# Describing tonal processes with formulas

Kirill Maslinsky

Version 1b

## Contents

**1 Synopsis** 2

**2 Underlying form** 3
- 2.1 Segmental part . . . 3
- 2.2 Tonemes . . . 6
- 2.3 Repeated m-items . . . 7

**3 Surface form** 7
- 3.1 Boundaries and spans . . . 7
- 3.2 Tonemes in the surface form . . . 8
- 3.3 Segments in the surface form . . . 10

**4 Edges and pauses** 11

**5 Correspondence between underlying and surface forms** 12
- 5.1 Implicit order-based correspondence . . . 12
- 5.2 Index notation . . . 12

**6 Definitions** 13

**7 Tonal operations** 15

**8 Morphosyntactic constraints** 16

**9 Allotony** 19

**A Symbols at a glance** 19

**B Repeated terms** 22

**C Order of the marks on a term** 22

We define *tonal process* as a situation where surface realizations of the same m-item in different contexts are not phonologically equivalent with respect to tone. This refers to alternations where either a set of tonal targets or their position relative to other segments of an m-item varies systematically depending on the context. By this definition, allotony is a tonal process too, at least where the allotones of a toneme are distributed over different realizations of the same morpheme.

The existing formulas for tonal processes in the ThotDB reflect the legacy of the autosegmental phonological paradigm. The goal of this document is to suggest a formal system for describing the tonal processes that will use the same categories and annotation conventions that are used in our text annotation.

## 1 Synopsis

The general structure of the formula is as follows:

```
UNDERLYING_FORM > SURFACE_FORM [/ CONSTRAINTS] [where DEFINITIONS]
```

Underlying form and surface form blocks of the formula reuse the corresponding conventions for underlying and surface annotation lines from the annotation rules. The constraints and definitions blocks are optional. The constraints block follows a slash (surrounded by spaces) and lists the morphosyntactic conditions that limit the range of contexts where the process applies. The definitions block follows the word *where* and restricts the values that the variables of the formula can take (§6). The word *where* is reserved: it cannot be used for anything else in a formula.

The signs that divide the parts, `>`, `/` and `where`, must be surrounded by spaces; one space on each side is the recommended style, but more are tolerated. The same holds for `=` and `!=` in the definitions. Inside the underlying and the surface forms, on the contrary, a space is meaningful (it is a word boundary), and exactly one is written.

Some other conditions relevant for defining the context of a rule's application are reflected in the underlying and surface forms themselves. The shape of the syllables is shown with the help of template variables (§2.1), and the edges of words and phrases are indicated by boundary markers (§4).

Here is a simple formula example (without definitions):

```
(H x)-(0 y) > [H x-y] / (root)-(suffix)
```

This formula characterizes a simple spread to the right of the High toneme of the lexical root that expands onto a toneless suffix (that the H belongs to a root is stated by the constraint after the slash). Note that $x$ and $y$ are variables that stand for the segmental co-exponents of the corresponding m-items. Using variables communicates that no specific constraints on the form of the segmental co-exponents are declared for this process.

A process can be described by several alternative formulas, for instance when it applies to forms of different shapes.

In the following sections, the detailed conventions will be discussed for each component of the formula.

## 2 Underlying form

The underlying part of the formula is a sequence of m-items, separated by the same boundary markers as in the underlying annotation line: spaces between prosodic words, hyphens between morphemes, equals signs before clitics, tildes between a reduplicant and its base, and backslashes before tonal morphemes that have no segments of their own.

As in the annotation rules, the m-items carry no syllabification. Dots and commas appear inside an m-item only before an alignment point (^), to reflect the relative position of a toneme that is counted in syllables or moras, exactly as in the annotation rules. Alignment points and the stress mark (*) are also used as in the annotation rules. The stress mark is put before the stressed syllable, for instance, the second syllable is stressed in the following example: `(H σ*σ)`.

### 2.1 Segmental part

The segmental part of an m-item is written with four kinds of terms.

**Chain variables.** When the specific segmental shape of a co-exponent is irrelevant, it is designated by a chain variable: one of the letters $x$, $y$, $z$, $w$. A chain variable stands for any segmental material of any non-zero length. A chain variable that occurs twice in the underlying form stands for the same string both times: `(H x) (L x)` is the same segmental word said once with H and once with L. A morpheme that has no segments is written as an m-item with only a tonal part, like `(L)`. When more than four chain variables are needed, digits can be added: `x1`, `x2`, etc. Other letters, including those reserved for the CV templates and indices described below, can be used as variables when (re)defined in the definitions block of the formula (§6).

**Syllables and moras.** When a rule applies in the context where a particular number of syllables or moras is relevant, it is shown by placeholders: σ stands for exactly one syllable, and μ for exactly one mora. They can be combined with chain variables: `(H σσ)` is a disyllabic word, and `(H xσ)` is a word of any length whose last syllable is singled out. A plus sign after a placeholder means "one or more": `σ+` stands for one or more syllables.

**CV templates.** When the shape of the segments is relevant, it can be made explicit using the common phonological $CV$ convention. Importantly, all the letters of a template are lowercase. Using lowercase template letters keeps tonal and segmental parts of the notation visually distinct: segmental variables are always lowercase, while tonal part is uppercase. For instance:

| | |
|---|---|
| `(H cv)` | simple open syllable |
| `(H cv:)` | long vowel |
| `(H cvn)` | nasal-final syllable |
| `(H cvc)` | closed syllable |

A template shows the sequence of segments letter by letter. How the segments divide into syllables, if this matters for the process, is shown with dots in the surface form. The following letters have a pre-defined interpretation:

| | |
|---|---|
| `v` | any vowel |
| `v:` | long vowel |
| `vv` | bivocalic core |
| `c` | any consonant |
| `t` | voiceless obstruent |
| `d` | voiced obstruent |
| `b` | obstruent |
| `r` | sonorant |
| `n` | nasal consonant |
| `g` | glide |
| `f` | fricative |
| `s` | sibilant |
| `q` | uvular |
| `h` | glottal |

These letters can be used in any combination to form a template, and each letter of a template stands for a segment of its own, so that the formula can refer to it separately. For instance, here is a docking of a Low toneme onto a final nasal:

```
(H cvn)\(L) > [H cv][L n]
```

A plus sign after a template letter means "one or more segments of this class", as it does after σ: `c+` is one or more consonants, so `(H c+v)` is a syllable with an onset cluster of any length. The plus sign may also follow a custom class defined in the definitions block, `c1+` (§6).

When necessary, the dot in the surface form can be used to denote the syllable boundary, as in the surface annotation line.

```
(H cv)-(0 cv)\(L) > [H cv].[L cv]
```

The same applies to comma as a moraic boundary. Without a dot or a comma, the formula says nothing about syllabification: the first formula above holds whether or not the nasal forms a syllable of its own (§3.1).

Any custom classes not listed above, such as, e.g., coronals, prenasalized consonants, a set of depressor consonants for a particular language, or particular classes of vowels, can be defined for a given formula in the definitions block (§6). The name of a custom class

follows the rules for variable names given at the end of this section. It may be a single letter, a letter with digits (`c1`), or even a predefined letter or an index letter, if it is redefined in the definitions block. While mnemonic names for classes are generally preferable to a human reader, it is not recommended to redefine the letters reserved for indices (*i j k m*) when this is avoidable without the loss of readability.

Toneme labels of some languages contain lowercase letters, like `xH` in §9. They are an exception to the "tonal part is uppercase" rule, and are recognised as tonal by their position: the tonal part comes first in an m-item, and right after the opening bracket in a tonal span.

**Literal segments.** A specific phoneme or a form can be used in the segmental part in double quotes, e.g., `"k"` stands for a phoneme *k*. When a process is restricted to particular morphemes or lexemes, their segments can be written verbatim, with the diacritics if needed: `(L "kèe")`. Unquoted literals are not allowed. When it is rather the identity of a morpheme than its form that matters, it is better identified by its gloss in the constraints (§8).

A quoted literal is a single term, a string of unspecified length, like a chain variable. Its segments are not counted, so a template that contains a literal, such as `cv"mb"v`, has no fixed length. An index put on a literal (§5.2) applies to the whole literal: `"kèe"_i`. Two literals are the same term when they are the same string. (Diacritics can be encoded in more than one way; formulas are brought to one encoding, Unicode NFC, before they are compared.)

Adjacent quoted strings are separate terms. If a formula needs to specify that on the surface the literal form is divided between tonal spans, the literal must therefore be divided in the underlying form too.

```
(L "kè""e")\(H) > [L "kè"][H "e"]
```

The same formula with an undivided literal is **incorrect**:

```
(L "kèe")\(H) > [L "kè"][H "e"]
```

**Variable type mixtures.** Any types of segmental variables defined above can be mixed in the same m-item. For instance, templates can be mixed with literal segments, e.g., a class of CVC syllables that end in *k* can be written as `cv"k"`. Chain variables can be mixed with literals in the same way: `x"k"` would stand for any string ending in *k*. Note that a sequence of terms is read as a sequence: `σ"k"` is a syllable *followed by k*, not a syllable ending in *k*. A syllable ending in *k* is a σ narrowed down in the definitions: `σ where σ = v"k" or cv"k" or ccv"k"`. A σ narrowed down this way keeps its meaning everywhere in the formula, so `σ+` then stands for one or more syllables of the defined shape.

**Variable names.** The name of a segmental variable is always a single lowercase letter (or σ, μ), optionally followed by digits: `c`, `c1`, `c2`. Longer variable names are not allowed, because the terms of a template are written without separators: in `crvn` every letter is a term of its own. The digits provide additional variable names for custom classes: `c1` and `c2` are two variables distinct from `c` and from each other.

In the segmental part, every letter outside quotes must be a chain variable, a σ or μ, a template letter, or a name defined in the definitions block. Any other letter is an error.

## 2.2 Tonemes

When individual tonemes participate in the process, the toneme is just indicated in the tonal part of the m-item, as usual. For instance, a (L) tonal morpheme docked to the last syllable of a high-toned disyllabic word would look like this:

```
(H σσ)\(L) > [H σ][L σ]
```

The tonal morpheme is joined with a backslash, as in the annotation rules.

When the identity of a particular toneme is irrelevant, i.e., any toneme can stand in a given position, it is denoted by a special variable $T$. For example, a tonal spread on a non-tonal morpheme:

```
(T x)-(0 y) > [T x-y]
```

Toneme labels are always built by concatenating the tonal levels of a closed list: *xxL*, *xL*, *L*, *M*, *H*, *xH*, *xxH*, and *D* for downstep (`HL`, `LxH`, `xHM`). An uppercase name outside this pattern, such as $T$ or the keywords `SBTR`, `RDPL` and `OPER` (§7), can never be mistaken for a toneme label.

In any particular instance, $T$ stands for exactly one toneme of the language. This can be a contour toneme (like HL in Dom, where it is a single toneme), but never a sequence of tonemes, and never 0, the absence of a toneme. (A numbered variable such as `T1` may be defined to include 0, §2.3.) A definition may give $T$ several possible values, `T = H or L` (§6), but each time the formula applies, $T$ takes only one of them. When $T$ occurs several times in a formula, it stands for the same toneme every time: `(T x) (T y)` describes two neighbouring words with identical tonemes. Tonemes that may differ are distinguished by indices, `T_i`, `T_j`, see §5.2. Different indices allow the tonemes to be different, but do not require it. When they must differ, this is stated in the definitions: `where T_j != T_i`. Like chain variables, $T$ can also be followed by digits, `T1`, `T2`; these are separate variables, just as `T_i` and `T_j` are.

A sequence of tonemes is written with semicolons, as a sequence of concrete tonemes is: `(T_i;T_j x)` is a morpheme with two tonemes, `(T_i;T_j;T_k x)` with three. Without indices, the rule of identity applies: `(T;T x)` is a morpheme with the same toneme twice. There is deliberately no way to say "any number of tonemes": languages normally limit how many tonemes a morpheme can carry. A process that applies to morphemes with two or three tonemes is described by two alternative formulas.

### 2.3 Repeated m-items

Some processes run over a stretch of any length: a toneme spreads over all the toneless words that follow it, or over all the parts of a compound. Such a stretch is written as a single m-item followed by a plus sign, which stands for one or more m-items of that shape: `(0 y)+`. The join written before it also stands between the repetitions, so `-(0 y)+` is `-(0 y)-(0 y)`..., and `␣(0␣y)+` is a sequence of words.

As with `σ+`, the repetitions are independent of each other: in each of them `y` stands for its own segments, and a toneme variable takes its own value. On the surface, the chain variable with a plus sign, `y+`, stands for the segments of all the repetitions, with the joins between them as they were. Here a High toneme spreads over any number of toneless words, up to the next word that has a toneme of its own:

```
(H x) (0 y)+ (T z) > [H x y+] [T z]
```

and here the first toneme of a compound spreads over the whole compound and the tonemes of all the other parts are deleted (the `/ (compound)` is explained in §8):

```
(T_i x)-(T y)+ > [T_i x-y+] / (compound)
```

The surface form refers to a group only through its chain variables; its tonemes cannot be referred to one by one, so a formula with a group describes a process in which they do not surface.

Since $T$ never stands for 0 (§2.2), neither `(T y)+` nor `(0 y)+` covers a stretch where toneless and toned m-items are mixed. For this, a numbered toneme variable may have 0 among its values:

```
(H x) (T1 y)+ (L z) > [H x y+] [L z] where T1 = H or L or 0
```

## 3 Surface form

The surface form reflects the distribution of the tonal spans of the tonemes over the segments. It follows the conventions of the surface annotation line, with a few additions and departures described in this section.

### 3.1 Boundaries and spans

In a formula, a bare hyphen marks a morpheme boundary and says nothing about syllables, whatever terms it stands between: `[H x-y]` means that the two morphemes are in the same tonal span, regardless of where exactly the syllable boundaries are, and `[L x-cv]` does not say whether `cv` is a syllable of its own. To state that the morpheme boundary is also a syllable boundary, write both: `[H x.-y]`. **This departs from the annotation rules**, where a bare hyphen in the surface line means that the boundary falls inside a syllable. In a

formula, the terms usually stand for material whose syllabification does not matter, and the shorter notation covers this general case.

The same holds for two tonal spans written next to each other with no boundary mark between them: `[H x][L y]` says nothing about whether the syllable boundary falls between `x` and `y`. Only a dot (or, for moras, a comma) asserts a boundary. A foot boundary is marked, as in the annotation rules, with a vertical bar: `[H xσ|σ-y]`.

A boundary mark that falls at the edge of a tonal span may stand between the spans, `[H x].[L y]`, before the closing bracket of the previous span, `[H x.][L y]` (the position prescribed in the current version of the annotation rules, §3.7), or after the opening bracket of the following one, `[H x][L .y]`. All of these mean the same, and a combined mark may even be split, as in `[H x].[L -y]`. This document writes the marks between the spans.

The angle brackets that mark the tonal span of a grammatical tone may be used in the surface form, but are not required: `(H σσ)\(L) > [H σ]<[L σ]>`.

One syllable can be shared by two tonemes. As in the annotation rules, the shared segment is then written twice, once in each tonal span. The doubled term must carry an index, and its two occurrences must stand in spans that touch, with nothing between them, not even a boundary mark: `...σ_i][L σ_i...`. Repeated anywhere else, an indexed term is a copy (§7). Here is a falling contour on the last syllable of a high-toned word:

```
(H σσ_i)\(L) > [H σσ_i][L σ_i]
```

The index shows that it is the same syllable that appears twice (§5.2). A shared part of a literal has to be split off in the underlying form, so that it can carry the index:

```
(H "kè""e"_i)\(L) > [H "kè""e"_i][L "e"_i]
```

The notation of the annotation rules where two tonemes share one span, separated by a semicolon, is allowed too: `[H;L σ]`.

### 3.2 Tonemes in the surface form

The provenance of every toneme in the surface form must be specified. It is either:

- a toneme of the underlying form, identified by its position or by an index;
- a toneme derived from an underlying one by a definition in the *where* block (§7);
- or a new toneme, explicitly marked as inserted.

An underlying toneme that does not surface (a deleted toneme) is simply absent from the surface form. For instance, here is a tonal spread of the High toneme to the right in compound words, where the toneme of the second word does not surface:

```
(H x)-(T y) > [H x-y] / (compound (word)-(word))
```

The two words of this compound form a single prosodic word, so they are joined by a hyphen in both forms, as in the annotation rules. In compounds whose parts remain separate prosodic words, the space is kept:

```
(H x) (T y) > [H x y] / (compound (word) (word))
```

**Insertion.** A toneme that has no source in the underlying form is marked with a plus sign. Here is a High toneme inserted between two Low tonemes:

```
(L xσ) (L y) > [L x][+H σ] [L y]
```

The plus sign means "inserted" before a toneme or a segment (§3.3), and "one or more" after σ, μ, a template letter, a class (§2.1), a chain variable on the surface or an m-item (§2.3). In one position both readings are possible: `σ+σ` could be `σ+` followed by `σ`, or `σ` followed by an inserted `+σ`. It is always read the first way, and likewise `c+v` is `c+` followed by `v`, and `y+v` is `y+` followed by `v`. A syllable inserted after a σ is written with the syllable boundary between them, `σ.+σ`. A segment inserted right after a template letter, with no boundary between them, is told apart by an index on the letter: since an index always follows the plus sign of "one or more" (appendix C), `c_i+v` can only be `c_i` followed by an inserted `v`. As every surface index, it must also occur in the underlying form. Here a High tonal morpheme is realized on a vowel inserted after the final consonant of the word:

```
(L xc_i)\(H) > [L x][H c_i+v]
```

**Copy.** When the underlying toneme is copied (the surface realization is interpreted as two separate spans), the index of the toneme is repeated:

```
(H_i x) (0 y) > [H_i x] [H_i y]
```

**Fusion.** When two underlying tonemes give one toneme on the surface, the surface toneme carries the indices of both:

```
(L_i cv)-(L_j v) > [L_ij cv-v]
```

A fused toneme is one toneme on the surface and counts as one. In fact, fusion differs from spread or deletion only in what we know about the process. When two tonemes of the same nature meet and only one of them surfaces, we write a spread or a deletion whenever there is some evidence of which toneme won, or a convention that decides it (grammatical tones take precedence over lexical ones). Fusion is written when there is no reason to prefer one of the tonemes as the winner.

**Realization.** A surface span can carry the label of an allotone instead of a toneme, with the index of the toneme it realizes. Here an L is realized as its rising allotone before a Low word:

```
(L_i x) (L y) > [LH_i x] [L y]
```

The label differs from the underlying one, but the index shows that it is the same toneme. Two fused tonemes can likewise be realized with a label of a third value, H and L together giving M:

```
(H_i x)\(L_j) > [M_ij x]
```

A label used this way must be a declared toneme or allotone of the language. See also §9 on how to use formulas to describe allotony.

A realization label stands for one toneme (or allotone) as a whole. Our model does not split a contour toneme into its levels, nor build a contour toneme out of two level ones, so there is no notation for "the first part of toneme $i$". How the tonal gestures are distributed inside a span is a phonetic detail that the formulas abstract away from.

**Downstep.** A Low toneme realized as a downstep of the following High is written, as in the annotation rules, as a span with an exclamation mark and no segments:

```
(H x)\(L) (H y) > [H x] [L !][H y]
```

In a language where downstep is a toneme of its own or is defined as an allotone, a `[D]` label can be used in this case with the same meaning. Otherwise, a bare toneme with no segments at all, like `[L]`, is not allowed in the surface form. Nor is a span without a toneme, like `[0 x]`: segments that carry no toneme on the surface are written outside any span (§4).

### 3.3 Segments in the surface form

The segments may change too. A segment deleted by a segmental rule is omitted from the surface form, just like a deleted toneme. The surface form may also contain segments that have no counterpart in the underlying form, for instance a vowel inserted by a segmental rule. Like an inserted toneme, an inserted segment is marked with a plus sign. This keeps an epenthetic vowel apart from a shared one. Here the Low toneme docks onto a vowel inserted after the root:

```
(H cv)\(L) > [H cv].[L +v]
```

while here it shares the root vowel:

```
(H cv_i)\(L) > [H cv_i][L v_i]
```

The surface form reuses the terms of the underlying form: a σ remains a σ, and a chain variable remains the same chain variable. The surface cannot be more specific than the underlying form, so `(H σσ) > [H cv].[L cv]` is not a valid formula; when the shape matters, it is stated in the underlying form. A segment that changes, on the contrary, may change its term, and an index then shows which segment it continues. Here a vowel is lengthened:

```
(H cv_i)\(L) > [H cv:_i][L v:_i]
```

The stress mark (*) and the clitic boundary (=) may appear on the surface as well, as in the surface annotation line.

## 4 Edges and pauses

The edges of prosodic units are shown with the boundary markers of the surface annotation line: a space is the edge of a prosodic word, and curly braces enclose a tonal phrase. A pause or the end of an utterance is marked with a sharp sign #, in both forms. For instance, a Low toneme that spreads over the following High-toned word before a pause:

```
(L x) (H y) # > [L x y] #
```

The beginning and the end of the form are not edges: a formula is open on both sides. `(H x)-(0 y) > [H x-y]` applies to a root and a suffix wherever they stand in a word, whether other prefixes or suffixes precede or follow them. When the process does depend on the edge of a word, this is stated in the constraints (§8).

Combined with σ, the edges allow positions to be counted from the end of a word. Here is a High toneme spreading up to the penultimate syllable of the word; the constraint says that the suffix is the last morpheme of the word, while any number of morphemes may precede the root (§8):

```
(H x)-(0 σ+σ) > [H x-σ+]σ / (word ...-()-())
```

Curly braces appear only in the surface form, since a tonal phrase is a prosodic unit and the underlying form has none. They say the same as a `prosodic_phrase` scope over the m-items they enclose (§8). The pause, on the other hand, is part of both forms; in the constraints it may be repeated, but need not be.

As in the surface annotation line, segments outside any tonal span are written without brackets.

**Boundary tones.** An intonational boundary tone is written as an m-item with a tonal part only. Its label is built from the same tonal levels as a toneme label, followed by `%` for a boundary tone at the right edge, `H%`, `L%`, or preceded by it at the left edge, `%H`. Like any tonal morpheme (§2.2), it follows the m-item on which it is realized, joined with a backslash: a

right boundary tone stands right before a pause or at the end of a tonal phrase, a left one after the first m-item that follows a pause or begins a phrase. Here the H% of a question shares the last syllable of the utterance:

```
(L xσ_i)\(H%) # > [L xσ_i][H% σ_i] #
```

Since a boundary tone is an m-item, it can be constrained by its meaning like any other morpheme, `/ ()\(Q)`, and it interacts with the tonemes of the language through the same notation for deletion, fusion, sharing and indices.

# 5 Correspondence between underlying and surface forms

The correspondence of the tonemes and the segments between the underlying and the surface forms is in most cases clear from their names. But the same term may occur more than once: several $T$, several σ, several CV letters of the same kind. For these cases, two notational conventions are provided.

## 5.1 Implicit order-based correspondence

Repeated identical terms are matched left to right: the first σ of the surface form corresponds to the first σ of the underlying form, and so on. This works when both forms contain exactly the same terms, and in the same order. Then they can be left without the indices:

```
(T cvcv)\(L) > [T cv].[L cv]
```

A segment shared by two spans (§3.1) always needs an explicit index: it appears twice in the surface form but only once in the underlying form, so the order-based correspondence fails. Appendix B sums up what a repeated term means for each kind of term.

## 5.2 Index notation

When the order of the identical terms changes, or when some of them do not surface and it matters which ones remain, the correspondence is made explicit by adding lowercase letter indices with an underscore, `T_i`, `T_j`, `σ_i`, `c_i`. For example, if two arbitrary tonemes participate in a process, and it is the first one that remains:

```
(T_i x)-(T_j y) > [T_i x-y]
```

An index is a single letter, optionally followed by digits: $i$, $j$, $i1$, $i2$. It belongs to the term right before it: in `cvc_i` only the last consonant is indexed. Four letters serve as indices without braces: $i$, $j$, $k$ and $m$. Inside braces (see below), any lowercase letter may be used, `_{a}`, though the four letters with digits should be enough for most formulas.

Indices for the tonemes and for the segments are independent, so the same index letters may be used simultaneously for tonemes and segments in the same formula.

A term can carry two indices when it continues two underlying terms, as in fusion: `[L_ij x]`. Several indices are written one after another, `_ij`, `_i1i2`, `_{ab}`; this is never ambiguous, since every index begins with a letter and the digits belong to the letter before them. Braces delimit the indices explicitly: `c_{ij}` is `c` with the indices $i$ and $j$, and `c_{i}j` is `c_i` followed by `j`. Without braces, every index letter after the underscore is read as an index, so the short form `_ij` means the same as `_{ij}`. The short form is perfectly fine wherever it cannot be misread, that is, when the index is followed by a space, a bracket or a boundary mark, as in `[L_ij x]`, `σσ_i` or `-c_j-v_i`; braces are not needed there.

Wherever the end of an index might be misread, write the braces. An unbraced index directly followed by a letter, like `c_ijv`, will result in a warning from the parser, even when none of the index letters is a predefined template letter or a chain variable. It should be written `c_{ij}v`. Braces are obligatory when a formula defines an index letter as a class (§2.1): if $j$ is a class, the class after an indexed `c` is written `c_{i}j`, while `c_ij` is still read as two indices, with a warning.

Infixes and transfixes change the order of the segments. Following the annotation rules, they are written in the underlying form after the morpheme they interrupt, but on the surface they appear inside it. When the segments are written with chain variables, their different names are enough to match them, and no indices are needed. Here is the Eton infix *-g-*:

```
(H;L xy)-(L^ ^z) > [H;L x].-[L z-y]
```

With CV templates, the letters are repeated, and the order of the consonants is not the same in both forms. The following formula is therefore **incorrect**:

```
(H;L cvcv)-(L^ ^c) > [H;L cvc].-[L c-v]
```

Here the indices become obligatory. The formula should be written with indices on the segments that change their position:

```
(H;L cvcv_i)-(L^ ^c_j) > [H;L cvc].-[L c_j-v_i]
```

# 6 Definitions

Definitions set the values that a variable can take in the given formula. A definition lists the permitted values separated by *or*. For instance, if a language has three tonemes in the inventory, /H/, /L/, and /M/, but only /H/ and /L/ can participate in a given process, e.g., a spread, a corresponding definition can be added:

```
(T x)-(0 y) > [T x-y] where T = H or L
```

When a formula uses indexed tonemes, a definition of a bare $T$ applies to every one of them, while a definition of `T_i` applies to that one only.

The same form can be used to define syllable structure constraints:

```
(T x)-(0 y) > [T x-y] where y = cvr or cvn
```

A CV letter can be narrowed down to particular segments, written as literals. Here is a spread of a Low toneme onto a syllable beginning with a voiced obstruent (a depressor consonant):

```
(L x)-(0 cv) > [L x-cv] where c = "b" or "d" or "g"
```

In these examples the definition narrows a variable down: its values are a subset of what the letter stands for anyway. This is the usual case, but not a requirement. A definition replaces the meaning of the letter for the given formula, whatever that meaning was, so a predefined letter can also be given values outside its class. Here `d` stands for prenasalized consonants:

```
(L x)-(0 dv) > [L x-dv] where d = "mb" or "nd" or "ŋg"
```

The same formula with a new name for the class, `c1`, is equivalent:

```
(L x)-(0 c1v) > [L x-c1v] where c1 = "mb" or "nd" or "ŋg"
```

If the same letter occurs several times in a formula, an index restricts only one of them: `where c_i = "b" or "d"`. A definition can also exclude values, with *not*. For instance, any consonant except a prenasalized one:

```
where c = not ( "mb" or "nd" or "ŋg" )
```

Parentheses are used to explicitly mark the logical operation precedence. The same condition can be written without parentheses using *nor*, which continues a *not*: *not* A *nor* B excludes both A and B.

```
where c = not "mb" nor "nd" nor "ŋg"
```

*Not* excludes values from the predefined meaning of the letter: `c = not "mb"` is any consonant except *mb*. For a name that has no predefined meaning, like `c1`, *not* excludes values from any phoneme of the language. Together with the plus sign (§2.1), this describes a stretch that does not contain some class of segments. Here a High toneme spreads onto a toneless word, but not across a depressor consonant (a voiced obstruent):

```
(H x) (0 c1+) > [H x c1+] where c1 = not d
```

Several definitions are separated by commas: `where T = H or L, y = cvr or cvn`.

A value is written with the same terms as the segmental part of an m-item: template letters, literals, chain variables, σ and μ, and names defined elsewhere in the same block (`y = c1v, c1 = "mb" or "nd"`). A name must not be defined, directly or through other names, by itself. Since a value stands for underlying material, which has no syllabification, it contains no dots and no commas. A pair of values joined with a colon (§7) is used for tonemes only: for segments the colon marks length.

# 7 Tonal operations

Some processes turn one toneme into another by a regular pattern: polarization, shifting tone levels up or down, and various morphological operations (OPER in the annotation rules). Such a pattern is defined by listing the pairs of values that two toneme variables can take together, separated by a colon. For instance, the Nominative in Kipsigis inverts the tonemes of the noun:

```
(T_i x)\(OPER) > [T_j x] where T_i:T_j = H:L or L:H
```

This reads: $T_j$ is L when $T_i$ is H, and H when $T_i$ is L. An operation shifting each toneme one step up, in a language with three levels:

```
(T_i x)\(OPER) > [T_j x] where T_i:T_j = L:M or M:H
```

H, which has nowhere to go, is not listed, so this formula does not apply to H-toned words. A toneme defined this way is not marked as inserted: the definition shows where it comes from.

For the common case of a binary system with H and L only, a tilde before a toneme may be used as a shorthand for "the opposite toneme". Here a toneless suffix takes the toneme opposite to that of the root:

```
(T x)-(0 y) > [T x].-[~T y]
```

This tilde cannot be confused with the tilde that joins a reduplicant: that one stands between m-items or between segments, never in the tonal part.

A subtractive tonal morpheme (SBTR in the annotation rules) deletes a toneme of the morpheme it attaches to. It can be written with a `SBTR` keyword in its tonal part. Here a suffix erases the toneme of the root:

```
(T x)-(SBTR y) > x-y / (root)-(suffix)
```

A full reduplication (RDPL) copies the segments of its base, so in the surface form the same variable $x$ is repeated:

```
(T x)~(RDPL) > [T x~x]
```

As in the annotation rules, the keywords may stand in either part of an m-item. When the reduplicant has a toneme of its own, `RDPL` takes the place of the segments:

```
(H RDPL)~(L x) > [H x]~[L x]
```

Likewise, `OPER` may stand in the tonal part, alone, `\(OPER)`, or before segments of its own, `(OPER^ ^z)`, or in the segmental part, after a toneme, `(H OPER)`.

In general, a term repeated in the surface form, whether a variable or an indexed toneme or segment, means that the same material is realized twice. The one exception is an indexed segment repeated in two spans that touch: that is a single unit shared by both spans (§3.1).

## 8 Morphosyntactic constraints

The constraints part of the rule follows the surface form after a slash. It is an optional part of the formula.

The simplest form of the constraint is a sequence of parentheses exactly corresponding to the sequence of the m-items in the underlying form. Each parenthesis specifies a morphological or morphosyntactic class for a corresponding m-item. If a particular m-item has no specific constraint, an empty parenthesis should be used: `()`.

Two kinds of labels can be used in the parentheses. Lowercase labels are classes from a fixed vocabulary:

- Parts of speech: noun, verb, adjective, pronoun, numeral, adverb, auxiliary, adposition, particle, determiner, ideophone.
- Morphemes: root, stem, suffix, prefix, clitic.
- Word: a morphosyntactic word of any internal structure. As the class of a single m-item, it says that the m-item makes up a whole word; as a scope, it covers all the m-items of one word.
- Scope categories (see below): the prosodic and syntactic domains of the rule form, compound, word, and the parts of speech.

The names of the domains correspond one-to-one to the domains offered in the rule editor in the ThotDB, so that the scope of a formula can be checked against the domains selected for the rule. Since a space in the constraints is already used as a separator, each name is written as a single word, with underscores in place of spaces:

| *In the rule form* | *In the formula* |
|---|---|
| syllable | `syllable` |
| foot | `foot` |
| prosodic word | `prosodic_word` |
| prosodic phrase | `prosodic_phrase` |
| head noun & dependent noun | `noun_noun` |
| noun & adjective | `noun_adjective` |
| noun & numeral | `noun_numeral` |
| noun/noun phrase & adposition | `noun_adposition` |
| subject & verbal predicate | `subject_predicate` |
| verbal predicate & direct object | `predicate_object` |
| verbal predicate & adverbials | `predicate_adverbial` |
| determiner or demonstrative & noun | `determiner_noun` |
| possessor & possessed noun | `possessor_possessed` |
| auxiliary & verb | `auxiliary_verb` |
| verb & subject or object marker | `verb_person_marker` |

Uppercase labels are interpreted as glosses (following the Leipzig Glossing Rules): `(PFV.AFF)`, `(CL1)`, `(3SG)`. They identify particular grammatical morphemes. A gloss is written with uppercase letters and digits, and may contain dots and the sign `>`, as in `(1SG>3PL)`. A lexical gloss is written in double quotes: `("child")` is the morpheme glossed *child*. For glosses, an exact match is assumed: `(PFV)` does not match a morpheme glossed *PFV.AFF*.

A comma inside a parenthesis means "or": `(CL1,CL9)` stands for a m-item that is either a class 1 or class 9 marker. A reserved keyword *not* excludes a class or a morpheme: `(not 3SG)`. *Not* applies to the whole list that follows it: `(not CL1,CL9)` is any morpheme other than CL1 and CL9.

A name with the sign `@` is a named class of morphemes: `(@yang)`, `(@class3)`. The name is only a label for a class that is documented outside the formula, in the description of the rule, so the class can be as large as needed. A small class may be listed in the definitions instead, as alternatives separated by *or*:

```
where @pfv = PFV.AFF or PFV.NEG
where @dan3 = "child" or "go" or "eat"
```

The `@` sign keeps such a name from being confused with a gloss.

Several conditions separated by a space must all hold: `(verb @class3)` is a verb of class 3, `(noun not CL1,CL9)` a noun of any class other than 1 and 9. The comma is of higher priority operation than the space, therefore *not* applies up to the next space.

For example, a spread that only happens from a prefix to a toneless root:

```
(M x)-(0 y) > [M x-y] / (prefix)-(root)
```

The same spread, triggered only by the prefixes of classes 1 and 9:

```
(M x)-(0 y) > [M x-y] / (CL1,CL9)-(root)
```

A constraint can also define a category for the whole scope of the process, be it a word or a construction. The scope is written as a further parenthesis around the m-items it covers, with the category immediately after the opening bracket: `(compound (word)-(word))`.

A scope implies that the m-items it encloses are the whole unit, no material could be added on either edge. For instance, `(word (root)-(suffix))` is a word made of exactly a root and a suffix. When the unit may contain additional m-items, not represented in the formula, they are shown at the edge or both edges of the scope with an ellipsis, `...`. Ellipsis covers any number of m-items. An example of the spread from the prefix above, but scoped to only nouns, which may or may not have suffixes:

```
(M x)-(0 y) > [M x-y] / (noun (prefix)-(root)-...)
```

The category of a scope may be a list separated by commas, which means "or", as it does in an m-item. The same spread in nouns and adjectives:

```
(M x)-(0 y) > [M x-y] / (noun,adjective (prefix)-(root)-...)
```

Each name in the list must be a scope category. A named class (`@nominal`), a gloss or *not* cannot be the category of a scope, even when the class is defined in the formula as a list of scope categories: the list is written out, or the process is described by alternative formulas.

When the scope covers all the m-items of a formula, the constraints can be shortened to the category alone:

```
(H x)-(0 y) > [H x-y] / (word)
(H x)-(0 y) > [H x-y] / (word ()-())
```

These two formulas are synonymous. Several conditions in the short form are scopes nested over the same m-items, outermost first, and a comma list is a scope with several categories. A spread in compounds that are nouns, or nouns and adjectives, can be written either way:

```
(T_i x) (T_j y) > [T_i x y] / (compound noun)
(T_i x) (T_j y) > [T_i x y] / (compound (noun () ()))

(T_i x) (T_j y) > [T_i x y] / (compound noun,adjective)
(T_i x) (T_j y) > [T_i x y] / (compound (noun,adjective () ()))
```

Only scope categories can be shortened so. A gloss, a named class, a morpheme class such as *root*, or *not* describes a single m-item, and with any of them the parenthesis is one m-item again: `(compound @nominal)` over two m-items is an error, which the editor reports with the scope written out. The short form applies when the underlying form has more than one m-item. With one m-item, a single label is simply the class of that m-item, which for *word* and the prosodic domains amounts to the same thing.

A scope can equally name a syntactic domain. Here a High toneme spreads from a noun onto the following adjective:

```
(H x) (0 y) > [H x y] / (noun_adjective (noun) (adjective))
```

The name of a syntactic domain says which construction is meant, not the order of its parts:

```
(noun_adjective (adjective) (noun))
(noun_adjective (noun) (adjective))
```

A scope may cover only some of the m-items, several scopes may stand side by side, and scopes may be nested:

```
() (noun (prefix)-(root)-...)
(prosodic_phrase (noun (root)-(suffix)) ())
```

Note that *word* in the constraints is a word in the morphosyntactic sense, whatever its internal structure: `(word (root)-(suffix))`. The prosodic word is called *prosodic_word*.

The constraints follow the m-items of the underlying form one by one. A pause (#) may be repeated in the constraints, but it is optional, since it should be present in the underlying and surface forms anyway. A repeated m-item (§2.3) is matched by a parenthesis with a plus sign, `()+` or `(suffix)+`.

There is no "elsewhere" condition. A formula states the conditions under which its process applies. If a process applies everywhere except in some context, that context is excluded with *not*.

## 9 Allotony

The formula syntax can also be used to formally specify the conditions for the selection of a specific allotone of a toneme. That does not imply that allotony is a process that transforms some underlying form of a toneme into its surface form. It is not always the case that we can easily select which allotone is the main one, so selection of the underlying form of a toneme will be arbitrary in such cases. In general, we do not subscribe to the idea that a toneme should have a single underlying form. Thus in case of allotony, the formula syntax is just a conventional way to formalize the allotone realization conditions (using the syllable templates, or morpho-syntactic conditions).

Free variation of the allotones poses a specific problem for the model of a transformative process implied by the formula structure. We suggest the following notation for such cases:

```
(xH_i x) > [T_i x] where T_i = xH or xHM
```

This formula states that the toneme identified as `xH` can be realized on the surface both as `xH` and `xHM`. Free variation of the realizations is implied by the inclusion of both allotones in the class `T_i` along with the lack of any constraints. As everywhere (§2.2), `T_i` takes one of its values each time the formula applies; the index ties it to the underlying `xH`, whose realizations the values are. One should regard the underlying form of a toneme in such cases just as a conventional label, not a theoretical statement.

## A Symbols at a glance

The table lists every symbol used in the formulas, with the part of the formula where it appears and what it means there. Most of them mean the same as in the annotation rules; the differences are discussed in the sections given in the last column.

| Symbol | Where | Meaning | § |
|---|---|---|---|
| *Parts of the formula* | | | |
| `>` | between forms | underlying form becomes surface form | 1 |
| `/` | after the surface form | introduces the constraints | 1, 8 |

| Symbol | Where | Meaning | § |
|---|---|---|---|
| `where` | at the end | introduces the definitions | 1, 6 |
| *Brackets* | | | |
| `( )` | underlying form | m-item: tonal part, then segmental part | 2 |
| | constraints | class or gloss of an m-item; with a category name (or a comma list of them) first, a scope, which pins both edges | 8 |
| `[ ]` | surface form | tonal span | 3 |
| `< >` | surface form | tonal span of a grammatical tone (optional) | 3.1 |
| `{ }` | surface form only | tonal phrase; same as a `prosodic_phrase` scope | 4 |
| `" "` | segmental part | literal segments, written as they are | 2.1 |
| | constraints | lexical gloss | 8 |
| *Boundaries and edges* | | | |
| space | both forms | boundary of a prosodic word | 2, 4 |
| `-` | between m-items or segments | morpheme boundary; on the surface, says nothing about syllables | 2, 3.1 |
| `=` | between m-items; inside surface spans | clitic boundary | 2, 3.3 |
| `~` | between m-items or segments | reduplicant and its base | 2, 7 |
| `\` | between m-items | tonal morpheme or operation without segments of its own | 2 |
| `.` | surface form | syllable boundary | 3.1 |
| | m-item, before `^` | counts syllables to the alignment point | 2 |
| `,` | surface form | mora boundary | 3.1 |
| | m-item, before `^` | counts moras to the alignment point | 2 |
| `\|` | surface form | foot boundary | 3.1 |
| `#` | both forms; optional in constraints | pause or end of utterance | 4 |
| `^` | m-item | alignment point | 2 |
| `H%`, `%H` | m-item, surface form | intonational boundary tone at the right, left edge | 4 |
| * | segmental part, surface form | stress, before the stressed syllable | 2 |
| *Tonemes* | | | |
| `H`, `L`… | tonal part | toneme labels of the language | 2.2 |
| `0` | tonal part | no toneme | 2.2 |
| | definitions | a value of a numbered toneme variable, `T1 = H or 0` | 2.3 |
| `T` | tonal part | any one toneme; the same toneme wherever it repeats | 2.2 |
| `T1`, `T2`… | tonal part | further toneme variables | 2.2 |
| `T_i;T_j` | tonal part | a sequence of two tonemes (`T;T`: the same toneme twice) | 2.2 |

| Symbol | Where | Meaning | § |
|---|---|---|---|
| `;` | tonal part | separates several tonemes of one m-item or one span | 3.1 |
| `+H` | surface form | inserted toneme | 3.2 |
| `~T` | surface form | the opposite toneme (binary H/L systems only) | 7 |
| `[L !]` | surface form | Low toneme realized as a downstep | 3.2 |
| `SBTR` | tonal part of an m-item | subtractive morpheme: deletes a toneme | 7 |
| `RDPL` | either part of an m-item | reduplication operation | 7 |
| `OPER` | either part of an m-item | tonal operation | 7 |
| *Segments* | | | |
| `x y z w` | segmental part | chain variables: any segments, at least one | 2.1 |
| `x1`, `c1`… | segmental part | a letter with digits: a further chain variable or custom class | 2.1 |
| `σ`, `μ` | segmental part | exactly one syllable, one mora | 2.1 |
| `σ+`, `μ+` | segmental part | one or more syllables, moras | 2.1 |
| `c v n`… | segmental part | CV template letters, see the table in §2.1 | 2.1 |
| `c+`, `c1+` | segmental part | one or more segments of the class | 2.1 |
| `(0 y)+` | underlying form | one or more m-items of this shape | 2.3 |
| `y+` | surface form | the segments of all the repetitions of `(0 y)+` | 2.3 |
| `:` | CV template | long vowel, `v:`; may correspond to an indexed `v_i` | 2.1, 3.3 |
| `+v` | surface form | inserted segment | 3.3 |
| *Correspondence* | | | |
| `_i` | after a toneme or segment | index (`i j k m`, with digits: `i1`): the same index means the same toneme or segment | 5.2 |
| `_ij` | surface form | a toneme continuing two underlying ones (fusion) | 3.2 |
| `_{ij}` | after a toneme or segment | indices delimited explicitly, any letter allowed (`_{ab}`); advised whenever a letter follows, required when an index letter is a class (`c_{i}j`) | 5.2 |
| *Definitions and constraints* | | | |
| `=` | definitions | the values of a variable in this formula (narrows or redefines it) | 6 |
| `or` | definitions | separates the permitted values | 6 |
| `not` | definitions, constraints | excludes values, classes or morphemes | 6, 8 |
| `nor` | definitions | after *not*: excludes a further value | 6 |
| `!=` | definitions | two variables must differ | 2.2 |
| `,` | definitions | separates several definitions | 6 |
| | constraints | “or” inside a parenthesis, and in the category of a scope | 8 |
| space | constraints | “and” inside a parenthesis | 8 |

| Symbol | Where | Meaning | § |
|---|---|---|---|
| `...` | constraints, at the edge of a scope | any number of m-items not shown | 8 |
| `@name` | constraints, definitions | named class of morphemes | 8 |
| `:` | definitions | pairs of values of two variables, `T_i:T_j = H:L` | 7 |
| lowercase | constraints | class from the fixed vocabulary | 8 |
| UPPERCASE | constraints | gloss of a grammatical morpheme | 8 |
| `()+` | constraints | matches a repeated m-item | 8 |

# B Repeated terms

What it means when the same term occurs more than once in a formula depends on the kind of term, and on whether the repetition is in the underlying or in the surface form.

| Term | Repeated in the underlying form | Repeated in the surface form |
|---|---|---|
| `T` | the same toneme | the same toneme (a copy needs an index) |
| chain variable `x` | the same string of segments | a copy of the same segments |
| template letter `c` | independent segments of the same class | independent segments, matched to the underlying ones by order |
| `σ`, `μ` | independent units | independent units, matched by order |
| literal `"k"` | equal strings | equal strings |
| indexed term `v_i` | the same term | a copy; in two spans that touch (`][`), one shared segment |
| m-item with `+` | independent m-items of the same shape | — (only `y+` refers to it) |

# C Order of the marks on a term

A term may carry several marks. They are written in a fixed order:

1. before the term: `+` (inserted), then `*` (stress) or `~` (the opposite toneme);
2. the term itself, with its digits: `c1`, `T2`;
3. `:` (length) or `+` (one or more: `σ+`, `c+`, `y+`);
4. the index: `_i`, `_ij`, `_{ij}`;
5. the alignment point `^`.

So the correct forms are `L_i^`, `σ+_i`, `v:_i` and `+*σ`, not `L^_i`, `σ_i+` or `v_i:`. The plus sign of a repeated m-item follows its closing bracket: `(0 y)+`.