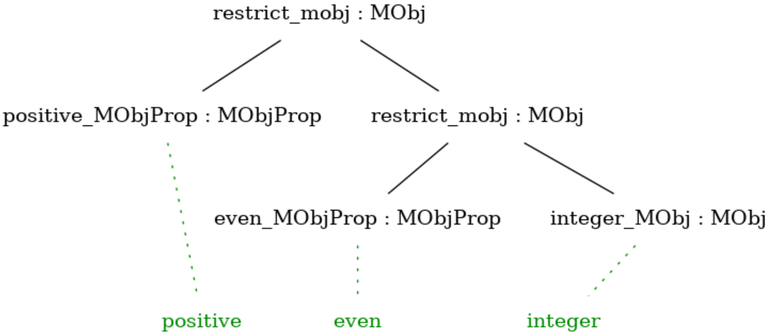
- **+** Parsing and logic generation in one tool (GF)
- **–** Grammar engineering gets complicated
- **–** We need an external tool for simplification and reasoning

Second Approach: Semantics Modelling in MMT



What is MMT?

- *Meta Meta Tool*
- Foundation-independent
- A lot of features for mathematical knowledge management
- See e.g. [Rab16]
- We'll focus only on a small "slice" of MMT



$\lambda x.\mathbf{pos}(x) \wedge \mathbf{even}(x) \wedge \mathbf{int}(x)$

$\lambda x.\mathbf{pos}(x) \wedge \mathbf{even}(x) \wedge \mathbf{int}(x)$

Type declarations for atoms:

even_MObjProp : $\iota \longrightarrow o$ |
positive_MObjProp : $\iota \longrightarrow o$ |
integer_MObj : $\iota \longrightarrow o$ |

$\lambda x.\mathbf{pos}(x) \wedge \mathbf{even}(x) \wedge \mathbf{int}(x)$

Type declarations for atoms:

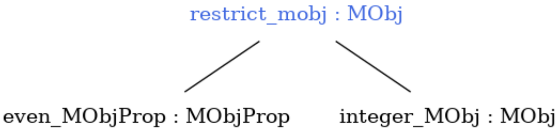
even_MObjProp : $\iota \longrightarrow o$
positive_MObjProp : $\iota \longrightarrow o$
integer_MObj : $\iota \longrightarrow o$

Better: Types for **`cat`**s in MMT:

MObj : type | = $\iota \longrightarrow o$
MObjProp : type | = $\iota \longrightarrow o$

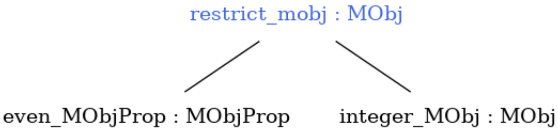
even_MObjProp : MObjProp
positive_MObjProp : MObjProp
integer_MObj : MObj

Grammar Nodes: **`restrict_mobj`**



Goal: **restrict even integer** $= \lambda x.$**even**$(x) \wedge$ **integer**(x)

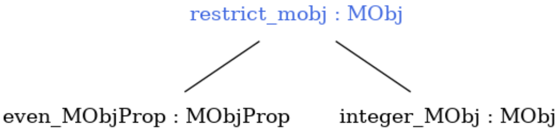
Grammar Nodes: `restrict_mobj`



Goal: **restrict even integer** $= \lambda x.$**even**$(x) \wedge$ **integer**(x)

restrict_MObj : MObjProp ⟶ MObj ⟶ MObj |
  = [mprop,mobj] [x] mprop(x) ∧ mobj(x) |

Grammar Nodes: `restrict_mobj`



Goal: **restrict even integer** $= \lambda x.$**even**$(x) \wedge$ **integer**(x)

restrict_MObj : MObjProp ⟶ MObj ⟶ MObj |
= [mprop,mobj] [x] mprop(x) ∧ mobj(x) |

We can map the GF tree to an MMT term:

`restrict_mobj even_MObjProp integer_MObj`

$\Downarrow$ GF-MMT-Bridge

restrict_MObj even_MObjProp integer_MObj

```
abstract Cats = {
  cat
    MObj;
    MObjProp;
}
```

```
theory Cats : ur:?LF =
   include ?FOL ❙

   MObj : type | = ι ⟶ o ❙
   MObjProp : type | = ι ⟶ o ❙
❚
```

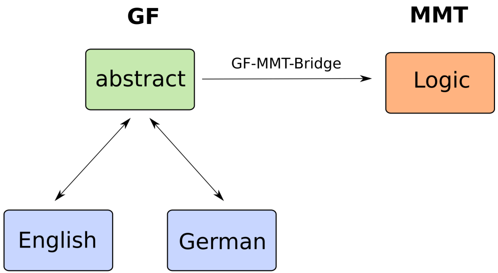
```
abstract Lexicon = Cats ** {
  fun
    even_MObjProp : MObjProp;
    positive_MObjProp :
                        MObjProp;
    integer_MObj : MObj;
}
```

```
theory Lexicon : ur:?LF =
   include ?Cats ❙

   even_MObjProp : MObjProp ❙
   positive_MObjProp : MObjProp ❙
   integer_MObj : MObj ❙
❚
```

```
abstract Grammar = Cats ** {
  fun
    restrict_MObj :
        MObjProp -> MObj ->
        MObj;
}
```

```
theory Grammar : ur:?LF =
   include ?Cats ❙
   restrict_MObj
      : MObjProp ⟶ MObj ⟶ MObj |
      = [mprop,mobj] [x]
                  (mprop x) ∧ (mobj x) ❙
❚
```



GF	(= *grammar development framework*)
+ MMT	(= *logic development framework*)
???	(= *semantics development framework*)

```
theory Cats : ur:?LF =
   include ?FOL ❙

   MObj : type | = ι ⟶ o ❙
   MObjProp : type | = ι ⟶ o ❙
❚
```

$\rightarrow$ Where does FOL come from?

```
theory Cats : ur:?LF =
  include ?FOL ❙

  MObj : type | = ι ⟶ o ❙
  MObjProp : type | = ι ⟶ o ❙
❚
```

→ Where does FOL come from?

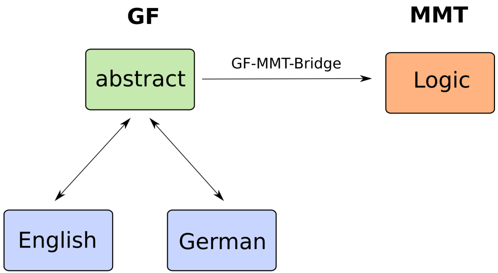
```
theory FOL : ur:?LF =
  include ?PL ❙
  // type for individuals ❙
  ind : type | # ι ❙

  // add quantifiers ❙
  forall : ( ι ⟶ o ) ⟶ o | # ∀ 1 ❙
  exists : ( ι ⟶ o ) ⟶ o | # ∃ 1 | = [p] ¬ ∀ [x] ¬ (p x) ❙
❚
```

```
theory PL : ur:?LF =
  // declare a type for propositions and introduce o as notation ❙
  prop : type ❘ # o ❙

  // declare basic logical operations ❙
  negation : o ⟶ o ❘ # ¬ 1 prec 30 ❙
  conjunction : o ⟶ o ⟶ o ❘ # 1 ∧ 2 prec 15 ❙

  // we can define other operations through ¬ and ∧ ❙
  disjunction : o ⟶ o ⟶ o ❘ = [a,b] ¬ (¬ a ∧ ¬ b) ❘ # 1 ∨ 2 prec 10 ❙
  implication : o ⟶ o ⟶ o ❘ = [a,b] ¬ a ∨ b ❘ # 1 ⇒ 2 prec 8 ❙
  equivalence : o ⟶ o ⟶ o ❘ = [a,b] (a ⇒ b) ∧ (b ⇒ a) ❘ # 1 ⇔ 2 prec 5 ❙
❚
```



GF	(= *grammar development framework*)
+ MMT	(= *logic development framework*)
???	(= *semantics development framework*)

Very hard!!

See also [Zin04] and [Wol13].

"From $A \subset B$ and $B \subset A$ it follows that $A = B$."

"$\subset$" might refer to "$\subseteq$" or to "$\subsetneq$"

$\rightarrow$ we get at least $2 \cdot 2 = 4$ parse trees

$\rightarrow$ we can discard some of them:

Example: Interpreting both "$\subset$" as "$\subsetneq$":

$(\subseteq (A, B) \land A \neq B) \land (\subseteq (B, A) \land A \neq B), \;\; A = B$

"an integer n is called even iff $2|n$*"*

even : int ⟶ o | = [n] div(2, n) |

- Math Linguistics/Translation
- Semantics Development Framework:

GF	*(= grammar development framework)*
+ MMT	*(= logic development framework)*
???	*(= semantics development framework)*

- Math Linguistics/Translation
- Semantics Development Framework:

GF	*(= grammar development framework)*
+ MMT	*(= logic development framework)*
???	*(= semantics development framework)*

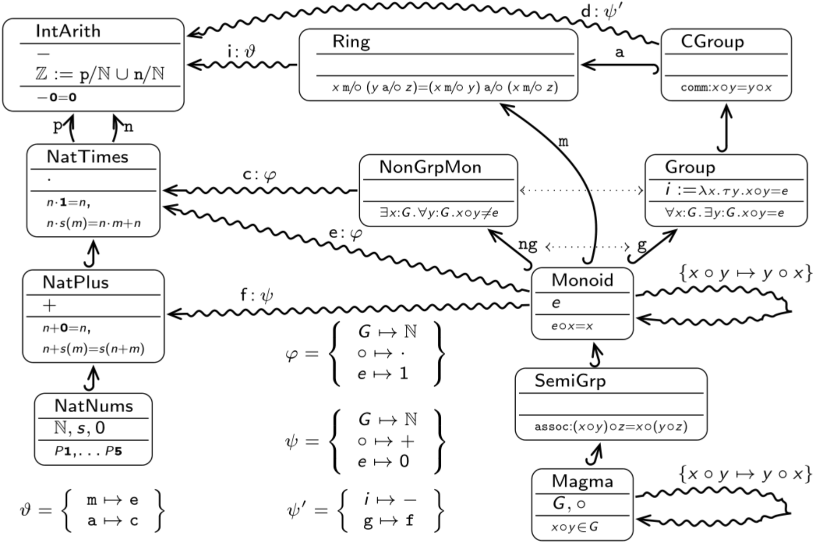
Advertisement: **SIGMathLing**
Special **I**nterest **G**roup on **Math**s **Ling**uistics
https://sigmathling.kwarc.info/

- *arXMLiv* corpus: 1,232,186 HTML5 scientific documents from the arXiv.org
- *GloVe* word embeddings for mathematics
- ...

```
theory PLSemantics : ur:?LF =
   include ?PLSyntax ❙

   ded : o ⟶ type | # ⊦ 1 ❙   // jder  not jvdash ❙

   trueIn : ⊦ ⊤ ❙
   andIn : {A,B} ⊦ A ⟶ ⊦ B ⟶ ⊦ A ∧ B | # andI 3 4 ❙
   negEl : {A} ⊦ A ⟶ ⊦ ¬ A ⟶ ⊦ ⊥ | # negE 2 3 ❙
   falseEl : {A} ⊦ ⊥ ⟶ ⊦ A | # falseE 2 ❙
   andElL : {A,B} ⊦ A ∧ B ⟶ ⊦ A | # andEL 3 ❙
```


IntArith
−
ℤ := p/ℕ ∪ n/ℕ
−0=0
Ring
x m/o (y a/o z)=(x m/o y) a/o (x m/o z)
CGroup
comm:x∘y=y∘x
d : ψ′
i : ϑ
a
p
n
NatTimes
·
n·1=n,
n·s(m)=n·m+n
c : φ
NonGrpMon
∃x:G.∀y:G.x∘y≠e
Group
i :=λx.ιy.x∘y=e
∀x:G.∃y:G.x∘y=e
m
e : φ
ng
g
NatPlus
+
n+0=n,
n+s(m)=s(n+m)
f : ψ
Monoid
e
e∘x=x
{x ∘ y ↦ y ∘ x}
φ = { G ↦ ℕ, ∘ ↦ ·, e ↦ 1 }
SemiGrp
assoc:(x∘y)∘z=x∘(y∘z)
NatNums
ℕ, s, 0
P1, . . . P5
ψ = { G ↦ ℕ, ∘ ↦ +, e ↦ 0 }
Magma
G, ∘
x∘y∈G
{x ∘ y ↦ y ∘ x}
ϑ = { m ↦ e, a ↦ c }
ψ′ = { i ↦ −, g ↦ f }

Deyan Ginev, Mihnea Iancu, Constantin Jucovshi, Andrea Kohlhase, Michael Kohlhase, Akbar Oripov, Jürgen Schefter, Wolfram Sperber, Olaf Teschke, and Tom Wiesing.
The SMGloM project and system. towards a terminology and ontology for mathematics.
In Gert-Martin Greuel, Thorsten Koch, Peter Paule, and Andrew Sommese, editors, *Mathematical Software - ICMS 2016 - 5th International Congress*, volume 9725 of *LNCS*. Springer, 2016.

Deyan Ginev.
The structure of mathematical expressions.
Master's thesis, Jacobs University Bremen, Bremen, Germany, August 2011.

Bruce Miller.
`LaTeXML`: A LaTeX to XML converter.

Florian Rabe.
Mmt: A foundation-independent approach to formal knowledge.
2016.

Magdalena A. Wolska.
Student's Language in Computer-Assisted Tutoring of Mathematical Proofs.
PhD thesis, ComputerLinguistik, Saarland University, 2013.

Claus Zinn.
Understanding Informal Mathematical Discourse.
PhD thesis, Technischen Fakultät der Universität Erlangen-Nürnberg, 2004.

Azad, H., and Biswas, I.
A note on real algebraic groups, 2013.

Harui, T., Komeda, J., and Ohbuchi, A.
The weierstrass semigroups on double covers of genus two curves, 2013.

Johst, K., and Person, Y.
On the multicolor ramsey number of a graph with m edges, 2013.

Puleo, G. J.
On a conjecture of erdos, gallai, and tuza, 2014.

You, C.
Simplified and equivalent characterizations of banach limit functional and strong almost convergence, 2009.