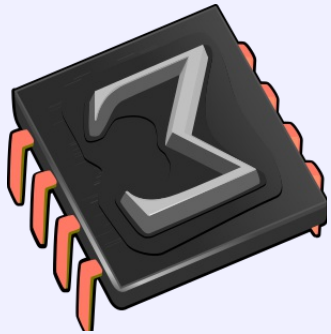
Guile and Guix Days in Strasbourg, June 21–22 2019

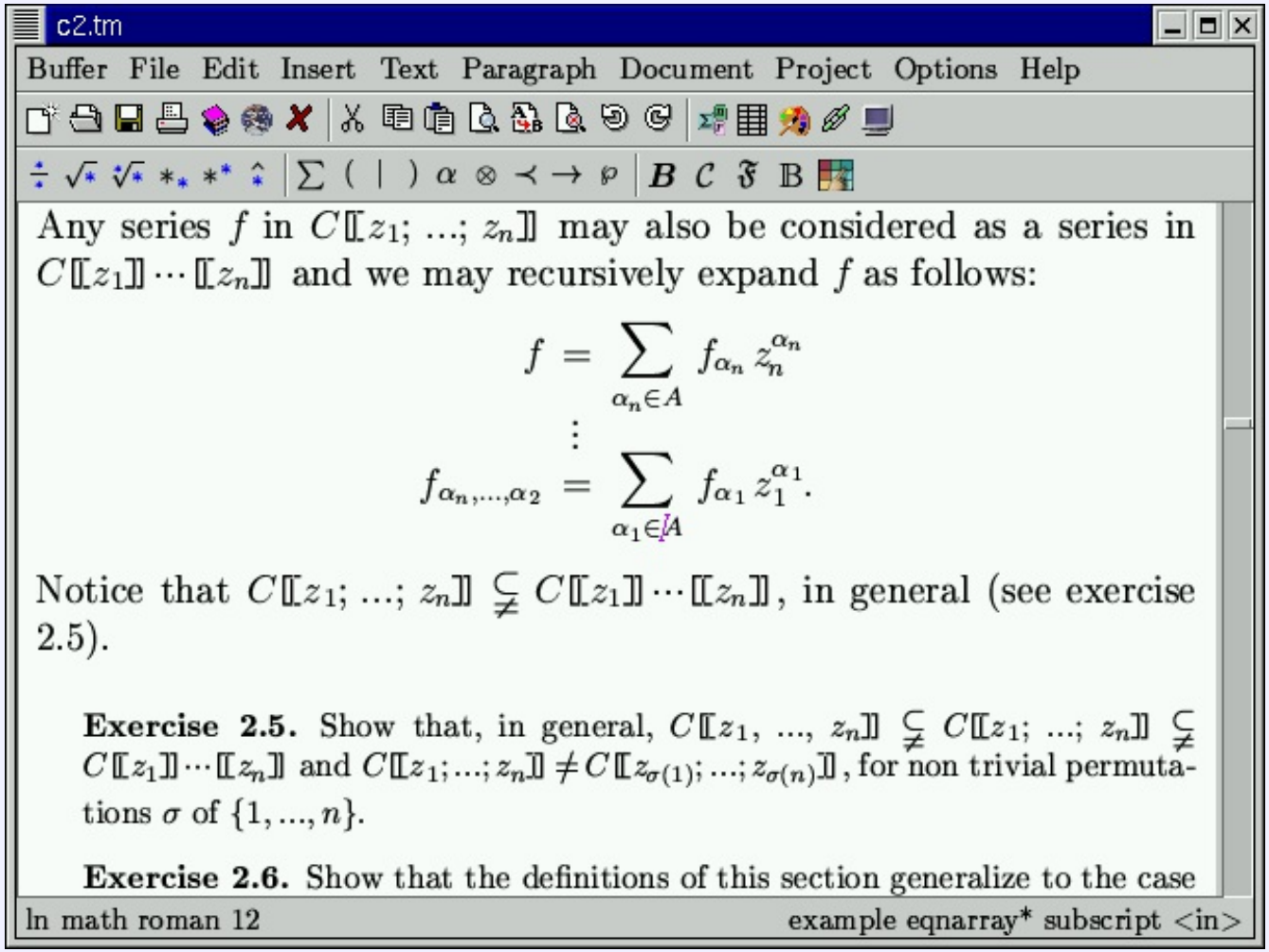
The Guile in TeXmacs



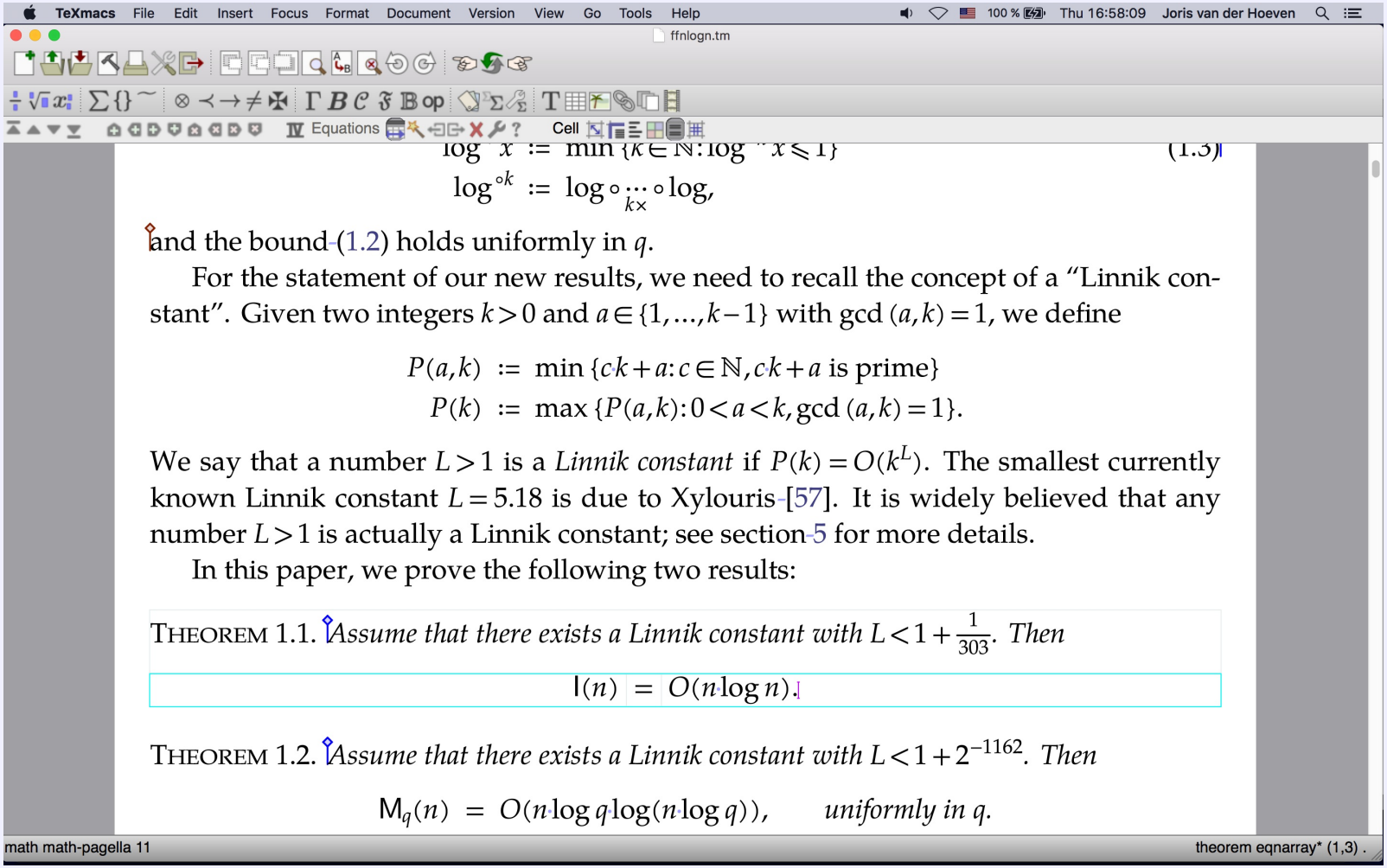
Massimiliano Gubinelli

- Visual **structured** editor: WYSWYG & WYSWYM
- Inspired by TEX and EMACS
- High-quality typesetting algorithms (including microtypography)
- Special features for mathematical typesetting and input
- Support for interactive sessions: Scheme, Python, R, Octave, Maxima, Axiom, Mathemagix (and other CAS).
- Multi-platform: Unix, MacOS, Windows (via QT)
- Own format (XML like). Native output to PDF and PS. Export to LATEX, HTML
- Internal image editor, interfaces to SVN and GIT, versioning tool, database tool, encryption of documents.
- Website and documentation written in TeXmacs

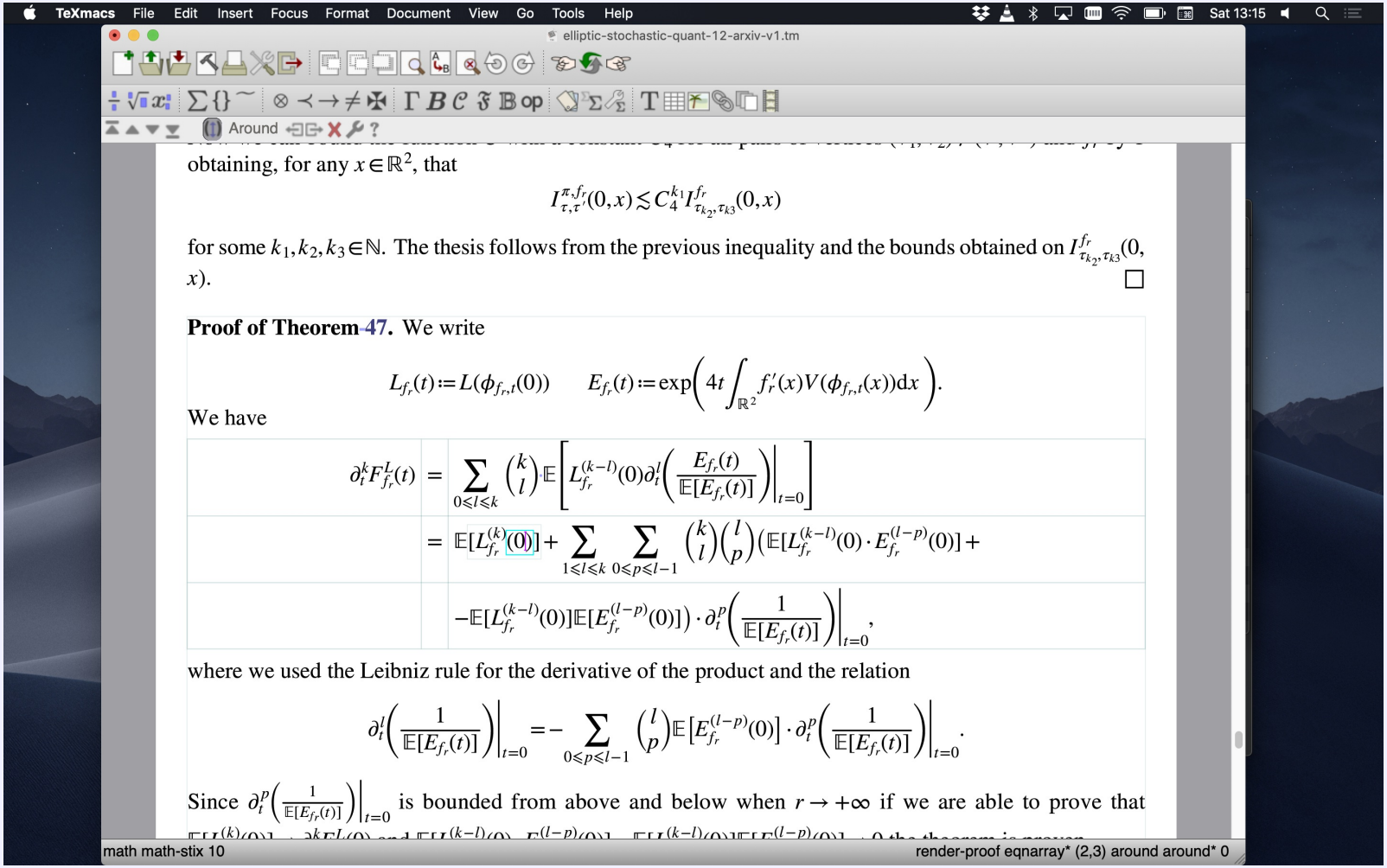
Gallery



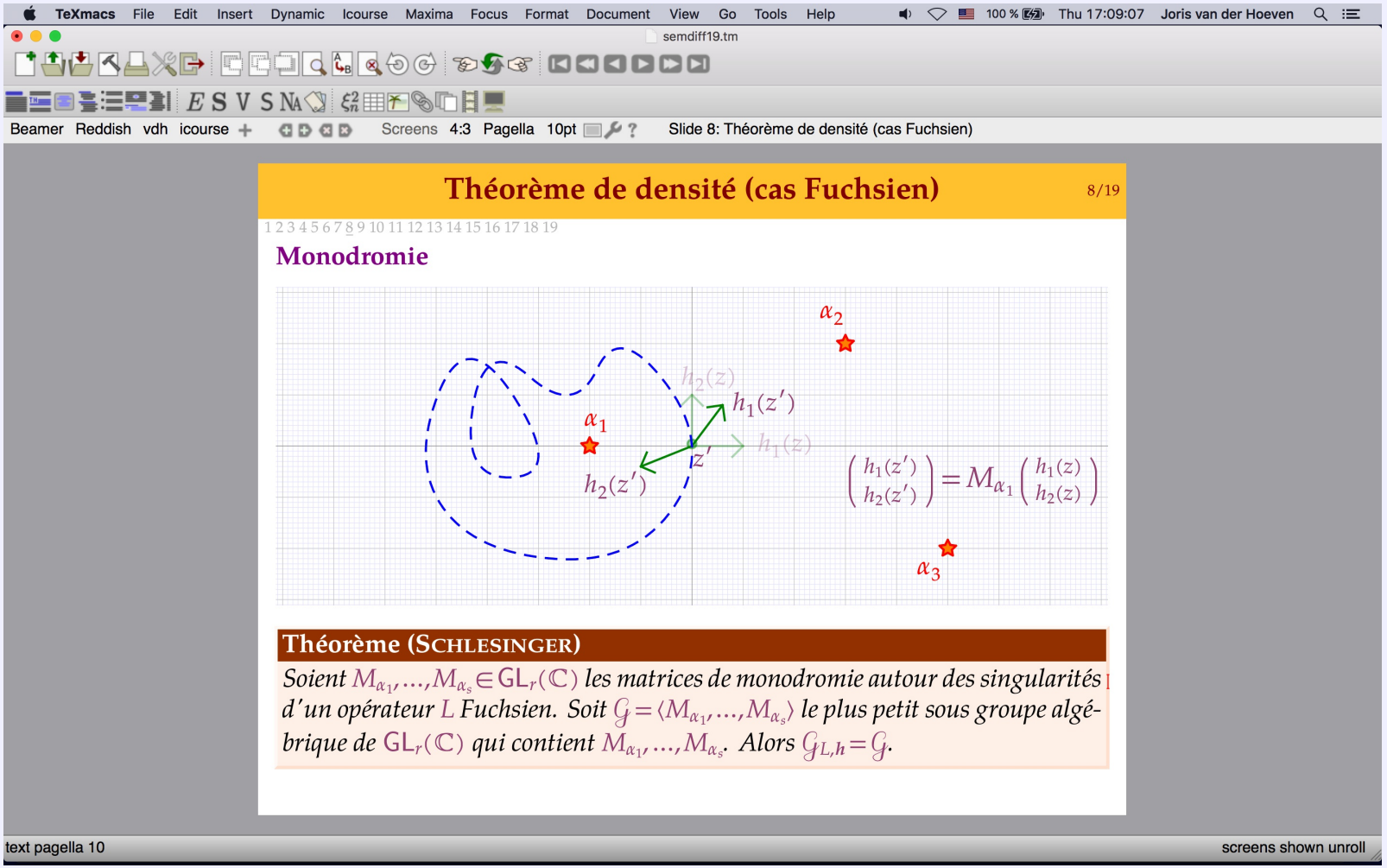
The legacy X11 backend



The QT backend, high quality typesetting



Structured editing, high quality math typesetting



Presentation mode

Autour du calcul numérique de groupes de Galois différentiels

Joris van der Hoeven
CNRS

Séminaire Différentiel — Palaiseau, 2 avril 2019

Le problème

$\mathbb{K} = \bar{\mathbb{Q}} \subseteq \mathbb{C}$ (ou sous corps de $\mathbb{C}$ avec $\mathbb{K} = \bar{\mathbb{K}}$ pour certains résultats)

Exemples

- $L = \partial - 1$,
  $h = (e^z)$

$$\sigma(e^z) = a e^z, \quad a \neq 0$$

Exemples (suite)

- $L = \partial^2 + (1 + z^{-1})\partial + z^{-1}$ («dérivée» de $h' + h = z^{-1}$)
  $h = (\frac{1}{z} + \frac{1}{z^2} + \frac{2}{z^3} + \frac{6}{z^4} + \cdots, e^{-z})$

$$\sigma(e^{-z}) = a e^{-z}, \quad a \in \mathbb{K}^*$$

Historique abrégée

1895, 1897. Schlesinger : théorème de densité (cas Fuchsien)

Factorisation de l'opérateur L

1894. Beke : facteurs d'ordre un

Solutions formelles autour d'une singularité

Point non singulier $z = \alpha$. On peut prendre $h = h^{(z)}$ unique telle que

$$\begin{pmatrix} h_1(z) & \cdots & h_r(z) \\ \vdots & & \vdots \\ h_1^{(r-1)}(z) & \cdots & h_r^{(r-1)}(z) \end{pmatrix} = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}.$$

Théorème de densité (cas Fuchsien)

Monodromie

Théorème de densité (cas général)

Matrices exponentielles

Induites par automorphismes σ exponentiels $\sigma(e^{P(1/z)}) = \lambda e^{P(1/z)}$

Théorème de densité (cas général, suite)

Accéléro-sommation d'Écalle

$$\begin{array}{ccccccc} \tilde{h} & & & & & & h \\ \downarrow & & & & & & \downarrow \\ \tilde{h}_1 & \xrightarrow[\Lambda_{0,1}]{} & \tilde{h}_2 & \longrightarrow \cdots \longrightarrow & \tilde{h}_{p-1} & \xrightarrow[\Lambda_{p-1,\infty}]{} & \tilde{h}_\infty \end{array}$$

Nombres holonomes

Théorème (Chudnovsky², van der Hoeven)

Pour $\mathbb{K} = \bar{\mathbb{Q}}$ et en prenant un point de base non singulier dans $\mathbb{K}$, on peut approximer en temps $O(n \log^3 n)$ les matrices de monodromie de L autour de chaque singularité avec une erreur d'au plus 2^{-n}.

Factorisation de L

1. Calculer des générateurs $M_1, \ldots, M_m \in \mathrm{GL}_r(\mathbb{C}^{\mathrm{calc}})$ de $\mathcal{G}_L$

Zoom sur l'étape 3

1. Calculer des générateurs $M_1, \ldots, M_m \in \mathrm{GL}_r(\mathbb{C}^{\mathrm{calc}})$ de $\mathcal{G}_L$
2. Fixer une précision p du calcul pour les «tests à zéro»
3. Déterminer un sous espace invariant V non trivial pour $M_1, \ldots, M_m$
4. Si un tel espace V n'existe pas, alors retourner L
5. À partir de V, reconstruire une factorisation $L = AB$ « candidat »
6. Si $L = AB$, alors retourner (A, B)
7. Doubler la précision et retourner à l'étape 3

Zoom sur l'étape 5

1. Calculer des générateurs $M_1, \ldots, M_m \in \mathrm{GL}_r(\mathbb{C}^{\mathrm{calc}})$ de $\mathcal{G}_L$
2. Fixer une précision p du calcul pour les «tests à zéro»
3. Déterminer un sous espace invariant V non trivial pour $M_1, \ldots, M_m$
4. Si un tel espace V n'existe pas, alors retourner L
5. À partir de V, reconstruire une factorisation $L = AB$ « candidat »
6. Si $L = AB$, alors retourner (A, B)
7. Doubler la précision et retourner à l'étape 3

Groupe Galois différentiel

Idée : calculer $\mathcal{G}$ comme variété sous la forme

$\mathcal{G} = \mathcal{F} e^{\mathcal{L}}$ $\quad (\forall N \in \mathcal{F},\ N e^{\mathcal{L}} = e^{\mathcal{L}} N)$
$\mathcal{F}$: ensemble fini contenant 1
$\mathcal{L}$: algèbre de Lie donnée par une base

Groupe Galois différentiel : l'algorithme

Étape 1. [Initialisation]
Calculer $(M_i) = \mathcal{F}_i e^{\mathcal{L}_i}$ pour tout $i \in (1, \ldots, m)$
$\mathcal{F} := \mathcal{F}_1 \cup \cdots \cup \mathcal{F}_m$
$\mathcal{L} := \mathrm{Lie}(\mathcal{L}_1 + \cdots + \mathcal{L}_m)$

Étape 2. [Clôture]
Tant qu'il existe un $N \in \mathcal{F} \setminus \{1\}$ avec $N \mathcal{L} N^{-1} \not\subseteq \mathcal{L}$
$\mathcal{L} := \mathrm{Lie}(\mathcal{L} + N \mathcal{L} N^{-1})$
Tant qu'il existe un $N \in \mathcal{F} \setminus \{1\}$ avec $N \in e^{\mathcal{L}}$, faire $\mathcal{F} := \mathcal{F} \setminus (N)$
Tant qu'il existe un $N \in \mathcal{F}^2$ avec $N \notin \mathcal{F} e^{\mathcal{L}}$ faire
Calculer $(N) = \mathcal{F}' e^{\mathcal{L}'}$
Si $\mathcal{L}' \not\subseteq \mathcal{L}$, alors $\mathcal{L} := \mathrm{Lie}(\mathcal{L} + \mathcal{L}')$, quitter la boucle, repéter l'étape 2
Sinon, $\mathcal{F} := \mathcal{F} \cup (N)$
Retourner $\mathcal{F} e^{\mathcal{L}}$

Calculs plus rapides avec la partie discrète

Représentation compacte des éléments dans $\mathcal{H} = \mathcal{G} / e^{\mathcal{L}}$

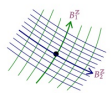
- Réduire au cas où $\mathcal{G} \subseteq \mathrm{Norm}(e^{\mathcal{L}})^o$
- Premier élément $M = B_1 = e^X$ de « la base » avec
  - $M e^{\mathcal{L}} \in \mathcal{H}$
  - $M e^{\mathcal{L}}$ génère $(e^{\mathbb{C}X} \cap \mathcal{G}) / e^{\mathcal{L}}$
  - M admet un ordre q maximal avec ces propriétés
- Posons $\mathcal{H}' := \{N \in \mathcal{H} : [M, N] = 0\}$, $\mathcal{L}' := \mathcal{L} \oplus \mathbb{C} X$, tels que

$$\mathcal{H} = (1, \ldots, M^{q-1}) (\mathcal{H}' / e^{\mathcal{L}'}).$$

- Autres éléments $B_2, \ldots, B_b$ de la base par récurrence, avec

$$\|B_1\|_{\mathcal{L}} \leqslant \cdots \leqslant \|B_b\|_{\mathcal{L}}$$

Réduction de réseaux non commutative

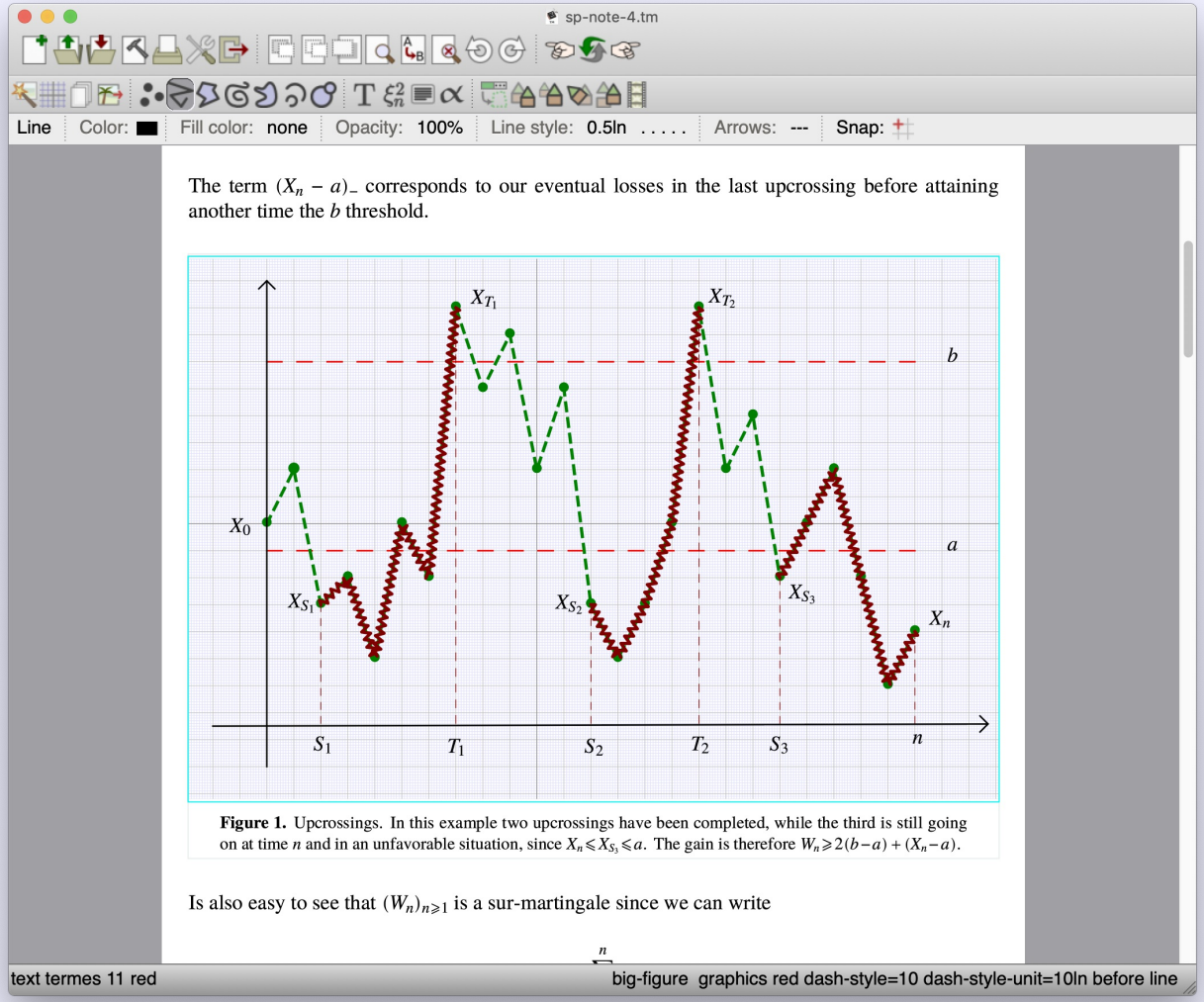


- Si $[B_i, B_j] = 0$, alors réduire en utilisant LLL.
- Si $[B_i, B_j] \neq 0$, alors $\|[B_i, B_j]\|_{\mathcal{L}} = O(\|B_i\|_{\mathcal{L}} \|B_j\|_{\mathcal{L}})$ ⇝ nouveaux éléments

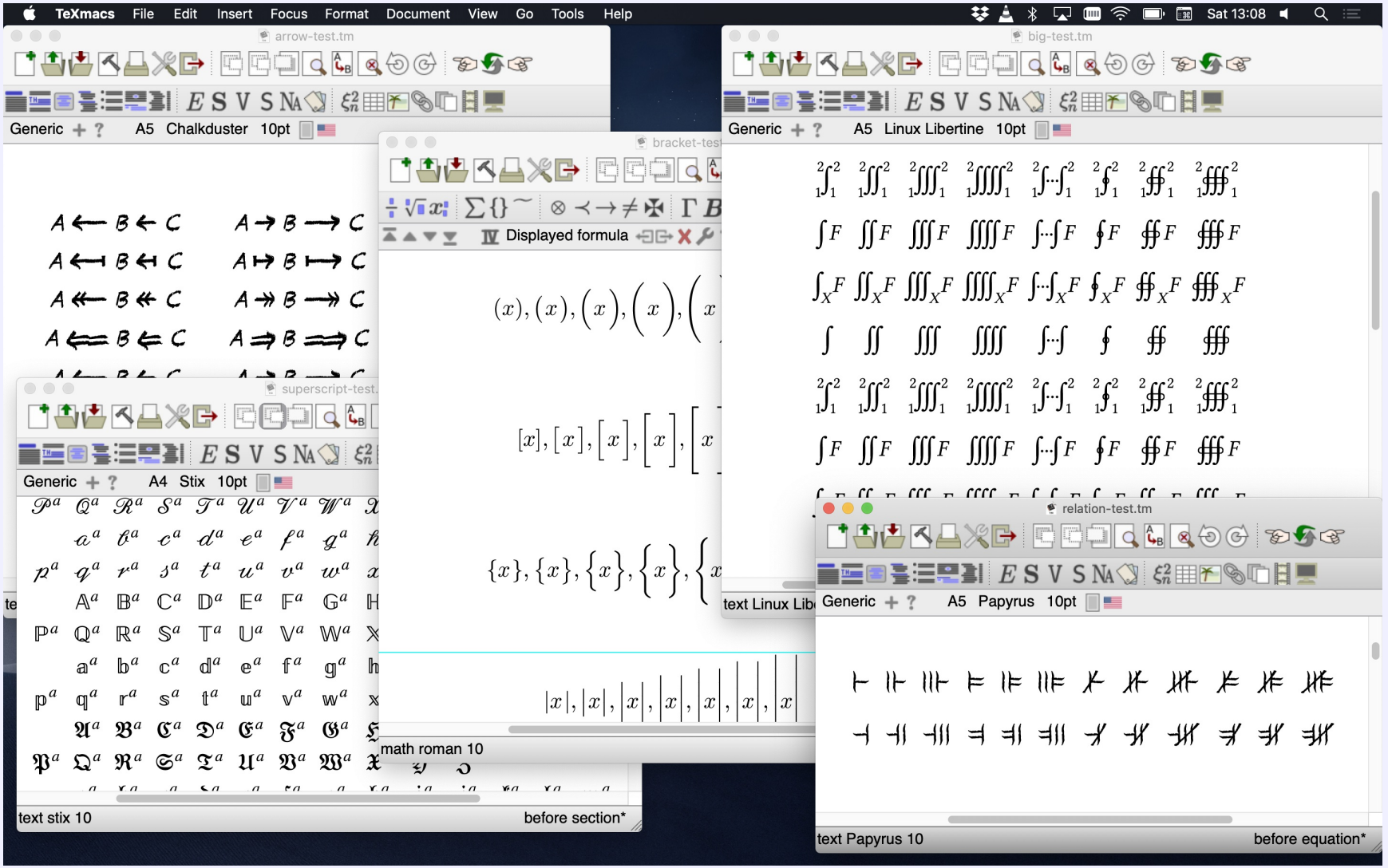
Merci !

http://www.TeXmacs.org

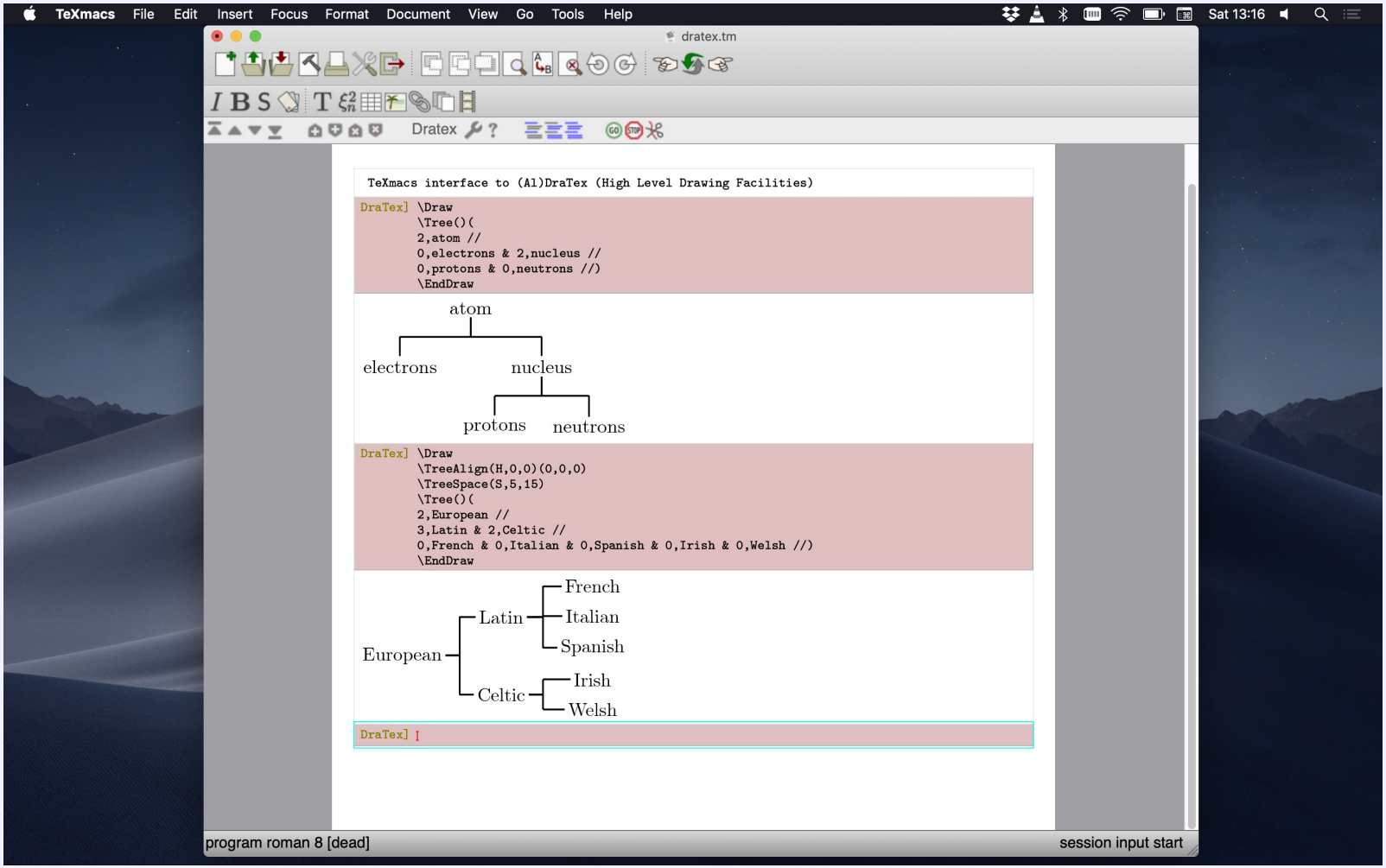
Panorama mode



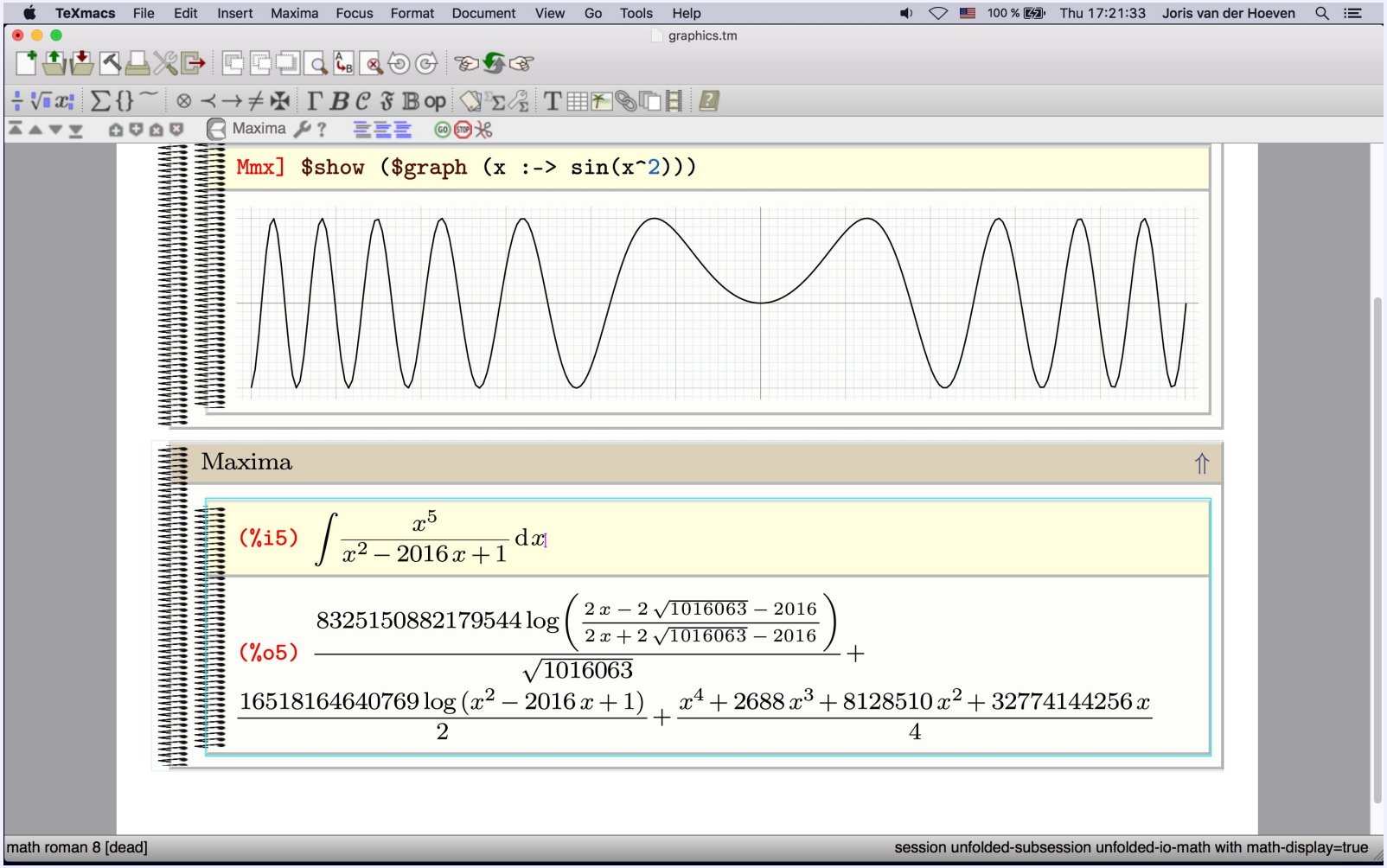
Graphics editor



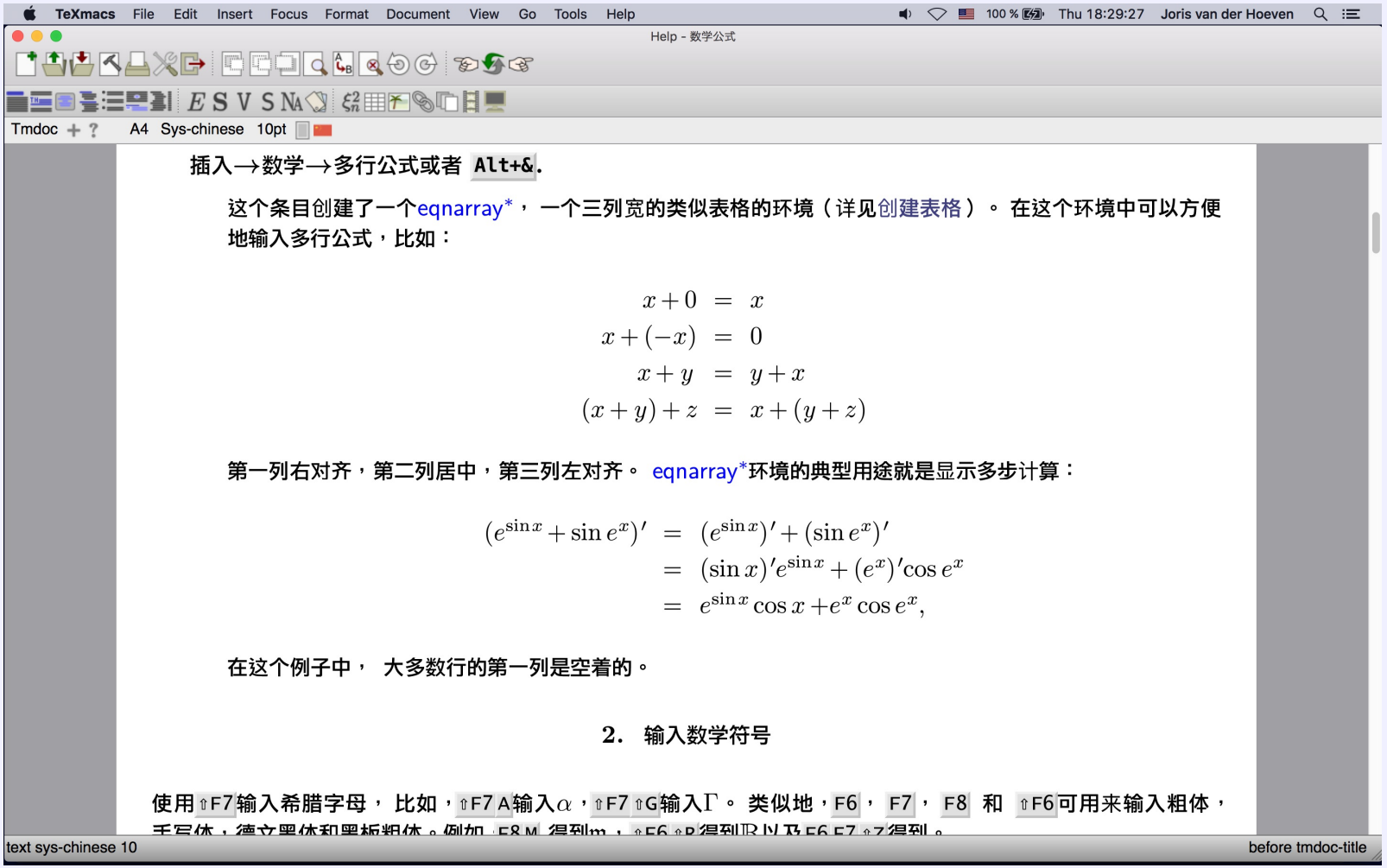
Microtypography, synthetic math fonts



Interfaces to external packages (here DraTEX)



Interfaces to external packages (here MATHEMAGIX and MAXIMA)



Support for oriental scripts

Development

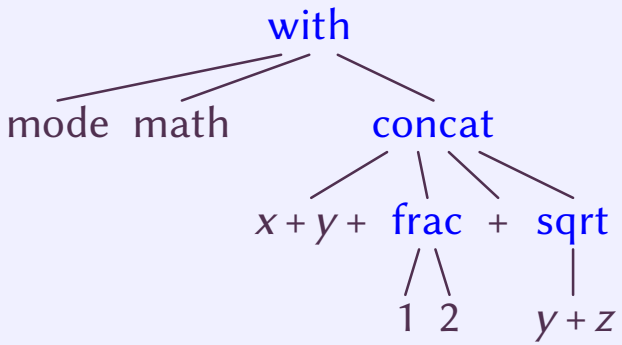
- Started in 1998 by Joris Van Der Hoeven
  - v0.2.3β released 26 Oct 1999
  - v1.0 (2002)
  - Qt backend in v1.0.7 (2008)
  - native Pdf support in v1.99.1 (2013)
  - currently version 1.99.9 (soon 2.1)
- Written in C++ (~300.000 loc) and Scheme (~150.000 loc) (from [openhub]).
- Fully modular, external dependencies (mostly) isolated via tight interfaces.
- Two UI backends: legacy X11 with custom widget library, modern Qt backend (cross-platform support).
- **GNU Guile as extension language**. C++ export basic manipulation routines and few internal datatypes.

Some of the (current) developers

TeXmacs' content model

All TeXmacs documents or document fragments can be thought of as *trees*.

For instance, the tree



typically represents the formula

$$x + y + \frac{1}{2} + \sqrt{y + z}$$

External representations

Serialization of TeXmacs documents without loss of informations

- TeXmacs format

```
<with|mode|math|x+y+<frac|1|2>+<sqrt|y+z>>
```

- XML format

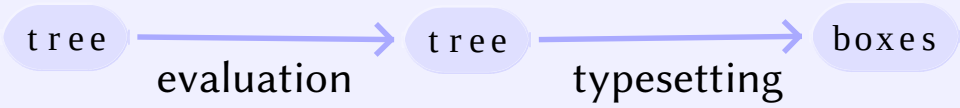
```
<frac><tm-arg>1</tm-arg><tm-arg>2</tm-arg></frac>+<sqrt>y+z</
sqrt>
```

- SCHEME format

```
(with "mode" "math" (concat "x+y+" (frac "1" "2") "+" (sqrt "y+z")))
```

Typesetting

Typesetting process converts TeXmacs trees into boxes:



The typesetting primitives are designed to be very fast and they are built-in into the editor:

> e.g. typesetting primitives for horizontal concatenations (concat), page breaks (page-break), mathematical fractions (frac), hyperlinks (hlink), and so on.

The rendering of many of the primitives may be customized through the built-in environment variables.

> e.g. the environment variable *color* specifies the current color of objects, *par-left* the current left margin of paragraphs, etc.

The stylesheet language allows the user to write new primitives (macros) on top of the built-in primitives.

> Contains primitives for defining macros, conditional statements, computations, delayed execution, etc. and a special extern tag to inject SCHEME expressions in order to write macros.

Macros

Evaluation of TeXmacs trees proceeds by reduction of the primitives, essentialy by evaluation of macro applications.

⟨assign|*hello*|⟨macro|*name*|Hello *name*, how are you today?⟩⟩

Macros have editable input fields. Examples here below (activate the macros):

⟨assign|*hello*|⟨macro|*name*|Hello name, how are you today?⟩⟩

⟨hello|dsdjskjds⟩

⟨assign|*seq*|⟨macro|*val*|(val$_1$, ..., val$_n$)⟩⟩

$$\langle \text{seq}|f\rangle = \langle \text{seq}|g\rangle$$

GUILE as extension language

TeXmacs is extendable and customizable in various ways:

- **GUILE** embedded as extension and scripting language
- A plugin system allows asyncronous communication with external programs
- Mechanism to dynamically load external code (via C interface)

GUILE is easy to embed and provides a reasonably fast implementation of SCHEME.

Why SCHEME?

1. Allows to mix programs and data in a common framework.
2. Allows to customize the language itself, by adding new programming constructs.
3. Allows to write programs on a very abstract level.

Menus

```
(menu-bind file-menu
  ("New" (new-buffer))
  ("Load" (choose-file load-buffer "Load file" ""))
  ("Save" (save-buffer))
  ...)
```

can be easily extended from user code:

```
(menu-bind insert-menu
  (former)
  ---
  (-> "Opening"
      ("Dear Sir" (insert "Dear Sir,"))
      ("Dear Madam" (insert "Dear Madam,")))
  (-> "Closing"
      ("Yours sincerely" (insert "Yours sincerely,"))
      ("Greetings" (insert "Greetings,"))))
```

Keybindings

```
(kbd-map
  ("D e f ." (make 'definition))
  ("L e m ." (make 'lemma))
  ("P r o p ." (make 'proposition))
  ("T h ." (make 'theorem)))
```

The file `my-init-buffer.scm` is executed every time a buffer is loaded, it allows some specific customizations. For example:

```
(if (not (buffer-has-name? (current-buffer)))
    (begin
      (init-style "article")
      (buffer-pretend-saved (current-buffer))))

(if (not (buffer-has-name? (current-buffer)))
    (make-session "maxima" (url->string (current-buffer))))
```

Scheme invocation

SCHEME commands can be invoked interactively (like in EMACS) using the ⌘^X shortcut.

A SCHEME session is started using the Insert→Session→Scheme menu item:

```
scheme] (define (square x) (* x x))
scheme] (square 1111111)
scheme] (kbd-map ("h i ." (insert "Hi there!")))
scheme] ;; try typing ``hi.''
```

SCHEME commands can be invoked from the command line:

```
texmacs text.tm -x "(print)" -q
```

Or scheme statement executed from inside TeXmacs macros:

```
⟨extern|(lambda (x) `(concat "Hallo " ,x))|Piet⟩
```

Scheme invocation

SCHEME commands can be invoked interactively (like in EMACS) using the ⌘^X shortcut.

A SCHEME session is started using the Insert→Session→Scheme menu item:

```
scheme] (define (square x) (* x x))
```

```
scheme] (square 1111111)
scheme] (kbd-map ("h i ." (insert "Hi there!")))
scheme] ;; try typing ``hi.''
```

SCHEME commands can be invoked from the command line:

```
texmacs text.tm -x "(print)" -q
```

Or scheme statement executed from inside TeXmacs macros:

```
⟨extern|(lambda (x) `(concat "Hallo " ,x))|Piet⟩
```

Scheme invocation

SCHEME commands can be invoked interactively (like in EMACS) using the ⌘^X shortcut.

A SCHEME session is started using the Insert→Session→Scheme menu item:

```
scheme] (define (square x) (* x x))
```

```
scheme] (square 1111111)
```

```
scheme] (kbd-map ("h i ." (insert "Hi there!")))
scheme] ;; try typing ``hi.''
```

SCHEME commands can be invoked from the command line:

```
texmacs text.tm -x "(print)" -q
```

Or scheme statement executed from inside TeXmacs macros:

```
⟨extern|(lambda (x) `(concat "Hallo " ,x))|Piet⟩
```

Scheme invocation

SCHEME commands can be invoked interactively (like in EMACS) using the ⌘^X shortcut.

A SCHEME session is started using the Insert→Session→Scheme menu item:

```
scheme] (define (square x) (* x x))
```

```
scheme] (square 1111111)
```

```
scheme] (kbd-map ("h i ." (insert "Hi there!")))
```

```
scheme] ;; try typing ``hi.''
```

SCHEME commands can be invoked from the command line:

```
texmacs text.tm -x "(print)" -q
```

Or scheme statement executed from inside TeXmacs macros:

```
⟨extern|(lambda (x) `(concat "Hallo " ,x))|Piet⟩
```

Scheme invocation

SCHEME commands can be invoked interactively (like in EMACS) using the ⌘^X shortcut.

A SCHEME session is started using the Insert→Session→Scheme menu item:

```
scheme] (define (square x) (* x x))
```

```
scheme] (square 1111111)
```

```
scheme] (kbd-map ("h i ." (insert "Hi there!")))
```

```
scheme] ;; try typing ``hi.''
```

SCHEME commands can be invoked from the command line:

```
texmacs text.tm -x "(print)" -q
```

Or scheme statement executed from inside TeXmacs macros:

```
⟨extern|(lambda (x) `(concat "Hallo " ,x))|Piet⟩
```

The tm-define macro

Contextual overloading

Function definition can depend on several run-time conditions (e.g. editor mode). This allows to develop modular user interfaces.

```
(tm-define (hello) (insert "Hello"))
(tm-define (hello) (:require (in-math?)) (insert-go-to "hello()" '(6)))
```

```
(tm-define (hello)
  (if (in-math?) (insert-go-to "hello()" '(6)) (former)))
```

```
(tm-define (my-replace what by)
  default-implementation)

(tm-define (my-replace what by)
  (:require (== what by))
  (noop))
```

```
(tm-define (square x)
  (:synopsis "Compute the square of @x")
  (:argument x "A number")
  (:returns "The square of @x")
  (* x x))
```

Used via e.g. `(help square)`. Allows for interactive input of parameters: typing ⌘^⇧X followed by `square` and ↩ and you will be prompted for "A number" on the footer (or in a dialog). Tab-completion.

```
(tm-property (choose-file fun text type)
  (:interactive #t))
```

to indicate interactive commands in menu items like:

```
("Load" (choose-file load-buffer "Load file" ""))
```

More meta

Check-marks for menu items:

```
(tm-define (toggle-session-math-input)
  (:check-mark "v" session-math-input?)
  (session-use-math-input (not (session-math-input?))))
```

```
(tm-define mouse-unfold
  (:secure #t)
  (with-action t
    (tree-go-to t :start)
    (fold)))
```

- This is a fold/unfold environment

More meta

Check-marks for menu items:

```
(tm-define (toggle-session-math-input)
  (:check-mark "v" session-math-input?)
  (session-use-math-input (not (session-math-input?))))
```

```
(tm-define mouse-unfold
  (:secure #t)
  (with-action t
    (tree-go-to t :start)
    (fold)))
```

- This is a fold/unfold environment

  It allows to toggle the display of its content by switching the tag from fold to unfold and back.

- **Passive trees** (stree)

$$\frac{a^2}{b+c}$$

is typically represented by

```
(frac (concat "a" (rsup "2")) "b+c")
```

convenient to manipulate content directly using standard SCHEME routines on lists.

- **Active trees** (tree). TEXMACS internal C++ type tree which is exported to SCHEME via the glue. Keeps track of the *position* of the tree inside the global document tree and can be used to programmatically modify documents.
- **Hybrid representation** (content). an expression of the type content is either a string, a tree or a list whose first element is a symbol and whose remaining elements are other expressions of type content.

```
scheme] (tree-set! t '(document "First line." "Second line."))
scheme] (tree-set t 1 "New second line.")
scheme] (tree-set t 0 `(strong ,(tree-ref t 0)))
```

A full example

```
(tm-define (swap-numerator-denominator t)
  (:require (tree-is? t 'frac))
  (with p (tree-cursor-path t)
    (tree-set! t `(frac ,(tree-ref t 1) ,(tree-ref t 0)))
    (tree-go-to t (cons (- 1 (car p)) (cdr p)))
    (tree-focus t)))
```

To be called as `(swap-numerator-denominator (focus-tree))`, or just add it as a structured variant to frac

```
(tm-define (variant-circulate t forward?)
  (:require (tree-is? t 'frac))
  (swap-numerator-denominator t))
```

Regular expressions

TEXMACS implements the routines `match?` and `select` for matching regular expressions and selecting subexpressions along a "path". For instance: in the current buffer search all expressions of the form

$$\frac{a}{1+\sqrt{b}}$$

where a and b are general expressions:

```
Scheme] (select (buffer-tree) '(:* (:match (frac :%1 (concat "1+" (sqrt :%1))))))
```

User preferences

```
(define-preferences
  ("Gnu's hair color" "brown" notify-gnu-hair-change)
  ("Snail's cruising speed" "1mm/sec" notify-Achilles))
```

New data formats and converters

```
(define-format blablah
  (:name "Blablah")
  (:suffix "bla"))

(converter blablah-file latex-file
  (:require (url-exists-in-path? "bla2tex"))
  (:shell "bla2tex" from ">" to))
```

When a format can be converted from or into TeXmacs, then it will automatically appear into the File→Export and File→Import menus. Similarly, when a format can be converted to POSTSCRIPT, then it also becomes a valid format for images. TeXmacs also attempts to combine explicitly declared converters into new ones.

Dialogues

```
Scheme] (user-ask "First number:"
          (lambda (a)
            (user-ask "Second number:"
              (lambda (b)
                (set-message (number->string (* (string->number a)
                                                (string->number b)))
                             "product")))))
```

Widgets

```
Scheme] (tm-widget (example3)
          (hlist
            (bold (text "Hello"))
            >>>
            (inert (explicit-buttons ("world" (display "!\n"))))))
Scheme] (top-window example3 "Some text")
Scheme]
```

```
Scheme] (define t
          (stree->tree
           '(root
             (library "Library" "$TEXMACS_PIXMAP_PATH/tm_german.xpm" 01
                      (collection "Cool stuff" 001)
                      (collection "Things to read" 002)
                      (collection "Current work" 003
                                  (collection "Forever current" 004)
                                  (collection "Read me" 005))))))
Scheme] (define dd
          (stree->tree
           '(list (library DisplayRole DecorationRole UserRole:1)
                  (collection DisplayRole UserRole:1))))
Scheme] (define (action clicked cmd-role . user-roles)
          (display* "clicked= " clicked ", cmd-role= " cmd-role
                    ", user-roles= " user-roles "\n")))
Scheme] (tm-widget (widget-library)
          (resize ("150px" "400px" "9000px") ("300px" "600px" "9000px")
            (vertical
              (bold (text "Testing tree-view"))
              ===
              (tree-view action t dd))))
Scheme] (top-window widget-library "Tree View")
Scheme]
```

Forms

```
Scheme] (tm-widget (form3 cmd)
          (resize "500px" "500px"
            (padded
              (form "Test"
                (aligned
                  (item (text "Input:")
                    (form-input "fieldname1" "string" '("one") "1w"))
                  (item === ===)
                  (item (text "Enum:")
                    (form-enum "fieldname2" '("one" "two" "three") "two" "2w"))
                  (item === ===)
                  (item (text "Choice:")
                    (form-choice "fieldname3" '("one" "two" "three") "one"))
                  (item === ===)
                  (item (text "Choices:")
                    (form-choices "fieldname4"
                                  '("one" "two" "three")
                                  '("one" "two"))))
                (bottom-buttons
                  ("Cancel" (cmd "cancel")) >>
                  ("Ok"
                   (display* (form-fields) " -> " (form-values) "\n")
                   (cmd "ok")))))))
Scheme] (dialogue-window form3 (lambda (x) (display* x "\n")) "Test of form3")
Scheme]
```

Bibliography styles

New styles can be defined via SCHEME modules like `example.scm` defined as follows:

```
(texmacs-module (bibtex example)
  (:use (bibtex bib-utils)))

(bib-define-style "example" "plain")

(tm-define (bib-format-date e)
  (:mode bib-example?)
  (bib-format-field e "year"))
```

This example style behaves in a similar way as the `plain` style, except that all dates are formatted according to our custom routine. Styles are stored in `$TEXMACS_PATH/progs/bibtex` and referred to as e.g. `tm-example` (for when used in a TEXMACS document.

Graphics

Graphics objects are also part of the TeXmacs format and can be manipulated programmatically from Scheme.

Actually, part of the graphics editor is written in Scheme.

```
Scheme] (stree->tree
          '(with gr-geometry (tuple "geometry" "200px" "100px" "center")
                 color "blue"
                (graphics (text-at "TeXmacs" (point "-2.5" "-1"))
                          (point 0 -1)
                          (line (point 0 0) (point 0 1)
                                (point 1 1) (point 1 0) (point 0 0)))))
Scheme]
```

Graphics

Graphics objects are also part of the TeXmacs format and can be manipulated programmatically from Scheme.

Actually, part of the graphics editor is written in Scheme.

```
Scheme] (stree->tree
          '(with gr-geometry (tuple "geometry" "200px" "100px" "center")
                 color "blue"
                (graphics (text-at "TeXmacs" (point "-2.5" "-1"))
                          (point 0 -1)
                          (line (point 0 0) (point 0 1)
                                (point 1 1) (point 1 0) (point 0 0)))))
```

TeXmacs

```
Scheme]
```

The future of TeXmacs

Many improvements ahead

- Version 2.1 to be released soon
- Update the backend to Qt 5 (currently Qt 4.8) [almost there]
- Adapt the scheme code to run on Guile 3. (currently Guile 1.8) [WIP]
- New website, documentations, videos [WIP]
- Jupyter plugins (protocol to interface to many computational kernels, e.g. Python, Julia, R, Haskell, Guile, ...)
- Improvements to the styling of presentations and posters [WIP]
- More documentation, more tutorial, grow community [Stackexchange proposal]
- Collaboration tools
- Bibliography management with Zotero

Many opportunities for contributions for all tastes

- From the outside
  - Write and review documentations, tutorials, videos, improve community, advertise
  - Develop plugins to your preferred system or to add your preferred feature, e.g.: literate programming tools with beautyful output
  - Write new document styles, templates, presentation styles, poster styles
  - Font tuning
- Hack the C++ code
  - Understand the code and write developer documentation
  - Improve the QT backend, fix bugs, add features, improve stability, better image handling and PDF export of TeXmacs features
  - Write new backends (COCOA), port to tablets or to the browser
- Hack SCHEME
  - Help porting to GUILE 3, improve speed
  - fix bugs, review code, add new cool features

Happy TeXmacsing!