# **Sound changes**

**Sound change rules** (also known as **phonological rules**) can be applied to all words using the **Sound Change** field. This notation can also in affix rules, **Custom Spelling**, and **Illegal combinations** field.

A basic rule like **a > o** means **change every a to o**. Try it yourself:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| alabama | a > o | **olobomo** |

You can also specify if the change should only happen in a particular environment: **a > o / \_b** means **change every a to o if it comes before a b**. The **/** means **in the environment of** and **\_** signifies where the **a** is.

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| alabama | a > o / \_b | **alobama** |

Be aware that letters with diaciritcs are treated as different phonemes to letters without. Therefore, **t > p** does not change an aspirated **tʰ**:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| utah tʰexas | t > p | **upah tʰexas** |

## **Word boundaries**

**#** signifies a word boundary. Notice how the following pattern only changes the first **a**.

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| alabama | a > o / #\_ | **olabama** |

You must use **##** for rules that go across word boundaries:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| alaska iowa ohio | a > u / \_##i | **alasku iowa ohio** |

Similarly, **a > ##** changes every **a** to a space.

## **Classes and sets**

**C** matches any consonant and **V** matches any vowel:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| alabama | V > o / C\_ | **alobomo** |

Additionally, **X** matches any phoneme at all, with or without diacritics on it; **D** macthes any IPA *letter* (does not capture diacritics); and **ᴰ** (superscript D) captures any diacritic symbol.

Capture a custom set of phonemes inside curly brackets: **{b,m}** matches **b** or **m**.

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| alabama | {b,m} > p | **alapapa** |
| alabama | a > o / \_{b,m} | **aloboma** |

To capture other classes of phonemes with common features, such as stops, nasals, etc., you should use [distinctive feature](https://www.vulgarlang.com/sound-changes/#distinctive-features) notation.

Note: Curly brackets cannot be used in the replacement side of the **>** symbol. Instead, bar symbol **|** creates a random choice between replacement options: **b > m | n | b** changes **b** to **m** or **n** or **b**. All three options are equally probable, but can be weighted with the **\*** symbol: **b > m\*3 | n\*2 | b** is the same as **b > m | m | m | n | n | b**.

## **Optional patterns**

Round brackets makes a pattern optional:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| alaksa kansas | a > o / \_(C)# | **alakso kansos** |

Adding a star **\*** after the brackets means *zero or more* instances of the pattern. Brackets can be ommited when the pattern is a single symbol: **C\*** and **(C)\*** both mean any number of consonants in a row, however **(VC)\*** is needed to match a succession of VC patterns in a row.

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| alaksa maryland | a > o / \_C\*# | **alakso marylond** |

*One or more* of a pattern can be achieved by making the star optional **C(\*)**, or simply writing the pattern twice like **CC\***.

## **Deletion**

To delete something, simply change to literally nothing:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| alabama | C > | **aaaa** |

You can also write a null symbol, **C > ∅**. Note that the null symbol (U+2205) should not be confused the Scandinavian letter **Ø** (U+00D8).

## **Insertion (infixing)**

Similar to deletion, changing *from* nothing (or null symbol **∅**) will insert a phoneme in the environment you specify:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| mtana vermt | > on / m\_ | **montana vermont** |

Note that this effectively how you do [infxes](https://en.wikipedia.org/wiki/Infix) in Vulgarlang. In linguistic writing, the convention is to write infixes with hyphens either side of it, e.g. **-on-**. However this does not tell us exactly where in the word the infix goes. A linguistic paper would given examples or describe where the infix goes, however in Vulgarlang we instead must insert it as a sound change at a particular environment.

## **Syllables**

**$** signifies a syllable boundary, and sigma **σ** matches a whole syllable. The difference between the two is that syllable boundary also includes word boundaries. So a rule such as **i > e / \_$** would match **i** at the end of word, where as **i > e / \_σ** wouldn't, because it's looking for a whole syllable after it.

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| hawaii | i > e / \_$ | **hawaee** |

Be wary of syllable boundaries because Vulgarlang may split the syllables where you don't agree they should be split! Since there's generally no agreement where syllables should be split. Is "syllable" si-la-bəl or sil-a-bəl? You can test exactly how the app splits syllables on this page by changing the syllable boundary to something like a full stop **$ > .**

## **Affix boundaries**

**+** signifies an affix boundary, **a > e / \_+** changes a before an affix boundary only. When an affix is added to a word in the Vulgarlang app, it adds a [zero-width](https://en.wikipedia.org/wiki/Zero_width) character at the affix boundary, which tells the sound change where the boundary is. In the below example, a zero-width character has been manually added bewteen **california + n**:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| california⁠n | a > e / \_+ | **californie⁠n** |

As noted in the **Insertion (infixing)** section, infixes in Vulgarlang have to be added with sound changes. By default, an insertion sound change does *not* add these affix boundaries either side of the insertion. If you want to make other sound changes that are sensitive to this infix boundary, you must add the **+** symbols into the insertion rule itself: **> +on+ / m\_**

## **Exceptions to rules**

The **!** symbol signifies an exception to a rule. The following changes every **a** to **o** except before **m**:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| alabama | a > o / !\_m | **olobamo** |

Exceptions can also be added to the end of a normal rule, eg: **a > o / \_C !\_m** means **a changes o before any consonant, except m**.

## **Distinctive features**

Square brackets are used to specify the presence (+) or absence (-) of distinctive features of a phoneme. The following rule nasalizes a vowel before another nasal:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| kansas | V > [+nasal] / \_[+nasal] | **kãnsas** |

**[+nasal]** matches both nasal vowels and nasal consonants. But you could specify that you only want it to affect nasal consonants with **[C +nasal]**:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| kãnsas | [C +nasal] > b | **kãbsas** |

Various features can be used to narrow down the match. The following rule deletes voiceless stops, which in this case is only **k**:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| nebraska | [+stop -voice] > [+alveolar] | **nebrasta** |

Here is a complete list:

| **Feature** | **Meaning** |
| --- | --- |
| [+affricate] | affricate consonants |
| [+alveolar] | alveolar consonants |
| [+approx] | approximants |
| [+aspirated] | aspirated consonants |
| [+back] | back vowels |
| [+click] | click consonants |
| C or [+consonant] | consonants |
| [+cons] | consonantals (different to consonants - see [Wikipedia](https://en.wikipedia.org/wiki/Distinctive_feature#Major_class)) |
| [+cont] | continuants |
| [+dorsal] | dorsals |
| [+fricative] | fricative consonants |
| [+front] | front vowels |
| [+glottal] | glottal consonants |
| [+high] | high vowels |
| [+implosive] | implosive consonants |
| [+labial] | labial consonants and round vowels |
|  |  |
| [+labialized] | labialized consonants |
| [+laryngeal] | laryngeal consonants |
| [+lat] | lateral consonants |
| [+liquid] | liquid consonants |
| [+long] | long vowels or long (geminate) consonants |
| [+low] | low vowels |
| [+nasal] | nasal consonants or nasal vowels |
| [C +nasal] | nasal consonants |
| [V +nasal] | nasal vowels |
| [+palatal] | palatal consonants |
| [+palatalized] | palatalized consonants |
| [+pharyngealized] | pharyngealized phonemes |
| [+retroflex] | retroflex consonants |
| [+round] | round vowels |
| [+son] | sonorants |
| [+stop] | stop consonants (a.k.a. plosives) |
| [V +stress] | stressed vowels |
| [+tap] | tap consonants |
| [+trill] | trill consonants |
| [+uvular] | uvular consonants |
| [±+velar] | velar consonants |
| [±+velarized] | velarized consonants |
| [+voice] | voiced consonant or vowel (unless specifically a voiceless vowel) |
| [C +voice] | voiced consonant |
| [V +voice] | voiced vowels |
| V or [+vowel] | vowels |

## **Assimilation**

Assimilation is the process where a phoneme takes on a distinctive feature of another phoneme. An example of this in English can be seen in the prefix **in-** which makes the opposite meaning of a word, eg: **tolerant** becomes **intolerant**. However if the word begins with **m** or **p** notice how the prefix becomes **im-** as in **immovable** and **impossible**. This is because the nasal consonant **n** assimilates to the "place of articulation" of the following phoneme; **m** and **p** are labial (pronounced with a closure of the lips) so **n** (alveolar nasal) becomes **m** (labial nasal). Meanwhile the place of articulation of **n** and **t** are both alveolar. And although it is not reflected in the spelling, the word **incomplete** is actually pronoucned with a ŋ [iŋkomplit]; **n** assimilates to the velar place of **k**.

It would be possible to capture this process with two separate rules: **n > [+labial] / \_[+labial]** and **n > [+velar] / \_[+velar]**. However we can write this as a single rule **n > [@place] / \_[@place]**:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| helsinki istanbul | n > [@place] / \_[@place] | **helsiŋki istambul** |

This is currently an experimental feature, and works for **@place**, **@manner** and **@voice** for consonants, and **@height** and **@backness** for vowels.

## **Stress**

You can use the feature **[V ±stress]**. For example **[V -stress] > ə** mimics English's tendency to turn unstressed vowels into **ə**. You can also target an individual vowel: **[ɪ +stress] > i**

To change the stress location of a word, you must target the location of the vowel, eg **V > [+stress] / \_C\*#** stresses the last vowel in a word: **C\*#** meaning before any number of consonants before the word boundary.

Additionally, **[V ±stress]** does not work for *spelling* rules. (The generator can't tell where syllables begin and end once it's dealing with your unique spelling system, and not pure IPA.) Instead you must check **Make spelling rules sensitive to stress symbol** option and make rules that account for the **ˈ** stress symbol. The following rule looks for the stress mark and any optional consonants before the vowel. A separate rule will be needed to remove the stress symbols:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| ˈkansas neˈbraska | a > á / ˈC\*\_ | **ˈkánsas neˈbráska** |

## **Same phoneme in a set**

There may be scenarios where you want to match two of the same phonemes from a class. To do this, use the a matching subscript number (**₁₂₃₄₅₆₇₈₉**) after the uppercase class. For example, **C₁C₁** means two of the same consonants in a row:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| massachusetts | V > o / \_C₁C₁ | **mossachusotts** |

Different numbers can signify that they are different phonemes. The following pattern matches two consonants in a row and swaps them:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| nebraska | C₁C₂ > C₂C₁ | **nerbaksa** |

For ease of typing, regular numbers also work: **C1C2 > C2C1**. If subscripts are left out, the program will make certain assumptions about what the rule means:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| ohio | VV > VxV | **ohixo** |

Subscripts can also be used after distinctive features: **[C +nasal]₁**; or sets: **{n,m}₁**. Multi-patterns can be captured inside curly brackets too:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| ahut | {VC}₁{VC}₂ > {VC}₂{VC}₁ | **utah** |

## **Reduplication**

Reduplication is a process in some languages where the whole word, or part of the word, is repeated exactly (or with a slight change). Although it often plays more of a grammatical function (such as forming the plural of a word) rather than being a global sound change.

A pattern can be reduplicated by changing the match to a double underscore **\_\_**. For example, the Pangasinan language (Philippines) reduplicates the first CV pattern to express the plural form:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| nevada | CV > \_\_ / #\_ | **nenevada** |

Full word reduplication can be achieved by matching the whole word **X\*** (where **X** matches any phoneme and **\*** matches any number of **X**) and replacing itself with double underscore:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| newyork | X\* > \_\_ | **newyorknewyork** |

Changing to a single underscore replaces the match with itself, which is not very useful unless it's combined with something else. For example in Pangasinan, if the noun starts with a CCV pattern, it reduplicates the first consonant only and first vowel. One way to achieve this would be to use subscript numbers on the CCV pattern like **C₁C₂V₁** and replace it with **C₁V₁\_**:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| florida | C₁C₂V₁ > C₁V₁\_ / #\_ | **foflorida** |

## **First and last match**

In same cases, you may want to only change the first or last match in the word. Use **<<** for first match and **>>** for last match:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| mississippi | s >> x | **missisxippi** |

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| mississippi | s << x | **mixsissippi** |

## **Coniditional statements**

There may be scenarios where you want to create sound changes based on whether a pattern exist in a word, and apply a different change if the pattern doesn't exist. Use the key words **IF** *this pattern is found* **THEN** *apply this sound change* **ELSE** *apply a different sound change*. The following rules tests if the word starts with **a**. If it doesn't, it applies a totally different sound change:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| alabama | IF #a THEN a > o ELSE a > e | **olobomo** |

The ELSE condition is optional and can be omitted. Multiple IF statements are permitted:

| **Input** | **Rule** | **Output** |
| --- | --- | --- |
| alabama | IF #s THEN #s > z IF a# THEN a > e | **elebeme** |

## **Non-Latin alphabets**

Custom spelling supports all Unicode alphabets and scripts, such as Japanese, Chinese, Cyrillic and even [Unicode Emojis](http://unicode.org/emoji/charts/full-emoji-list.html).

## **Order of rules**

The order of rules matter. Vulgar will find-and-replace the first rule to a word, then apply the next rule over the top of what it just did. This can be a problem if a IPA symbol appears in multiple patterns. For instance, the following ordering is problematic:

**ʊ > u  
aʊ > ow**

The intent here is for the diphthong **aʊ** (as in the word "cow") to change to **ow**. However, a previous rule has already changed any **ʊ** (as in the word "put") to **u**, which means **aʊ** has already changed to **au**. The easiest solution is to reverse the order of the rules:

**aʊ > ow  
ʊ > u**

Another solution is to use the exception **!** symbol: **ʊ > u / !a\_**.

## **Comments**

Double forward slahes **//** create a comment line that is ignored by the generator. This allows you to make notes to yourself in the Sound Change options:

// a turns to o before b

a > o / \_b

Comments can be used in all input boxes in Vulgarlang, as well as inside grammar tables in the Grammar Editor.