An HTML/CSS Schema for TEX Primitives

Generating High-Quality Responsive HTML from generic TEX

Dennis Müller

TUG, 14. July 2023

RusTeX In A Nutshell

This talk concerns RusTeX – a (somewhat experimental) TeX engine implemented in Rust for converting TeX documents to HTML.
(or alternatively: Just a fun experiment regarding "boxes" in TeX vs in CSS)

RusTeX attemps to

- mimic the behaviour of `pdflatex` as closely as possible

⇒ (ideally) full coverage

- avoid implementing dedicated support for (otherwise expandable) LaTeX macros and packages

⇒ maps TeX *primitives* directly to HTML with adequate CSS

- mimic the optics of the PDF in HTML (Modulo sensible differences)

⇒ Can we mimic the behavior of elaborate macros/environments on the basis of TeX primitives alone, using HTML and CSS? (Surprisingly well!)

`https://github.com/slatex/RusTeX/releases`

...why do we need *yet another* TEX to HTML converter?

The sTEX Idea: (M. Kohlhase) Represent both the *layouting* and the *(flexi-)formal semantics* of (primarily) mathematical statements in a *single* document, via semantic annotations on (otherwise) arbitrary text.
⇒ allows for formal knowledge management services (e.g. disambiguation, type checking, etc.) (See TUG 2022)

Example: Instead of "`The series $\sum_{i=0}^\infty a_i$ converges`", write

```
\importmodule{calculus?convergence}
\vardef{seqa}[def=...]{...}
The \symname{series} $\series{\seqa}$ \symname{converges}
```

⇒ same output, but with computer-actionable formal semantics!
⇒ modularity allows for reusability of concepts, definitions, notations, etc.

sTeX3

Pre-2021: Dedicated sTeX document classes, additional software required, convertable to OMDoc via an elaborate and invasive LaTeXML plugin, import OMDoc to MMT for actual services.
⇒ complicated pipeline with several software components, not integrable in arbitrary LaTeX documents, separate opaque representations for presentation (PDF) and formal content (OMDoc), bitrot.
⇒ by 2017 practically unusable.

≈**2020:** *We can do better!* We want:

- A single LaTeX package, compatible with arbitrary other packages/document classes (can write papers, books, lecture notes/slides, talks, etc.)
- No additional software required, no `--shell-escape` (except for formal knowledge management services)
- Have a single, transparent, and integrable representation for both presentation and formal content

⇒ Convert to HTML, add semantic information as XML attributes

⇒ sTeX3

So we need...

A TEX-to-HTML converter that

- can (ideally) deal with arbitrary valid LATEX files,
- can preserve semantic information via XML attributes in the HTML output,
- is (virtually) guaranteed to succeed for any valid LATEX file. (even if it looks broken – at least the semantic information is there!)

Also: research project VoLL-KI approved by DFG – time is of the essence!

...and the existing converters did not work for that

(Statistics on my private corpus of tex files attempted, but would be misleadingly inaccurate)
(≈60% success with TEX4ht)

Compare:

PDF

Theorem 2.3. *The freely generated Heyting algebra $\mathcal{H}_A$ is the* Tarski-Lindenbaum algebra *of* IpC *over A. This means for any formulae $a, b \in \mathcal{F}(A)$ we have $[a] \sim [b]$ iff $\vdash_I a \leftrightarrow b$.*

Proof. We have to show the following:

1. For every axiom $[t_1] \doteq [t_2]$ of T_H we have $\vdash_I t_1 \leftrightarrow t_2$.
2. *Modus Ponens:* If $[\varphi] = 1$ and $[\varphi] \hookrightarrow [\psi] = 1$, then $[\psi] = 1$.
3. For every axiom $\varphi(\psi_1, \psi_2, \psi_3)$ of IpC we have $\mathcal{H}_A \models \forall x, y, z[\varphi(x, y, z)] \doteq 1$.

Proof of 1. We have already proven, that in IpC conjunction is commutative and associative, the same holds for disjunction (follows immediately by IpC5, IpC6 and Modus Ponens). The axioms for 1 and 0 hold by definition of $\top$ and $\bot$.

Idempotence: By IpC4 and IpC5 we have $\vdash_I x \to (x \vee x)$ and $\vdash_I (x \wedge x) \to x$. IpC8 gives us

$$\vdash_I (x \to x) \to ((x \to x) \to ((x \vee x) \to x))$$

and therefore $\vdash_I (x \vee x) \to x$. By IpC3 we have $\vdash_I x \to (x \to (x \wedge x))$ and thus $\vdash_I x \to (x \wedge x)$.

Absorption: We have $\vdash_I x \wedge (x \vee y) \to x$ (IpC4), $\vdash_I x \to x \vee y$ (IpC5), $\vdash_I x \to x$ and thus $\vdash_I x \to (x \wedge (x \vee y))$. The converse is IpC4.

TEX4ht

Theorem 2.3. *The freely generated Heyting algebra H_A is the* Tarski-Lindenbaum algebra *of* IpC *over A. This means for any formulae $a, b \in F(A)$ we have $[a] \sim [b]$ iff $\vdash_I a \leftrightarrow b$.*

Proof. We have to show the following:

1. For every axiom $[t_1] \doteq [t_2]$ of T_H we have $\vdash_I t_1 \leftrightarrow t_2$.
2. *Modus Ponens:* If $[\varphi] = 1$ and $[\varphi] \hookrightarrow [\psi] = 1$, then $[\psi] = 1$.
3. For every axiom $\varphi(\psi_1, \psi_2, \psi_3)$ of IpC we have $H_A \models \forall\ x, y, z[\varphi(x, y, z)] \doteq 1$.

- We have already proven, that in IpC conjunction is commutative and associative, the same holds for disjunction (follows immediately by IpC5, IpC6 and Modus Ponens). The axioms for 1 and 0 hold by definition of $\top$ and $\bot$.
  - By IpC4 and IpC5 we have $\vdash_I x \to (x \vee x)$ and $\vdash_I (x \wedge x) \to x$. IpC8 gives us
    $$\vdash_I (x \to x) \to ((x \to x) \to ((x \vee x) \to x))$$
    and therefore $\vdash_I (x \vee x) \to x$. By IpC3 we have $\vdash_I x \to (x \to (x \wedge x))$ and thus $\vdash_I x \to (x \wedge x)$.
  - We have $\vdash_I x \wedge (x \vee y) \to x$ (IpC4), $\vdash_I x \to x \vee y$ (IpC5), $\vdash_I x \to x$ and thus $\vdash_I x \to (x \wedge (x \vee y))$. The converse is IpC4.

Compare:

LaTeXML

Theorem 2.3. *The freely generated Heyting algebra $\mathcal{H}_A$ is the* Tarski-Lindenbaum algebra *of IpC over A. This means for any formulae $a, b \in \mathcal{F}(A)$ we have $[a] \sim [b]$ iff $\vdash_I a \leftrightarrow b$.*

Proof. We have to show the following:

1. For every axiom $[t_1] \triangleq [t_2]$ of T_H we have $\vdash_I t_1 \leftrightarrow t_2$.
2. *Modus Ponens:* If $[\varphi] = 1$ and $[\varphi] \hookrightarrow [\psi] = 1$, then $[\psi] = 1$.
3. For every axiom $\varphi(\psi_1, \psi_2, \psi_3)$ of IpC we have $\mathcal{H}_A \models \forall x, y, z[\varphi(x, y, z)] \triangleq 1$.

Proof of 1. We have already proven, that in IpC conjunction is commutative and associative, the same holds for disjunction (follows immediately by IpC5, IpC6 and Modus Ponens). The axioms for 1 and 0 hold by definition of $\top$ and $\bot$.

Idempotence: By IpC4 and IpC5 we have $\vdash_I x \to (x \vee x)$ and $\vdash_I (x \wedge x) \to x$. IpC8 gives us

$$\vdash_I (x \to x) \to ((x \to x) \to ((x \vee x) \to x))$$

and therefore $\vdash_I (x \vee x) \to x$. By IpC3 we have $\vdash_I x \to (x \to (x \wedge x))$ and thus $\vdash_I x \to (x \wedge x)$.

Absorption: We have $\vdash_I x \wedge (x \vee y) \to x$ (IpC4), $\vdash_I x \to x \vee y$ (IpC5), $\vdash_I x \to x$ and thus $\vdash_I x \to (x \wedge (x \vee y))$. The converse is IpC4.

Compare:

PDF

Theorem 2.3. *The freely generated Heyting algebra $\mathcal{H}_A$ is the* Tarski-Lindenbaum algebra *of* lpC *over A. This means for any formulae $a, b \in \mathcal{F}(A)$ we have $[a] \sim [b]$ iff $\vdash_I a \leftrightarrow b$.*

Proof. We have to show the following:

1. For every axiom $[t_1] \doteq [t_2]$ of T_H we have $\vdash_I t_1 \leftrightarrow t_2$.
2. *Modus Ponens:* If $[\varphi] = 1$ and $[\varphi] \hookrightarrow [\psi] = 1$, then $[\psi] = 1$.
3. For every axiom $\varphi(\psi_1, \psi_2, \psi_3)$ of lpC we have $\mathcal{H}_A \models \forall x, y, z[\varphi(x, y, z)] \doteq 1$.

Proof of 1. We have already proven, that in lpC conjunction is commutative and associative, the same holds for disjunction (follows immediately by lpC5, lpC6 and Modus Ponens). The axioms for 1 and 0 hold by definition of $\top$ and $\bot$.

Idempotence: By lpC4 and lpC5 we have $\vdash_I x \to (x \vee x)$ and $\vdash_I (x \wedge x) \to x$. lpC8 gives us

$$\vdash_I (x \to x) \to ((x \to x) \to ((x \vee x) \to x))$$

and therefore $\vdash_I (x \vee x) \to x$. By lpC3 we have $\vdash_I x \to (x \to (x \wedge x))$ and thus $\vdash_I x \to (x \wedge x)$.

Absorption: We have $\vdash_I x \wedge (x \vee y) \to x$ (lpC4), $\vdash_I x \to x \vee y$ (lpC5), $\vdash_I x \to x$ and thus $\vdash_I x \to (x \wedge (x \vee y))$. The converse is lpC4.

LaTeXML (with `max-width` and fonts)

Theorem 2.3. *The freely generated Heyting algebra $\mathcal{H}_A$ is the* Tarski-Lindenbaum algebra *of* lpC *over A. This means for any formulae $a, b \in \mathcal{F}(A)$ we have $[a] \sim [b]$ iff $\vdash_I a \leftrightarrow b$.*

Proof. We have to show the following:

1. For every axiom $[t_1] \doteq [t_2]$ of T_H we have $\vdash_I t_1 \leftrightarrow t_2$.
2. *Modus Ponens:* If $[\varphi] = 1$ and $[\varphi] \hookrightarrow [\psi] = 1$, then $[\psi] = 1$.
3. For every axiom $\varphi(\psi_1, \psi_2, \psi_3)$ of lpC we have $\mathcal{H}_A \models \forall x, y, z[\varphi(x, y, z)] \doteq 1$.

Proof of 1. We have already proven, that in lpC conjunction is commutative and associative, the same holds for disjunction (follows immediately by lpC5, lpC6 and Modus Ponens). The axioms for 1 and 0 hold by definition of $\top$ and $\bot$.

Idempotence: By lpC4 and lpC5 we have $\vdash_I x \to (x \vee x)$ and $\vdash_I (x \wedge x) \to x$. lpC8 gives us

$$\vdash_I (x \to x) \to ((x \to x) \to ((x \vee x) \to x))$$

and therefore $\vdash_I (x \vee x) \to x$. By lpC3 we have $\vdash_I x \to (x \to (x \wedge x))$ and thus $\vdash_I x \to (x \wedge x)$.

Absorption: We have $\vdash_I x \wedge (x \vee y) \to x$ (lpC4), $\vdash_I x \to x \vee y$ (lpC5), $\vdash_I x \to x$ and thus $\vdash_I x \to (x \wedge (x \vee y))$. The converse is lpC4.

Compare:

PDF

Theorem 2.3. *The freely generated Heyting algebra $\mathcal{H}_A$ is the* Tarski-Lindenbaum algebra *of IpC over A. This means for any formulae $a, b \in \mathcal{F}(A)$ we have $[a] \sim [b]$ iff $\vdash_I a \leftrightarrow b$.*

Proof. We have to show the following:

1. For every axiom $[t_1] \doteq [t_2]$ of T_H we have $\vdash_I t_1 \leftrightarrow t_2$.
2. *Modus Ponens:* If $[\varphi] = 1$ and $[\varphi] \hookrightarrow [\psi] = 1$, then $[\psi] = 1$.
3. For every axiom $\varphi(\psi_1, \psi_2, \psi_3)$ of IpC we have $\mathcal{H}_A \models \forall x, y, z[\varphi(x, y, z)] \doteq 1$.

Proof of 1. We have already proven, that in IpC conjunction is commutative and associative, the same holds for disjunction (follows immediately by IpC5, IpC6 and Modus Ponens). The axioms for 1 and 0 hold by definition of $\top$ and $\bot$.

Idempotence: By IpC4 and IpC5 we have $\vdash_I x \to (x \vee x)$ and $\vdash_I (x \wedge x) \to x$. IpC8 gives us

$$\vdash_I (x \to x) \to ((x \to x) \to ((x \vee x) \to x))$$

and therefore $\vdash_I (x \vee x) \to x$. By IpC3 we have $\vdash_I x \to (x \to (x \wedge x))$ and thus $\vdash_I x \to (x \wedge x)$.

Absorption: We have $\vdash_I x \wedge (x \vee y) \to x$ (IpC4), $\vdash_I x \to x \vee y$ (IpC5), $\vdash_I x \to x$ and thus $\vdash_I x \to (x \wedge (x \vee y))$. The converse is IpC4.

RusTeX

Theorem 2.3. *The freely generated Heyting algebra $\mathcal{H}_A$ is the* Tarski-Lindenbaum algebra *of IpC over A. This means for any formulae $a, b \in \mathcal{F}(A)$ we have $[a] \sim [b]$ iff $\vdash_I a \leftrightarrow b$.*

Proof. We have to show the following:

1. For every axiom $[t_1] \doteq [t_2]$ of T_H we have $\vdash_I t_1 \leftrightarrow t_2$.
2. *Modus Ponens:* If $[\varphi] = 1$ and $[\varphi] \hookrightarrow [\psi] = 1$, then $[\psi] = 1$.
3. For every axiom $\varphi(\psi_1, \psi_2, \psi_3)$ of IpC we have $\mathcal{H}_A | = \forall x, y, z[\varphi(x, y, z)] \doteq 1$.

Proof of 1. We have already proven, that in IpC conjunction is commutative and associative, the same holds for disjunction (follows immediately by IpC5, IpC6 and Modus Ponens). The axioms for 1 and 0 hold by definition of $\top$ and $\bot$.

Idempotence: By IpC4 and IpC5 we have $\vdash_I x \to (x \vee x)$ and $\vdash_I (x \wedge x) \to x$. IpC8 gives us

$$\vdash_I (x \to x) \to ((x \to x) \to ((x \vee x) \to x))$$

and therefore $\vdash_I (x \vee x) \to x$. By IpC3 we have $\vdash_I x \to (x \to (x \wedge x))$ and thus $\vdash_I x \to (x \wedge x)$.

Absorption: We have $\vdash_I x \wedge (x \vee y) \to x$ (IpC4), $\vdash_I x \to x \vee y$ (IpC5), $\vdash_I x \to x$ and thus $\vdash_I x \to (x \wedge (x \vee y))$. The converse is IpC4.

Consider the Primitives: \hbox

PDF

\hbox{ Box Content }
\hbox{ Box Content }
\hbox{ Box Content }
\hbox{ \hbox{ Box Content } }
\hbox{ \hbox{ Box Content } }
\hbox{ \hbox{ Box Content } } }
BoxContent
\hbox{ \hbox{..Box..Content..}.... \hbox{..Box..Content..}..... }

\hbox{ Box Content }
\hbox{ Box Content }
\hbox{ \hbox{Box Content} }
\hbox{\hbox{Box Content}}

RusTEX

\hbox{ Box Content }
\hbox{ Box Content }
\hbox{ Box Content }
\hbox{ \hbox{ Box Content } }
\hbox{ \hbox{ Box Content } }
\hbox{ \hbox{ Box Content } } }
BoxContent
\hbox{ \hbox{..Box..Content..}.. \hbox{..Box..Content..}.. }

\hbox{ Box Content }
\hbox{ Box Content }
\hbox{ \hbox{Box Content} }
\hbox{\hbox{Box Content}}

TEX4ht

\hbox{ Box Content } \hbox{ Box Content } \hbox{ Box Content } \hbox{ \hbox{ Box Content } } \hbox{ \hbox{ Box Content } } \hbox{ \hbox{ Box Content } } BoxContent \hbox{ \hbox{ Box Content } ..... \hbox{ Box Content } ..... } \hbox{ Box Content } \hbox{ Box Content } \hbox{ \hbox{Box Content} } \hbox{ \hbox{Box Content} }

LATEXML

\hbox{ Box Content } \hbox{ Box Content }\hbox{ Box Content }

\hbox{ \hbox{ Box Content } } \hbox{ \hbox{ Box Content } } }

\hbox{ \hbox{ Box Content } } }

BoxContent

\hbox{ \hbox{ Box Content }..... \hbox{ Box Content }..... } \hbox{ Box Content }

\hbox{ Box Content } \hbox{ \hbox{Box Content} } \hbox{\hbox{Box Content}}

Cost

```
<div class="rustex-hbox-container" style="min-width:var(--document-width);width:var(--document-width);">
  <div class="rustex-hbox" style="line-height:1;">
    <span class="rustex-contents" style="color:#0000FF;">
      <span class="rustex-text rustex-reset-font rustex-script-font" style="font-size:100%;">\</span>
      <span class="rustex-text">hbox</span>
      <span class="rustex-text rustex-reset-font rustex-script-font" style="font-size:100%;">{</span>
    </span>
    <div class="rustex-space-in-hbox"> </div>
    <span class="rustex-text">Box</span>
    <div class="rustex-space-in-hbox"> </div>
    <span class="rustex-text">Content</span>
    <div class="rustex-space-in-hbox"> </div>
    <span class="rustex-text rustex-reset-font rustex-script-font" style="font-size:100%;color:#0000FF;">}</span>
  </div>
</div>
```

(Just the first \hbox{ Box Content })

Consider the Primitives: \vtop

PDF

\vtop{ Box Content } \vtop{ Box Content }

\hbox{ \vtop{ Box Content } }

RusTeX

\vtop{ Box Content } \vtop{ Box Content }

\hbox{ \vtop{ Box Content } }

TeX4ht

\hbox{ \vtop{ Box Content } \vtop{ Box Content } \vtop{ Box Content } }

LaTeXML

\hbox{ \vtop{ Box Content } \vtop{ Box Content } \vtop{ Box Content } }

Consider the Primitives: Skips

PDF

```
\hbox{Foo} \hss
          \hss \hbox{Foo} \hss
                    \hss \hbox{Foo} \hss
\hbox{Foo} \hfil
          \hss \hbox{Foo} \hfil
                    \hfill \hbox{Foo} \hss
                   \hfill \hbox{Foo} \hfil
                         \hfill \hbox{Foo}
```

RusTeX

```
\hbox{Foo} \hss
          \hss \hbox{Foo} \hss
                    \hss \hbox{Foo} \hss
\hbox{Foo} \hfil
          \hss \hbox{Foo} \hfil
                    \hfill \hbox{Foo} \hss
                   \hfill \hbox{Foo} \hfil
                         \hfill \hbox{Foo}
```

TeX4ht

```
\hbox{Foo}\hss \hss \hbox{Foo}\hss \hss \hbox{Foo}\hss \hbox{Foo}\hfil \hss
\hbox{Foo}\hfil \hfill \hbox{Foo}\hss \hfill \hbox{Foo}\hfil \hfill
\hbox{Foo}
```

LaTeXML

```
\hbox{Foo} \hss \hss \hbox{Foo} \hss \hss \hbox{Foo} \hss \hbox{Foo} \hfil \hss
\hbox{Foo} \hfil \hfill \hbox{Foo} \hss \hfill \hbox{Foo} \hfil \hfill \hbox{Foo}
```

Why so little coverage?

Speculation:

- Focus on the macros/packages where "expected HTML output" is well-defined in the first place, provide dedicated HTML-"translations" for common packages.
- Keep the HTML plain and simple, so that users can easily provide their own CSS.
- Avoid fixed dimensions (e.g. `width=50px`) (responsive HTML)

...which is perfectly valid!

...but as a result: Easily trip up for unexpected, *intrusive* macros/packages or *redefinitions*, (modifying existing macros or exploiting "side effects")
"neglect" *in-depth* support for "low-level" TEX primitives
⇒ Contrary to common practice, i.e. copy-pasted macro definitions and preambles.
The average user is not a TEX expert!

Idea: If we get the primitives "right", we

- don't need to provide special support for "high-level" LaTeX macros, (Neverending work!)
- can guarantee that any document that compiles with `pdflatex` will also compile with RusTeX,
- (ideally) can make sure that *any* valid LaTeX produces (somewhat) decent output
- but **at the cost of document semantics**, (No `<li>`, `<ol>`, `<ul>`, `<h1>`, ...)
  "ugly" HTML sources (deeply nested divs)
  and less opportunity for custom CSS.

(For details on the CSS and general architecture, see the accompanying preprint/paper)

Lists

PDF

3.5 Objectives

The OAF and OpenDreamKit projects have a common aim, differing only in their domains of applicability – or aspects of the tetrapod (see Chapter 1): The integration of (libraries of) formal systems (*inference*) in the former, and mathematical software and databases (*computation/tabulation*) in the latter. The goal of this thesis is to outline an approach to solve this integration problem and to demonstrate its feasibility. This entails the following objectives:

O1 **Represent the logical foundations, libraries and ontologies** of theorem prover systems, computer algebra systems and other sources of mathematical knowledge desired for the OpenDreamKit project **in a unifying framework**.

O2 As lamented in Section 2.3, the logical frameworks currently available are not sufficient to formalize the foundations and distinct primitive features of real world theorem prover systems conveniently. Consequently, we need to **develop a modular, sufficiently powerful logical framework** in order to both achieve **O1**, as well as formalize the mathematical knowledge needed in the **Math-in-the-Middle library** as conveniently as possible.

O3 Having the libraries of various systems accessible from within a unifying framework, the goal is to **develop knowledge management service** on a generic level in order to systematically **integrate formal libraries**, facilitating knowledge and structure transfer and **translate expressions** between foundations and systems.

HTML

3.5 Objectives

The OAF and OpenDreamKit projects have a common aim, differing only in their domains of applicability — or aspects of the tetrapod (see Chapter 1): The integration of (libraries of) formal systems (*inference*) in the former, and mathematical software and databases (*computation/tabulation*) in the latter. The goal of this thesis is to outline an approach to solve this integration problem and to demonstrate its feasibility. This entails the following objectives:

O1 **Represent the logical foundations, libraries and ontologies** of theorem prover systems, computer algebra systems and other sources of mathematical knowledge desired for the OpenDreamKit project **in a unifying framework**.

O2 As lamented in Section 2.3, the logical frameworks currently available are not sufficient to formalize the foundations and distinct primitive features of real world theorem prover systems conveniently. Consequently, we need to **develop a modular, sufficiently powerful logical framework** in order to both achieve **O1**, as well as formalize the mathematical knowledge needed in the **Math-in-the-Middle library** as conveniently as possible.

O3 Having the libraries of various systems accessible from within a unifying framework, the goal is to **develop knowledge management service** on a generic level in order to systematically **integrate formal libraries**, facilitating knowledge and structure transfer and **translate expressions** between foundations and systems.

Tables

PDF

4.1.1 Mmt Syntax

Intuitively, OMDoc/Mmt is a declarative language for theories and views over an arbitrary object language. For the purposes of this thesis, we will work with the (only slightly simplified) grammar given in Figure 4.2.

Thy ::=	$T[:T] = \{(\mathsf{Inc})^* \ (\mathsf{Const})^*\}$	Theory	Modules
View ::=	$T : T \to T = \{(\mathsf{Ass})^*\}$	View	
Const ::=	$C[:t][= t]$	Constant Declarations	Declarations
Inc ::=	include T	Includes	
Ass ::=	$C \mapsto t$	Assignments	
Γ ::=	$(x[:t][:= t])^*$	Variable Contexts	Objects
t ::=	$x \mid C \mid t\,[\Gamma]\,(t)$	Terms	

T represents a module name, C a constant name and x a variable name

Figure 4.2: The MMT Grammar

HTML

4.1.1 Mmt Syntax

Intuitively, OMDoc/Mmt is a declarative language for theories and views over an arbitrary object language. For the purposes of this thesis, we will work with the (only slightly simplified) grammar given in Figure 4.2.

Thy ::=	$T[:T] = \{(\mathsf{Inc})^* \ (\mathsf{Const})^*\}$	Theory	Modules
View ::=	$T : T \to T = \{(\mathsf{Ass})^*\}$	View	
Const ::=	$C[:t][:= t]$	Constant Declarations	Declarations
Inc ::=	include T	Includes	
Ass ::=	$C \mapsto t$	Assignments	
Γ ::=	$(x[:t][:= t])^*$	Variable Contexts	Objects
t ::=	$x \mid C \mid t\,[\Gamma]\,(t)$	Terms	

T represents a module name, C a constant name and x a variable name

Figure 4.2: The MMT Grammar

Listings

PDF

Listing 5.5: Equality Rule[5]

```
1  /** equality-checking: the rule
2    * |- pi1(t1) = pi1(t2) : A , |- pi2(t1) = pi2(t2) : B(pi1(t)) ---> t1 = t2 : Sigma x:A. B **/
3  object TupleEquality extends TypeBasedEqualityRule(Nil,Sigma.path) {
4    override def alternativeHeads:List[GlobalName] = List(Product.path)
5    override def applicableToTerm(tm: Term): Boolean = true
6
7    def apply(solver: Solver)(tm1: Term, tm2: Term, tp: Term)
8          (implicit stack: Stack, history: History): Option[Boolean]
9      = tp match {
10       case Sigma(x, tpA, tpB) =>
11         solver.check(Equality(stack, Proj1(tm1), Proj1(tm2), Some(tpA)))
12         Some(solver.check(Equality(stack, Proj2(tm1), Proj2(tm2),
13           Some(tpB ^? (x / Proj1(tm1))))))
14       case _ => None // should be impossible
15     }
16 }
```

HTML

Listing 5.5: Equality Rule[5]

```
1  /** equality-checking: the rule
2    * |- pi1(t1) = pi1(t2) : A , |- pi2(t1) = pi2(t2) : B(pi1(t)) ---> t1 = t2 : Sigma x:A. B **/
3  object TupleEquality extends TypeBasedEqualityRule(Nil,Sigma.path) {
4    override def alternativeHeads:List[GlobalName] = List(Product.path)
5    override def applicableToTerm(tm: Term): Boolean = true
6
7    def apply(solver: Solver)(tm1: Term, tm2: Term, tp: Term)
8          (implicit stack: Stack, history: History): Option[Boolean]
9      = tp match {
10       case Sigma(x, tpA, tpB) =>
11         solver.check(Equality(stack, Proj1(tm1), Proj1(tm2), Some(tpA)))
12         Some(solver.check(Equality(stack, Proj2(tm1), Proj2(tm2),
13           Some(tpB ^? (x / Proj1(tm1))))))
14       case _ => None // should be impossible
15     }
16 }
```

PDF

Chapter 18

Bibliography

Online Source References

[LFX] *MathHub MMT/LFX Git Repository.* URL: http://gl.mathhub.info/MMT/LFX (visited on 05/15/2015).

[Mita] *MitM/Foundation.* URL: https://gl.mathhub.info/MitM/Foundation (visited on 09/01/2017).

[Mitb] *MitM/Interfaces.* URL: https://mathhub.info/MitM/interfaces (visited on 06/21/2017).

[Mitc] *MitM/smglom.* URL: https://gl.mathhub.info/MitM/smglom (visited on 02/01/2018).

[MMT] *UniFormal/MMT — The MMT Language and System.* URL: https://github.com/UniFormal/MMT (visited on 10/24/2017).

[OMU] *OMDoc/MMT Urtheories.* URL: https://gl.mathhub.info/MMT/urtheories (visited on 06/02/2017).

[PRA] *Public Repository for Alignments.* https://gl.mathhub.info/alignments/Public. (Visited on 05/21/2017).

HTML

Chapter 18

Bibliography

Online Source References

[LFX] *MathHub MMT/LFX Git Repository.* URL: http://gl.mathhub.info/MMT/LFX (Visited on 05/15/2015).

[Mita] *MitM/Foundation.* URL: https://gl.mathhub.info/MitM/Foundation (Visited on 09/01/2017).

[Mitb] *MitM/Interfaces.* URL: https://mathhub.info/MitM/interfaces (Visited on 06/21/2017).

[Mitc] *MitM/smglom.* URL: https://gl.mathhub.info/MitM/smglom (Visited on 02/01/2018).

[MMT] *UniFormal/MMT — The MMT Language and System.* URL: https://github.com/UniFormal/MMT (Visited on 10/24/2017).

[OMU] *OMDoc/MMT Urtheories.* URL: https://gl.mathhub.info/MMT/urtheories (Visited on 06/02/2017).

[PRA] *Public Repository for Alignments.* https://gl.mathhub.info/alignments/Public. (Visited on 05/21/2017).

Algorithms

PDF

The algorithm defined by the function euclid below which computes the greatest commmon divisor of two integers is called the **Euclidean algorithm**.

function euclid(n, m)
  Input: Two integers n, m
  Output: The greatest common divisor of n and m
  begin
    if $n = m$ **then**
      return n
    end
    else if $m<n$ **then**
      return euclid($n - m, m$)
    end
    else if $n<m$ **then**
      return euclid($n, m - n$)
    end
    else
      return euclid($n - m, m$)
    end
  end

HTML

Definition 0.1. The algorithm defined by the function euclid below which computes the greatest commmon divisor of two integers is called the Euclidean algorithm.

function euclid(n, m)
  Input: Two integers n, m
  Output: The greatest common divisor of n and m
  begin
    if $n = m$ **then**
      return n
    end
    else if $m < n$ **then**
      return euclid($n - m$, m)
    end
    else if $n < m$ **then**
      return euclid(n, $m - n$)
    end
    else
      return euclid($n - m$, m)
    end
  end

PDF

Again, we can easily show the corresponding subtyping and η-rules:

Theorem 5.1:

The following subtyping rule is derivable:

$$\frac{\Gamma \vdash A' <: A \quad \Gamma, x : A' \vdash B' <: B}{\Gamma \vdash \sum_{x:A'} B' <: \sum_{x:A} B}$$

Furthermore, the η-rule (i.e. for any $p : \sum_{x:A} B$ we have $\langle \pi_\ell(p), \pi_r(p) \rangle = p$) holds.

Proof. Analogous to Theorem 4.1 □

HTML

Again, we can easily show the corresponding subtyping and η-rules:

Theorem 5.1:

The following subtyping rule is derivable:

$$\frac{\Gamma \vdash A' <: A \quad \Gamma, x : A' \vdash B' <: B}{\Gamma \vdash \sum_{x : A'} B' <: \sum_{x : A} B}$$

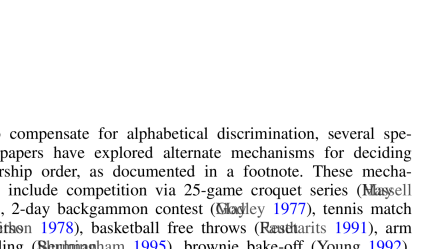
Furthermore, the η-rule (i.e. for any $p : \sum_{x : A} B$ we have $\langle \pi_\ell(p), \pi_r(p) \rangle = p$) holds.

Proof. Analogous to Theorem 4.1 □

General Weirdness

Every Author as First Author – Demaine, E. and Demaine, M.

PDF



To compensate for alphabetical discrimination, several specific papers have explored alternate mechanisms for deciding authorship order, as documented in a footnote. These mechanisms include competition via 25-game croquet series (May 1974), 2-day backgammon contest (Madley 1977), tennis match (Griffiths 1978), basketball free throws (Rastharits 1991), arm wrestling (Birmingham 1995), brownie bake-off (Young 1992), a game of chicken (Reichenstein 1983), or rock paper scissors (Wiebking 2004); by coin toss (Miller 1992), dice roll (Schofield 2011), the outcome of famous cricket games (Ochoa 2010), currency exchange rate fluctuation (Mitchell-Olds 2003), or dog treat consumption order (Wilkinson 2019); or by authors' height (Woodward 2005), fertility (Oliveock 1992), proximity to tenure (Gillespie 1998), reverse alphabetical order (Wiessenger 2006), or degree of belief in the paper's thesis (Chalmers 1998). Others have proposed games such as Russian roulette ("publish and perish") (Purvis 2016). See the excellent surveys (Duffy 2016; Deville 2014; Obscura 2014) and their comments.

HTML

To compensate for alphabetical discrimination, several specific papers have explored alternate mechanisms for deciding authorship order, as documented in a footnote. These mechanisms include competition via 25-game croquet series (May 1974), 2-day backgammon contest (Madley 1977), tennis match (Griffiths 1978), basketball free throws (Rastharits 1991), arm wrestling (Birmingham 1995), brownie bake-off (Young 1992), a game of chicken (Reichenstein 1983), or rock paper scissors (Wiebking 2004); by coin toss (Miller 1992), dice roll (Schofield 2011), the outcome of famous cricket games (Ochoa 2010), currency exchange rate fluctuation (Mitchell-Olds 2003), or dog treat consumption order (Wilkinson 2019); or by authors' height (Woodward 2005), fertility (Oliveock 1992), proximity to tenure (Gillespie 1998), reverse alphabetical order (Wiessenger 2006), or degree of belief in the paper's thesis (Chalmers 1998). Others have proposed games such as Russian roulette ("publish and perish") (Purvis 2016). See the excellent surveys (Duffy 2016; Deville 2014; Obscura 2014) and their comments.

The average sTEX document

The introduction of quantifiers to first-order logic brings a new phenomenon: variables that are under the scope of a quantifiers will behave very differently from the ones that are not. Therefore we build up a vocabulary that distinguishes the two.

Free and Bound Variables

▷ **Definition 15.2.9.** We call an occurrence of a variable X bound in a formula $\mathbf{A}$, iff it occurs in a sub-formula $\forall X.\mathbf{B}$ of $\mathbf{A}$. We call a variable occurrence free otherwise.

For a formula $\mathbf{A}$, we will use $\mathsf{BVar}(\mathbf{A})$ (and $\mathsf{free}(\mathbf{A})$) for the set of bound (free) variables of $\mathbf{A}$, i.e. variables that have a free/bound occurrence in $\mathbf{A}$.

▷ **Definition 15.2.10.** We define the set $\mathsf{free}(\mathbf{A})$ of frees variable of a formula $\mathbf{A}$:

$$\begin{aligned}
&\mathsf{free}(X) := \{X\}\\
&\mathsf{free}(f(\mathbf{A}_1,\ldots,\mathbf{A}_n)) := \bigcup_{1\leq i\leq n}\mathsf{free}(\mathbf{A}_i)\\
&\mathsf{free}(p(\mathbf{A}_1,\ldots,\mathbf{A}_n)) := \bigcup_{1\leq i\leq n}\mathsf{free}(\mathbf{A}_i)\\
&\mathsf{free}(\neg\mathbf{A}) := \mathsf{free}(\mathbf{A})\\
&\mathsf{free}(\mathbf{A}\wedge\mathbf{B}) := \mathsf{free}(\mathbf{A})\cup\mathsf{free}(\mathbf{B})\\
&\mathsf{free}(\forall X.\mathbf{A}) := \mathsf{free}(\mathbf{A})\backslash\{X\}
\end{aligned}$$

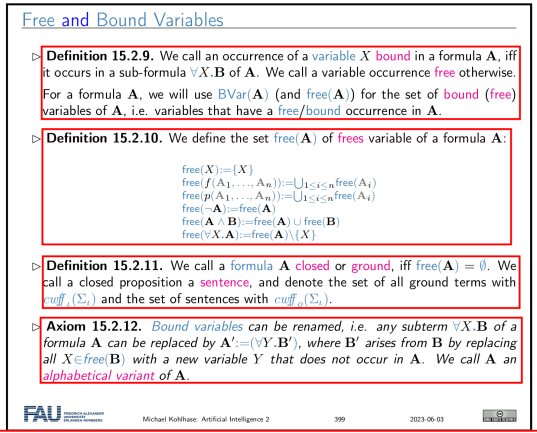
▷ **Definition 15.2.11.** We call a formula $\mathbf{A}$ closed or ground, iff $\mathsf{free}(\mathbf{A}) = \emptyset$. We call a closed proposition a sentence, and denote the set of all ground terms with $\mathit{cuff}_\iota(\Sigma_\iota)$ and the set of sentences with $\mathit{cuff}_o(\Sigma_\iota)$.

▷ **Axiom 15.2.12.** *Bound variables can be renamed, i.e. any subterm $\forall X.\mathbf{B}$ of a formula $\mathbf{A}$ can be replaced by $\mathbf{A}' := (\forall Y.\mathbf{B}')$, where $\mathbf{B}'$ arises from $\mathbf{B}$ by replacing all $X\in\mathsf{free}(\mathbf{B})$ with a new variable Y that does not occur in $\mathbf{A}$. We call $\mathbf{A}$ an alphabetical variant of $\mathbf{A}$.*

We will be mainly interested in (sets of) sentences – i.e. closed propositions – as the representations of meaningful statements about individuals. Indeed, we will see below that free variables do not gives us expressivity, since they behave like constants and could be replaced by them in all situations, except the recursive definition of quantified formulae. Indeed in all situations where variables occur freely, they have the character of meta-variables, i.e. syntactic placeholders that can be instantiated with terms when needed in an inference calculus.

The average sTEX document

The introduction of quantifiers to first-order logic brings a new phenomenon: variables that are under the scope of a quantifiers will behave very differently from the ones that are not. Therefore we build up a vocabulary that distinguishes the two.



Free and Bound Variables

▷ **Definition 15.2.9.** We call an occurrence of a variable X bound in a formula $\mathbf{A}$, iff it occurs in a sub-formula $\forall X.\mathbf{B}$ of $\mathbf{A}$. We call a variable occurrence free otherwise.

For a formula $\mathbf{A}$, we will use $\mathsf{BVar}(\mathbf{A})$ (and $\mathsf{free}(\mathbf{A})$) for the set of bound (free) variables of $\mathbf{A}$, i.e. variables that have a free/bound occurrence in $\mathbf{A}$.

▷ **Definition 15.2.10.** We define the set $\mathsf{free}(\mathbf{A})$ of frees variable of a formula $\mathbf{A}$:

$$\begin{aligned}
&\mathsf{free}(X) := \{X\}\\
&\mathsf{free}(f(\mathbf{A}_1,\ldots,\mathbf{A}_n)) := \bigcup_{1\leq i\leq n}\mathsf{free}(\mathbf{A}_i)\\
&\mathsf{free}(p(\mathbf{A}_1,\ldots,\mathbf{A}_n)) := \bigcup_{1\leq i\leq n}\mathsf{free}(\mathbf{A}_i)\\
&\mathsf{free}(\neg\mathbf{A}) := \mathsf{free}(\mathbf{A})\\
&\mathsf{free}(\mathbf{A}\wedge\mathbf{B}) := \mathsf{free}(\mathbf{A})\cup\mathsf{free}(\mathbf{B})\\
&\mathsf{free}(\forall X.\mathbf{A}) := \mathsf{free}(\mathbf{A})\setminus\{X\}
\end{aligned}$$

▷ **Definition 15.2.11.** We call a formula $\mathbf{A}$ closed or ground, iff $\mathsf{free}(\mathbf{A}) = \emptyset$. We call a closed proposition a sentence, and denote the set of all ground terms with $\mathit{cwff}_\iota(\Sigma_\iota)$ and the set of sentences with $\mathit{cwff}_o(\Sigma_\iota)$.

▷ **Axiom 15.2.12.** *Bound variables can be renamed, i.e. any subterm $\forall X.\mathbf{B}$ of a formula $\mathbf{A}$ can be replaced by $\mathbf{A}' := (\forall Y.\mathbf{B}')$, where $\mathbf{B}'$ arises from $\mathbf{B}$ by replacing all $X\in\mathsf{free}(\mathbf{B})$ with a new variable Y that does not occur in $\mathbf{A}$. We call $\mathbf{A}$ an alphabetical variant of $\mathbf{A}$.*

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG Michael Kohlhase: Artificial Intelligence 2 399 2023-06-03 sTEX

We will be mainly interested in (sets of) sentences – i.e. closed propositions – as the representations of meaningful statements about individuals. Indeed, we will see below that free variables do not gives us expressivity, since they behave like constants and could be replaced by them in all situations, except the recursive definition of quantified formulae. Indeed in all situations where variables occur freely, they have the character of meta-variables, i.e. syntactic placeholders that can be instantiated with terms when needed in an inference calculus.

(individual files)

...and as HTML

The introduction of quantifiers to first-order logic brings a new phenomenon: variables that are under the scope of a quantifiers will behave very differently from the ones that are not. Therefore we build up a vocabulary that distinguishes the two.

Free and Bound Variables

▷ **Definition 1.2.9.** We call an occurrence of a variable X bound in a formula $\mathbf{A}$, iff it occurs in a sub-formula $\forall X.\mathbf{B}$ of $\mathbf{A}$. We call a variable occurrence free otherwise.
For a formula $\mathbf{A}$, we will use $\mathsf{BVar}(\mathbf{A})$ (and $\mathsf{free}(\mathbf{A})$) for the set of bound (free) variables of $\mathbf{A}$, i.e. variables that have a free/bound occurrence in $\mathbf{A}$.

▷ **Definition 1.2.10.** We define the set $\mathsf{free}(\mathbf{A})$ of frees variable of a formula $\mathbf{A}$:

$$\begin{aligned}
&\mathsf{free}(X) := \{X\}\\
&\mathsf{free}(f(\mathbf{A}_1, \ldots, \mathbf{A}_n)) := \bigcup_{1\leq i\leq n} \mathsf{free}(\mathbf{A}_i)\\
&\mathsf{free}(p(\mathbf{A}_1, \ldots, \mathbf{A}_n)) := \bigcup_{1\leq i\leq n} \mathsf{free}(\mathbf{A}_i)\\
&\mathsf{free}(\neg\mathbf{A}) := \mathsf{free}(\mathbf{A})\\
&\mathsf{free}(\mathbf{A}\wedge\mathbf{B}) := \mathsf{free}(\mathbf{A})\cup\mathsf{free}(\mathbf{B})\\
&\mathsf{free}(\forall X.\mathbf{A}) := \mathsf{free}(\mathbf{A})\backslash\{X\}
\end{aligned}$$

▷ **Definition 1.2.11.** We call a formula $\mathbf{A}$ closed or ground, iff $\mathsf{free}(\mathbf{A}) = \emptyset$. We call a closed proposition a sentence, and denote the set of all ground terms with $cwff_\iota(\Sigma_\iota)$ and the set of sentences with $cwff_o(\Sigma_\iota)$.

▷ **Axiom 1.2.12.** Bound variables can be renamed, i.e. any subterm $\forall X.\mathbf{B}$ of a formula $\mathbf{A}$ can be replaced by $\mathbf{A}' := (\forall Y.\mathbf{B}')$, where $\mathbf{B}'$ arises from $\mathbf{B}$ by replacing all $X \in \mathsf{free}(\mathbf{B})$ with a new variable Y that does not occur in $\mathbf{A}$. We call $\mathbf{A}'$ an alphabetical variant of $\mathbf{A}$.

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG : ©2023-06-02 CC BY NC SA

We will be mainly interested in (sets of) sentences — i.e. closed propositions — as the representations of meaningful statements about individuals. Indeed, we will see below that free variables do not gives us expressivity, since they behave like constants and could be replaced by them in all situations, except the recursive definition of quantified formulae. Indeed in all situations where variables occur freely, they have the character of meta-variables, i.e. syntactic placeholders that can be instantiated with terms when needed in an inference calculus.

(files dynamically included)

...and as HTML

The introduction of quantifiers to first-order logic brings a new phenomenon: variables that are under the scope of a quantifiers will behave very differently from the ones that are not. Therefore we build up a vocabulary that distinguishes the two.

Free and Bound Variables

▷ **Definition 1.2.9.** We call an occurrence of a variable X bound in a formula $\mathbf{A}$, iff it occurs in a sub-formula $\forall X.\mathbf{B}$ of $\mathbf{A}$. We call a variable occurrence free otherwise.
For a formula $\mathbf{A}$, we will use $\mathsf{BVar}(\mathbf{A})$ (and $\mathsf{free}(\mathbf{A})$) for the set of bound (free) variables of $\mathbf{A}$, i.e. variables that have a free/bound occurrence in $\mathbf{A}$.

▷ **Definition 1.2.10.** We define the set $\mathsf{free}(\mathbf{A})$ of frees variable of a formula $\mathbf{A}$:

$$\begin{aligned}
\mathsf{free}(X) &:= \{X\}\\
\mathsf{free}(f(\mathbf{A}_1, \ldots, \mathbf{A}_n)) &:= \bigcup_{1\leq i\leq n}\mathsf{free}(\mathbf{A}_i)\\
\mathsf{free}(p(\mathbf{A}_1, \ldots, \mathbf{A}_n)) &:= \bigcup_{1\leq i\leq n}\mathsf{free}(\mathbf{A}_i)\\
\mathsf{free}(\neg\mathbf{A}) &:= \mathsf{free}(\mathbf{A})\\
\mathsf{free}(\mathbf{A}\wedge\mathbf{B}) &:= \mathsf{free}(\mathbf{A})\cup\mathsf{free}(\mathbf{B})\\
\mathsf{free}(\forall X.\mathbf{A}) &:= \mathsf{free}(\mathbf{A})\setminus\{X\}
\end{aligned}$$

▷ **Definition 1.2.11.** We call a formula $\mathbf{A}$ closed or ground, iff $\mathsf{free}(\mathbf{A}) = \emptyset$. We call a closed proposition a sentence. and denote the set of all ground terms with $\mathit{cwff}(\Sigma)$ and the set of sentence

▷ **Axiom 1.2** can be replace new variable Y

Definition 0.1.
Given a set Σ of constants and a set $\mathcal{V}$ of variables (usually assumed to be countably infinite), the set well-formed formulae over the signature Σ is written as $\mathit{wff}(\Sigma, \mathcal{V})$.
As the particular choice of $\mathcal{V}$ does not matter (any formed formulae, we often simply write $\mathit{wff}(\Sigma)$

▷ **Definition 0.1.** We say that a set A is countably infinite, iff there is a bijective function $f\colon A\to\mathbb{N}$. A set is called countable, iff it is finite or countably infinite.

FAU · 62023-06-02 · CC BY NC SA

We will be mainly interested in (sets of) sentences — i.e. closed propositions — as the representations of meaningful statements about individuals. Indeed, we will see below that free variables do not gives us expressivity, since they behave like constants and could be replaced by them in all situations, except the recursive definition of quantified formulae. Indeed in all situations where variables occur freely, they have the character of meta-variables, i.e. syntactic placeholders that can be instantiated with terms when needed in an inference calculus.

Conclusion

Obviously: Work to be done to produce better HTML, support for certain package macros unavoidable and desired,...

- RusTeX started as a hobby project (Initially in Scala, to preprocess sTeX for machine learning) (≈70% of runtime spent in garbage collection)
- I only learned both Rust and TeX while implementing it, with no real expectation of it being useful (...and CSS)
- Now used in production, HTML deployed to 1000s of students

  `https://courses.voll-ki.fau.de/` `https://url.mathhub.info/stex`
- ...and now I (basically) know what I **should** have done (e.g. support for XeTeX files (currently) not feasible)

Next hobby project: Entirely rewrite RusTeX as a fully modular/generic, performant Rust crate for TeX engines (Exchange/adapt/combine freely any of: mouth, gullet, stomach, state, *character types*, token types, box types, output format, font types, physical/virtual file system, shipout routines, primitive commands,...)