Typst `leipzig-glossing` Documentation

1. Introduction

Interlinear morpheme-by-morpheme glosses are common in linguistic texts to give information about the meanings of individual words and morphemes in the language being studied. A set of conventions called the **Leipzig Glossing Rules** was developed to give linguists a general set of standards and principles for how to format these glosses. The most recent version of these rules can be found in PDF form at **this link**, provided by the Department of Linguistics at the Max Planck Institute for Evolutionary Anthropology.

There is a staggering variety of LaTex packages designed to properly align and format glosses (including `gb4e`, `ling-macros`, `linguex`, `expex`, and probably even more). These modules vary in the complexity of their syntax and the amount of control they give to the user of various aspects of formatting. The `typst-leipzig-glossing` module is designed to provide utilities for creating aligned Leipzig-style glosses in Typst, while keeping the syntax as intuitive as possible and allowing users as much control over how their glosses look as is feasible.

This PDF will show examples of the module's functionality and detail relevant parameters. For more information or to inform devs of a bug or other issue, visit the module's Github repository **https://github.com/neunenak/typst-leipzig-glossing**

2. Basic glossing functionality

As a first example, here is a gloss of a text in Georgian, along with the Typst code used to generate it:

from "Georgian and the Unaccusative Hypothesis", Alice Harris, 1982

ბავშვ-ი	ატირდა
bavšv-i	aṭirda
child-NOM	3S/cry/INCHO/II

The child burst out crying

```
#import "leipzig-gloss.typ": gloss
#gloss(
  header: [from "Georgian and the Unaccusative Hypothesis", Alice Harris, 1982],
  source: ([ბავშვ-ი], [ატირდა]),
  transliteration: ([bavšv-i], [aṭirda]),
  morphemes: ([child-#smallcaps[nom]], [3S/cry/#smallcaps[incho]/II]),
  translation: [The child burst out crying],
)
```

The function `gloss()` typesets *bare* interlinear glosses (including styling, see Section 2.1 and Section 2.2). Normally when adding linguistic examples use the `example()` function, which calls `gloss()` internally and includes functionality that has to do with linguistic examples: numbering, labelling/referencing and sub-examples. `gloss()` is to be used when only the basic function of typesetting interlinear glosses is needed. Unlike `gloss()`, the function `example()` does not take the different parameters directly, but takes a list of dictionaries whose keys and values correspond to `gloss()`'s parameters (with added options such as `label` and `numbering`). It also indents the text even when numbering is not enabled:

from “Georgian and the Unaccusative Hypothesis”, Alice Harris, 1982

ბავშვ-ი	ატირდა
bavšv-i	aṭirda
child-NOM	3S/cry/INCHO/II

The child burst out crying

```
#example(
  (
    header: [from "Georgian and the Unaccusative Hypothesis", Alice Harris,
1982],
    source: ([ბავშვ-ი], [ატირდა]),
    transliteration: ([bavšv-i], [aṭirda]),
    morphemes: ([child-#smallcaps[nom]], [3S/cry/#smallcaps[incho]/II]),
    translation: [The child burst out crying],
  )
)
```

2.1. Styling

Each of the aforementioned text parameters has a corresponding style parameter, formed by adding `-style` to its name: `header-style`, `source-style`, `transliteration-style`, `morphemes-style`, and `translation-style`. These parameters allow you to specify formatting that should be applied to each entire line of the gloss. This is particularly useful for the aligned gloss itself, since otherwise one would have to modify each content item in the list individually.

In addition to these parameters, Typst's usual content formatting can be applied to or within any given content block in the gloss. Formatting applied in this way will override any contradictory line-level formatting.

This text is about eating your head.

I'm	*eat-ing*	*your*	*head*
1SG.SBJ=to.be	eat-PROG	2SG.POSS	head

I'm eating your head!

```
#example(
  (
    header: [This text is about eating your head.],
    header-style: text.with(weight: "bold", fill: green),
    source: (text(fill:black)[I'm], [eat-ing], [your], [head]),
    source-style: text.with(style: "italic", fill: red),
    morphemes: ([1#sg.#sbj\=to.be], text(fill:black)[eat-#prog], [2#sg.#poss],
[head]),
    morphemes-style: text.with(fill: blue),
    translation: text(weight: "bold")[I'm eating your head!],
  )
)
```

An example for English which exhibits some additional styling, and uses imports from another file for common glossing abbreviations:

I'm eat-ing your head
1SG.SBJ=to.be eat-PROG 2SG.POSS head
"I'm eating your head!"

```
#example(
  (
    source: ([I'm], [eat-ing], [your], [head]),
    source-style: (item) => text(fill: red)[#item],
    morphemes: ([1#sg.#sbj\=to.be], [eat-#prog], [2#sg.#poss], [head]),
    morphemes-style: text.with(size: 10pt, fill: blue),
    translation: text(weight: "semibold")[I'm eating your head!],
    translation-style: (item) => ["#item"],
  )
)
```

The `gloss()` function has three pre-defined parameters for glossing levels: `source`, `transliteration`, and `morphemes`. It also has two parameters for unaligned text: `header` for text that precedes the gloss, and `translation` for text that follows the gloss.

The `morphemes` param can be skipped, if you just want to provide a source text and translation, without a gloss:

Trato de entender, debo comprender, qué es lo que ha hecho conmigo
I try to understand, I must comprehend, what she has done with me

```
#example(
  (
    source: ([Trato de entender, debo comprender, qué es lo que ha hecho
conmigo],),
    source-style: emph,
    translation: [I try to understand, I must comprehend, what she has done with
me],
  )
)
```

Note that it is still necessary to wrap the `source` argument in an array of length one.

Here is an example of a lengthy gloss that forces a line break:

Ich **arbeite** **ein** **Jahr** **um** **das** **Geld** **zu** **verdienen,** **das** **dein** **Bruder**
I work one year to the money to earn, that your brother
an **einem** **Wochenende** **ausgibt.**
on one weekend spends.
“I work one year to earn the money that your brother spends in one weekend”

```
#example(
  (
    source:    ([Ich],[arbeite],[ein],[Jahr],[um],[das],[Geld], [zu],
[verdienen,],[das], [dein],[Bruder], [an],[einem],[Wochenende],[ausgibt.]),
    source-style: text.with(weight: "bold"),
    morphemes: ([I],  [work],[  one], [year],[to],[the],[money],[to],[earn,],
[that],[your],[brother],[on],[one],  [weekend],   [spends.]),
    translation: ["I work one year to earn the money that your brother spends in
one weekend"]
  )
)
```

2.2. Additional lines

To add more than three glossing lines, there is an additional parameter `additional-lines` that can take a list of arbitrarily many more glossing lines, which will appear below those specified in the aforementioned parameters:

Hunzib (van den Berg 1995:46)

ождиг	хоʰхе	мукъер
oʒdig	χõχe	muq’er
ož-di-g	xõxe	m-uq’e-r
boy-OBL-AD	tree(G4)	G4-bend-PRET
at boy	tree	bent

“Because of the boy, the tree bent.”

```
#example(
  (
    header: [Hunzib (van den Berg 1995:46)],
    source: ([ождиг],[хо#super[н]хе],[мукъер]),
    transliteration: ([oʒdig],[χõχe],[muq’er]),
    morphemes: ([ož-di-g],[xõxe],[m-uq'e-r]),
    additional-lines: (
        ([boy-#smallcaps[obl]-#smallcaps[ad]], [tree(#smallcaps[g4])],
[#smallcaps[g4]-bend-#smallcaps[pret]]),
        ([at boy], [tree], [bent]),
    ),
    translation: ["Because of the boy, the tree bent."]
  )
)
```

2.3. Sub-examples

Sub-examples can be achieved by adding more dictionaries of glossing fields, separated by commas. A global `header` field for the set can be added.

Coptic; transliterated and glossed based on *Coptic in 20 lessons* (2007) by Layton Bently (§ 28)

(a) Indefinite articles
hen-maein mn-hen-špêre
INDF.PL-sign with-INDF.PL-wonder
signs and wonders

(b) Definite articles
m-maein mn-ne-špêre
DEF.PL-sign with-DEF.PL-wonder
the signs and the wonders

(c) Definite pronouns
nei-maein mn-nei-špêre
DEM.PL-sign with-DEM.PL-wonder
these signs and these wonders

```
#example(
  header: [Coptic; transliterated and glossed based on _Coptic in 20 lessons_
(2007) by Layton Bently (§~28)],
  (
    header: [Indefinite articles],
    source: ([hen-maein], [mn-hen-špêre]),
    morphemes: ([#indf.#pl\-sign], [with-#indf.#pl\-wonder]),
    translation: [signs and wonders]
  ),
  (
    header: [Definite articles],
    source: ([m-maein], [mn-ne-špêre]),
    morphemes: ([#def.#pl\-sign], [with-#def.#pl\-wonder]),
    translation: [the signs and the wonders]
  ),
  (
    header: [Definite pronouns],
    source: ([nei-maein], [mn-nei-špêre]),
    morphemes: ([#dem.#pl\-sign], [with-#dem.#pl\-wonder]),
    translation: [these signs and these wonders]
  ),
)
```

2.4. Numbering Glosses

The `example()` function takes a boolean parameter `numbering` which will add an incrementing count to each gloss. A function `numbered-example` is exported for convenience; this is defined as simply `#let numbered-example = example.with(numbering: true)`, and is called with the same arguments as `example()`:

(1) გვ-ფრცქვნ-ი
gv-prtskvn-i
1PL.OBJ-peel-FMNT
You peeled us

(2) მ-ფრცქვნ-ი
m-prtskvn-i
1SG.OBJ-peel-FMNT
You peeled me

```
#example(
  (
    source: ([გვ-ფრცქვნ-ი],),
    transliteration: ([gv-prtskvn-i],),
    morphemes: ([1#pl.#obj\-peel-#fmnt],),
    translation: "You peeled us",
  ),
  numbering: true,
)

#numbered-example(
  (
    source: ([მ-ფრცქვნ-ი],),
    transliteration: ([m-prtskvn-i],),
    morphemes: ([1#sg.#obj\-peel-#fmnt],),
    translation: "You peeled me",
  )
)
```

The displayed number for numbered glosses is iterated for each numbered gloss that appears throughout the document. Unnumbered glosses do not increment the counter for the numbered glosses.

The gloss count is controlled by the Typst counter variable `example-count`. This variable can be imported from the `leipzig-gloss` package and manipulated using the standard Typst counter functions to control gloss numbering:

(21) from *Standard Basque: A Progressive Grammar* by Rudolf de Rijk, quoting P. Charriton
Bada beti guregan zorion handi baten nahia.
There always is in us a will for a great happiness.

```
#example-count.update(20)

#numbered-example(
  (
    header: [from _Standard Basque: A Progressive Grammar_ by Rudolf de Rijk,
quoting P. Charriton],
    source: ([Bada beti guregan zorion handi baten nahia.],),
    translation: [There always is in us a will for a great happiness.],
  )
)
```

References to individual examples can be achieved using the `label` argument and the referencing mechanism of Typst:

See Example 22:

(22) Middle Welsh; modified from *Grammatical number in Welsh* (1999) by Silva Nurmio (§ 2.1.1)

ac	ny	allvs	y	dewinyon	atteb	idav
and	NEG	be_able.PRET.3SG	DEF	sorcerer.PL	answer.INF	to.3SG.M

and the sorcerers could not answer him

As we have seen in Example 22, […].

```
See @sorcerers:

#numbered-example(
  (
    header: [Middle Welsh; modified from _Grammatical number in Welsh_ (1999) by
Silva Nurmio (§~2.1.1)],
    source: ([ac], [ny], [allvs], [y], [dewinyon], [atteb], [idav]),
    morphemes: ([and], [#neg], [be_able.#smallcaps[pret].3#sg],
[#smallcaps[def]], [sorcerer.#pl], [answer.#smallcaps[inf]],
[to.3#sg.#smallcaps[m]]),
    translation: [and the sorcerers could not answer him],
  ),
  label: "sorcerers",
  label-supplement: [Example]
)

As we have seen in @sorcerers, […].
```

Labelling uses the Typst **figure** document element. The `label-supplement` parameter fills in the `suppliment` parameter of a `figure`, which is `[example]` by default. Note that `label` and `label-supplement` are top-level arguments of `example()` and `numbered-example()`, not of the interlinear glosses surrounded by `(` and `)`.

Labelling of sub-examples is possible as well, using the same `label` and `label-supplement` fields but within the parentheses surrounding the sub-example in question.

(23) Hausa; from *Toward a functional typology of adpositions* (2022) by Zygmunt Frajzyngier (§ 3.2)

(a) àkwai mutânè dà yawà a kanò
exist People ASSC many PRED Kano
There are a lot of people in Kano.

(b) àkwai makaɗantā a nan gàrin
exist school PRED DEM town
There is a school in this town.

In example 23 there are two sub-examples: example 23a deals with people and example 23b with a school.

```
#numbered-example(
    header: [Hausa; from _Toward a functional typology of adpositions_ (2022) by
Zygmunt Frajzyngier (§~3.2)],
    label: "hausa",
    (
        source: ([àkwai], [mutānè], [dà], [yawā], [a], [kanō]),
        morphemes: ([exist], [People], [#smallcaps[assc]], [many], [#pred],
[Kano]),
        translation: [There are a lot of people in Kano.],
        label: "people"
    ),
    (
        source: ([àkwai], [makaɗantā], [a], [nan], [gàrin]),
        morphemes: ([exist], [school], [#pred], [#dem], [town]),
        translation: [There is a school in this town.],
        label: "school",
    ),
)

In @hausa there are two sub-examples: @people deals with people and @school with
a school.
```

3. Standard Abbreviations

The Leipzig Glossing Rules define a commonly-used set of short abbreviations for grammatical terms used in glosses, such as ACC for “accusative (case)”, or PTCP for “participle” (see “Appendix: List of Standard Abbreviations in the Leipzig Glossing Rules document)

By convention, these are typeset using SMALLCAPS. This package contains a module value `abbreviations`. Individual abbreviations may be accessed either with Typst field access notation or by importing them from `abbreviations`:

(from *Why Caucasian Languages?*, by Bernard Comrie, in *Endangered Languages of the Caucasus and Beyond*)
[qálɐ-m ∅-kw'-á] ɬ'ə́-r
city-OBL 3SG-go-PRF man-ABS
The man who went to the city.

```
#import "leipzig-gloss.typ": abbreviations
#import abbreviations: obl, sg, prf

#example(
  (
    header: [(from _Why Caucasian Languages?_, by Bernard Comrie, in _Endangered
Languages of the Caucasus and Beyond_)],
    source: ([\[qálɐ-m], [ø-kw'-á\]], [ɬ'ə́-r]),
    morphemes: ([city-#obl], [3#sg\-go-#prf], [man-#abbreviations.abs]),
    translation: "The man who went to the city."
  )
)
```

The full list of abbreviations is as follows:

3.1. Full list of abbreviations

1 - `1` - first person
2 - `2` - second person
3 - `3` - third person
A - `a` - agent-like argument of canonical transitive verb
ABL - `abl` - ablative
ABS - `abs` - absolutive
ACC - `acc` - accusative
ADJ - `adj` - adjective
ADV - `adv` - adverb(ial)
AGR - `agr` - agreement
ALL - `all` - allative
ANTIP - `antip` - antipassive
APPL - `appl` - applicative
ART - `art` - article
AUX - `aux` - auxiliary
BEN - `ben` - benefactive
CAUS - `caus` - causative
CLF - `clf` - classifier
COM - `com` - comitative
COMP - `comp` - complementizer
COMPL - `compl` - completive
COND - `cond` - conditional
COP - `cop` - copula
CVB - `cvb` - converb
DAT - `dat` - dative
DECL - `decl` - declarative
DEF - `def` - definite
DEM - `dem` - demonstrative

DET - `det` - determiner
DIST - `dist` - distal
DISTR - `distr` - distributive
DU - `du` - dual
DUR - `dur` - durative
ERG - `erg` - ergative
EXCL - `excl` - exclusive
F - `f` - feminine
FOC - `foc` - focus
FUT - `fut` - future
GEN - `gen` - genitive
IMP - `imp` - imperative
INCL - `incl` - inclusive
IND - `ind` - indicative
INDF - `indf` - indefinite
INF - `inf` - infinitive
INS - `ins` - instrumental
INTR - `intr` - intransitive
IPFV - `ipfv` - imperfective
IRR - `irr` - irrealis
LOC - `loc` - locative
M - `m` - masculine
N - `n` - neuter
N- - `n-` - non- (e.g. NSG nonsingular, NPST nonpast)
NEG - `neg` - negation, negative
NMLZ - `nmlz` - nominalizer/nominalization
NOM - `nom` - nominative
OBJ - `obj` - object
OBL - `obl` - oblique
P - `p` - patient-like argument of canonical transitive verb
PASS - `pass` - passive
PFV - `pfv` - perfective
PL - `pl` - plural
POSS - `poss` - possessive
PRED - `pred` - predicative
PRF - `prf` - perfect
PRS - `prs` - present
PROG - `prog` - progressive
PROH - `proh` - prohibitive
PROX - `prox` - proximal/proximate
PST - `pst` - past
PTCP - `ptcp` - participle
PURP - `purp` - purposive
Q - `q` - question particle/marker
QUOT - `quot` - quotative
RECP - `recp` - reciprocal
REFL - `refl` - reflexive
REL - `rel` - relative
RES - `res` - resultative

S - `s` - single argument of canonical intransitive verb
SBJ - `sbj` - subject
SBJV - `sbjv` - subjunctive
SG - `sg` - singular
TOP - `top` - topic
TR - `tr` - transitive
VOC - `voc` - vocative

3.2. Custom abbreviations

Custom abbreviations may be defined using the `abbreviations.emit-abbreviation` function:

(from *Georgian: A Structural Reference Grammar*, by George Hewitt)
g-nax-av-en
you$_{2}$-see(FUT)$_{4}$-TS$_{7}$-they$_{11}$
they will see you

```
#import "leipzig-gloss.typ": abbreviations
#import abbreviations: obl, sg, prf, fut, emit-abbreviation

#let ts = emit-abbreviation("TS")

#example(
  (
    header: [(from _Georgian: A Structural Reference Grammar_, by George
Hewitt)],
    source: ([g-nax-av-en],),
    morphemes: ([you#sub[2]-see(#fut)#sub[4]-#ts#sub[7]-they#sub[11]],),
    translation: "they will see you",
  )
)
```

3.3. Building used-abbreviations pages

A user of `leipzig-glossing` might wish to generate an introductory page displaying which abbreviations were actually used in the document. The `abbreviations.with-used-abbreviations` function may be used for this purpose; see the `abbreviations-used-example.typ` file in `leipzig-glossing` source for an example.

4. Further Example Glosses

These are the first twelve example glosses given in **https://www.eva.mpg.de/lingua/pdf/Glossing-Rules.pdf**. along with the Typst markup needed to generate them:

(1) Indonesian (Sneddon 1996:237)

Mereka di Jakarta sekarang.
they in Jakarta now
They are in Jakarta now

```
#numbered-example(
  (
    header: [Indonesian (Sneddon 1996:237)],
    source: ([Mereka], [di], [Jakarta], [sekarang.]),
    morphemes: ([they], [in], [Jakarta], [now]),
    translation: "They are in Jakarta now",
  )
)
```

(2) Lezgian (Haspelmath 1993:207)

Gila abur-u-n ferma hamišaluğ güğüna amuq'-da-č.
now they-OBL-GEN farm forever behind stay-FUT-NEG
Now their farm will not stay behind forever.

```
#numbered-example(
  (
    header: [Lezgian (Haspelmath 1993:207)],
    source: ([Gila], [abur-u-n], [ferma], [hamišaluğ], [güğüna], [amuq'-da-č.]),
    morphemes: ([now], [they-#obl\-#gen], [farm], [forever], [behind], [stay-
#fut\-#neg]),
    translation: "Now their farm will not stay behind forever.",
  )
)
```

(3) West Greenlandic (Fortescue 1984:127)

palasi=lu niuirtur=lu
priest=and shopkeeper=and
both the priest and the shopkeeper

```
#numbered-example(
  (
    header: [West Greenlandic (Fortescue 1984:127)],
    source: ([palasi=lu], [niuirtur=lu]),
    morphemes: ([priest=and], [shopkeeper=and]),
    translation: "both the priest and the shopkeeper",
  )
)
```

(4) Hakha Lai
a-nii -láay
3SG-laugh-FUT
s/he will laugh

```
#numbered-example(
  (
    header: [Hakha Lai],
    source: ([a-nii -láay],),
    morphemes: ([3#sg\-laugh-#fut],),
    translation: [s/he will laugh],
  )
)
```

(5) Russian

My	s	Marko	poexa-l-i	avtobus-om	v	Peredelkino
1PL	COM	Marko	go-PST-PL	bus-INS	ALL	Peredelkino
we	with	Marko	go-PST-PL	bus-by	to	Peredelkino

Marko and I went to Perdelkino by bus

```
#numbered-example(
  (
    header: [Russian],
    source: ([My], [s], [Marko], [poexa-l-i], [avtobus-om], [v], [Peredelkino]),
    morphemes: ([1#pl], [#com], [Marko], [go-#pst\-#pl], [bus-#ins], [#all],
[Peredelkino]),
    additional-lines: (([we], [with], [Marko], [go-#pst\-#pl], [bus-by], [to],
[Peredelkino]),),
    translation: "Marko and I went to Perdelkino by bus",
  )
)
```

(6) Turkish
çık-mak
come.out-INF
to come out

```
#numbered-example(
  (
    header: [Turkish],
    source: ([çık-mak],),
    morphemes: ([come.out-#inf],),
    translation: "to come out",
  )
)
```

(7) Latin
insul-arum
island-GEN-PL
of the islands

```
#numbered-example(
  (
    header: [Latin],
    source: ([insul-arum],),
    morphemes: ([island-#gen\-#pl],),
    translation: "of the islands",
  )
)
```

(8) French
aux chevaux
to-ART-PL horse.PL
to the horses

```
#numbered-example(
  (
    header: [French],
    source: ([aux], [chevaux]),
    morphemes: ([to-#art\-#pl],[horse.#pl]),
    translation: "to the horses",
  )
)
```

(9) German
unser-n Väter-n
our-DAT-PL father.PL-DAT.PL
to our fathers

```
#numbered-example(
  (
    header: [German],
    source: ([unser-n], [Väter-n]),
    morphemes: ([our-#dat\-#pl],[father.#pl\-#dat.#pl]),
    translation: "to our fathers",
  )
)
```

(10) Hittite (Lehmann 1982:211)

n=an	apedani	mehuni	essandu.
CONN=him	that.DAT.SG	time.DAT.SG	eat.they.shall

They shall celebrate him on that date

```
#numbered-example(
  (
    header: [Hittite (Lehmann 1982:211)],
    source: ([n=an], [apedani], [mehuni],[essandu.]),
    morphemes: ([#smallcaps[conn]=him], [that.#dat.#sg], [time.#dat.#sg],
[eat.they.shall]),
    translation: "They shall celebrate him on that date",
  )
)
```

(11) Jaminjung (Schultze-Berndt 2000:92)

nanggayan	guny-bi-yarluga?
who	2DU.A.3SG.P-FUT-poke

Who do you two want to spear?

```
#numbered-example(
  (
    header: [Jaminjung (Schultze-Berndt 2000:92)],
    source: ([nanggayan], [guny-bi-yarluga?]),
    morphemes: ([who], [2#du.#A.3#sg.#P\-#fut\-poke]),
    translation: "Who do you two want to spear?",
  )
)
```

(12) Turkish (cf. 6)

çık-mak
come_out-INF

'to come out'

```
#numbered-example(
  (
    header: [Turkish (cf. 6)],
    source: ([çık-mak],),
    morphemes: ([come_out-#inf],),
    translation: ['to come out'],
  )
)
```