# Package: jugglr (via r-universe)

July 6, 2026

**Title** Juggling Pattern Validation and Visualisation

**Version** 0.0.0.9000

**Description** Validate and visualise juggling patterns expressed in siteswap notation. Supports vanilla, synchronous, multiplex, synchronous multiplex, and passing siteswap, with tools to check pattern validity and to retrieve raw throw data for custom visualisations. Patterns can be plotted as timeline arc diagrams or ladder diagrams, and animated via the 'JugglingLab' animation server (<https://jugglinglab.org/html/animinfo.html>).

**License** GPL (>= 3)

**Encoding** UTF-8

**Language** en-GB

**Config/roxygen2/version** 8.0.0

**Config/roxygen2/markdown** TRUE

**URL** https://github.com/EllaKaye/jugglr, https://ellakaye.github.io/jugglr/

**BugReports** https://github.com/EllaKaye/jugglr/issues

**Depends** R (>= 4.1.0)

**Imports** cli, dplyr, ggplot2, marquee, purrr, rlang, S7, stringr

**Suggests** doclisting, knitr, rmarkdown, testthat (>= 3.0.0), withr

**Config/testthat/edition** 3

**Collate** 'animate.R' 'generics.R' 'jugglr-package.R' 'utils-siteswap.R' 'utils-multiplex.R' 'utils-plotting.R' 'utils.R' 'utils-passing.R' 'utils-sync-multiplex.R' 'siteswap.R' 'multiplexSiteswap.R' 'passingSiteswap.R' 'utils-sync.R' 'synchronousMultiplexSiteswap.R' 'synchronousSiteswap.R' 'vanillaSiteswap.R'

**VignetteBuilder** knitr

**Config/pak/sysreqs** libfontconfig1-dev libfreetype6-dev libfribidi-dev libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev

**Repository** https://ellakaye.r-universe.dev

**Date/Publication** 2026-07-06 09:03:30 UTC

**RemoteUrl** https://github.com/EllaKaye/jugglr

**RemoteRef** HEAD

**RemoteSha** 3b320aa4d47c98b6dbcd21f5d19be3ce0a8ed447

# Contents



---

`animate` *Animate a juggling pattern*

---

## Description

Generates an animated GIF of a juggling pattern via the JugglingLab GIF server and displays it in the RStudio viewer (or default browser). If `path` is given the GIF is saved to disk instead.

## Usage

```
animate(
  pattern,
  colors = NULL,
  prop = NULL,
  bps = NULL,
  width = NULL,
  height = NULL,
  fps = NULL,
  slowdown = NULL,
  ...,
  path = NULL
)
```

## Arguments

| | |
|---|---|
| `pattern` | A siteswap pattern string (e.g. `"531"`) or any siteswap object: vanillaSiteswap, synchronousSiteswap, multiplexSiteswap, synchronousMultiplexSiteswap, or passingSiteswap. Passing patterns in p-notation (e.g. `"<3p 3\|3p 3>"`) animate correctly; passing patterns in fractional notation (e.g. `"<4.5 3 3 \| 3 4 3.5>"`) are not recognised by JugglingLab and cannot be animated. |
| `colors` | Optional. A vector of R colours (one per prop), or one of the special strings `"mixed"` or `"orbits"`. Passed to JugglingLab. |
| `prop` | Prop type: `"ball"`, `"ring"`, or `"image"`. If `NULL` (default) the JugglingLab default (a ball) is used. |
| `bps` | Beats per second (numeric scalar). Controls the animation speed. |
| `width, height` | Width and height of the animation in pixels (numeric scalars). |
| `fps` | Frames per second (numeric scalar). |
| `slowdown` | Slowdown factor (numeric scalar). The JugglingLab default is `2.0`; values greater than this slow the animation further, values less than `2.0` speed it up. |
| `...` | Additional named arguments passed to the JugglingLab GIF server. Pattern-setting arguments include `dwell`, `hands`, `body`, `propdiam`, `gravity`, `bouncefrac`, `squeezebeats`, `hss`, `handspec`, `dwellmax`, and `hold`. Animation arguments include `stereo`, `border`, `camangle`, `showground`, and `hidejugglers`. All values must be scalars. |
| `path` | Path to save the GIF (must end in `.gif`). If `NULL` (default) the animation is displayed in the viewer. |

## Details

Note that setting `colors` can introduce a short delay while the server renders the animation.

## Value

Invisibly returns the path to the temporary HTML file (when displaying) or the save path (when `path` is given).

## Examples

```
## Not run:
animate("531")
animate(vanillaSiteswap("531"), prop = "ring", bps = 5)
animate("531", path = tempfile(fileext = ".gif"))

## End(Not run)
```

`ladder` *Plot a ladder diagram for a siteswap*

## Description

Produces a ladder diagram showing prop paths between hands across beats. The rails represent the hands and the rungs represent the beats. Each throw connects to its catch: cross-hand throws are drawn as straight segments between the rails, while same-hand throws curve back to the same rail. Throws are coloured by prop using the colour-blind-friendly Okabe-Ito palette (a ggplot2 default scale is used beyond seven props), and beat numbers are labelled along the axis.

## Usage

```
ladder(
  siteswap,
  n_cycles = 3,
  direction = c("horizontal", "vertical", "h", "v"),
  title = TRUE,
  subtitle = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `siteswap` | A vanillaSiteswap, synchronousSiteswap, multiplexSiteswap, synchronousMultiplexSiteswap, or passingSiteswap object. |
| `n_cycles` | Number of complete cycles to simulate (default 3). |
| `direction` | Orientation of the diagram: `"horizontal"` (default, time runs left to right) or `"vertical"` (time runs top to bottom). Shorthands `"h"` and `"v"` are also accepted. |
| `title` | Logical. If `TRUE` (default), adds a title showing the siteswap sequence. Set to `FALSE` to suppress; override with `ggplot2::labs()` on the returned plot. |
| `subtitle` | Logical. If `TRUE` (default), adds a subtitle showing the siteswap type and number of props. Set to `FALSE` to suppress; override with `ggplot2::labs()` on the returned plot. |
| `...` | Additional arguments passed to methods. For `passingSiteswap` objects, `hand_gap` (default 2) sets the vertical spacing between each juggler's two hands in the diagram. |

## Details

`ladder()` is an S7 generic. There are methods available for the following classes:

- `jugglr::multiplexSiteswap`
- `jugglr::passingSiteswap`

- `jugglr::synchronousMultiplexSiteswap`
- `jugglr::synchronousSiteswap`
- `jugglr::vanillaSiteswap`

Where multiplex throws overlap (e.g. `[33]`), the lines fan apart slightly so they are visible. Passing patterns use one pair of rails per juggler, spaced by `hand_gap` and throws between jugglers cross between the rails.

The result is a standard ggplot2 object, so beyond the `title` and `subtitle` toggles you can modify it by adding any ggplot2 function, for example `ggplot2::labs()` or `ggplot2::scale_colour_manual()`.

## Value

A ggplot2 object.

## See Also

The siteswap constructors, e.g. `siteswap()` and `vanillaSiteswap()`.

Other siteswap visualisation: `throw_data()`, `timeline()`

## Examples

```
# Vanilla: straight cross-hand throws, curved same-hand throws
ladder(siteswap("423"))

# Synchronous: even-only beat numbering, crossing denoted by "x"
ladder(siteswap("(4,2x)(2x,4)"))
ladder(siteswap("(2,6x)([6x4x],2x)"))

# Multiplex: identical simultaneous throws fan into separate curves
ladder(siteswap("[33]"), n_cycles = 6)

# Passing: one pair of rails per juggler
ladder(siteswap("<3p 3|3p 3>"))
ladder(siteswap("<4.5 3 3 | 3 4 3.5>"))

# Vertical orientation, and customise like any ggplot2 object
ladder(siteswap("423"), direction = "v", subtitle = FALSE) +
  ggplot2::labs(title = "423 or W")
```

---

`multiplexSiteswap` *Multiplex siteswap*

---

## Description

Creates a multiplex siteswap object from a sequence in which square brackets group simultaneous throws from the same hand. For example, `"[43]1"` means the juggler throws heights 4 and 3 simultaneously on the first beat, then a single 1 on the second beat. Any slot without brackets is a single throw exactly as in vanilla siteswap.

## Usage

```
multiplexSiteswap(sequence = character(0))
```

## Arguments

| | |
|---|---|
| sequence | A single character string of multiplex siteswap notation, e.g. `"[43]1"` or `"[33]"`. |

## Value

A `multiplexSiteswap` S7 object.

## Additional properties

`@type` Always `"multiplex"` (read-only).

`@slots` List of integer vectors, one element per time slot. Single-throw slots are length-1 vectors; multiplex slots have length > 1 (read-only).

`@throws` Integer vector of all individual throw heights (flattened across all slots) (read-only).

`@period` Number of time slots per cycle (read-only).

`@symmetry` `"symmetrical"` when period is odd; `"asymmetrical"` when period is even (read-only).

`@n_props` Total of all throw heights divided by the period — the number of props required. (Note: `sum(throws) / period`, not `mean(throws)`.) (read-only).

`@can_throw` `TRUE` if for every slot the number of props thrown equals the number of props landing (no conservation violation) (read-only).

`@satisfies_average_theorem` `TRUE` if `n_props` is a whole number (read-only).

`@valid` `TRUE` if both `can_throw` and `satisfies_average_theorem` are `TRUE` (read-only).

## See Also

The visualisation functions `timeline()`, `ladder()` and `throw_data()`.

Other siteswap constructors: `passingSiteswap()`, `siteswap()`, `synchronousMultiplexSiteswap()`, `synchronousSiteswap()`, `vanillaSiteswap()`

## Examples

```
multiplexSiteswap("[43]1")
multiplexSiteswap("[33]")

s <- multiplexSiteswap("[43]1")
s@n_props
s@valid
```

passingSiteswap *Passing siteswap*

## Description

Creates a passing siteswap object from a multi-juggler sequence in <A|B> notation. Each section separated by | gives one juggler's throws; a p suffix (e.g. "3p") marks a throw that passes to the next juggler. Fractional notation (e.g. "<4.5 3 3 | 3 4 3.5>") is also supported, where .5 throws are passes and the second juggler is half a beat later.

## Usage

```
passingSiteswap(sequence = character(0))
```

## Arguments

| | |
|---|---|
| sequence | A single character string of passing siteswap notation, e.g. "<3p 3 3 3 3 \| 3p 3 3 3 3>" or "<4.5 3 3 \| 3 4 3.5>". Spaces within sections are ignored; <3p33\|3p33> and <3p 3 3\|3p 3 3> are equivalent. |

## Details

In p-notation, both jugglers are fully synchronised: they throw on the same beat and alternate hands together. In fractional notation, the second juggler is offset by half a beat in the combined timeline.

## Value

A passingSiteswap S7 object.

## Additional properties

@type Always "passing" (read-only).

@is_fractional TRUE for fractional notation; FALSE for p-notation (read-only).

@n_jugglers Number of jugglers (sections in the sequence) (read-only).

@sequences_by_juggler List of character vectors, one per juggler, each giving the raw tokens (e.g. c("3p", "3", "3")) (read-only).

@throws_by_juggler List of numeric vectors, one per juggler, of throw heights (integers for p-notation; may include .5 for fractional) (read-only).

@is_pass_by_juggler List of logical vectors, one per juggler, indicating which throws are passes (read-only).

@period Number of throws per juggler per cycle (read-only).

@symmetry "symmetrical" when period is odd; "asymmetrical" when period is even (read-only).

@n_props Total number of props (sum(all throws) / period) (read-only).

`@can_throw` `TRUE` if no two throws land on the same `(juggler, beat, hand)` triple (no collision) (read-only).

`@satisfies_average_theorem` `TRUE` if `n_props` is a whole number (read-only).

`@valid` `TRUE` if both `can_throw` and `satisfies_average_theorem` are `TRUE` (read-only).

## See Also

The visualisation functions `timeline()`, `ladder()` and `throw_data()`.

Other siteswap constructors: `multiplexSiteswap()`, `siteswap()`, `synchronousMultiplexSiteswap()`, `synchronousSiteswap()`, `vanillaSiteswap()`

## Examples

```
passingSiteswap("<3p 3 3 3 3 3 | 3p 3 3 3 3 3>")
passingSiteswap("<4p 3 | 3 4p>")

# Compact and spaced forms are equivalent
s1 <- passingSiteswap("<3p33|3p33>")
s2 <- passingSiteswap("<3p 3 3 | 3p 3 3>")
identical(s1@throws_by_juggler, s2@throws_by_juggler)

s <- passingSiteswap("<4p 3 | 3 4p>")
s@n_props
s@valid
```

---

# `siteswap` *Create a siteswap object*

---

## Description

Creates a typed siteswap object by detecting the notation style of `sequence`. Vanilla siteswaps (alphanumeric strings such as `"531"`) produce a vanillaSiteswap object; synchronous siteswaps in `(a,b)` notation such as `"(4,2x)*"` produce a synchronousSiteswap object; multiplex siteswaps with square-bracket groups such as `"[43]1"` produce a multiplexSiteswap object; passing siteswaps in `<A|B>` notation such as `"<3p 3 3 3 3 3 | 3p 3 3 3 3 3>"` produce a passingSiteswap object.

## Usage

```
siteswap(sequence)
```

## Arguments

| | |
|---|---|
| `sequence` | A single character string of siteswap notation. |

## Value

A vanillaSiteswap, synchronousSiteswap, multiplexSiteswap, synchronousMultiplexSiteswap, or passingSiteswap S7 object.

## See Also

The visualisation functions `timeline()`, `ladder()` and `throw_data()`.

Other siteswap constructors: `multiplexSiteswap()`, `passingSiteswap()`, `synchronousMultiplexSiteswap()`, `synchronousSiteswap()`, `vanillaSiteswap()`

## Examples

```
siteswap("531")
siteswap("(4,2x)*")
siteswap("[43]1")
siteswap("(2,4)([4x4],2x)")
siteswap("<3p 3 3 3 3 3 | 3p 3 3 3 3 3>")
```

---

# synchronousMultiplexSiteswap

*Synchronous multiplex siteswap*

---

## Description

Creates a synchronous multiplex siteswap object. This combines synchronous notation (both hands throw simultaneously, written as pairs in parentheses) with multiplex notation (square brackets group simultaneous throws from the same hand). For example, `"(2,4)([4x4],2x)"` is a 4-prop pattern where the second slot has hand 0 throwing two balls simultaneously.

## Usage

```
synchronousMultiplexSiteswap(sequence = character(0))
```

## Arguments

| | |
|---|---|
| sequence | A single character string of synchronous multiplex siteswap notation, e.g. `"(2,4)([4x4],2x)"`. |

## Details

All individual throw heights must be even. An `x` suffix marks a crossing throw; a trailing `*` indicates the pattern alternates between two mirrored versions.

## Value

A `synchronousMultiplexSiteswap` S7 object.

### Additional properties

`@type` Always `"synchronous multiplex"` (read-only).

`@full_sequence` The expanded sequence with the `*` shorthand resolved (read-only).

`@throws` Character vector of all slot throw strings across one expanded cycle. Multiplex groups are kept as single elements, e.g. `"[4x4]"` (read-only).

`@throws_by_hand` Named list with elements `hand_1` and `hand_2`, each a character vector of throw strings per slot (one per sync slot) (read-only).

`@period` Number of throw slots per full cycle (2 × number of sync slots, since both hands throw on every beat; always even) (read-only).

`@symmetry` `"symmetrical"` if the pattern is its own mirror image; `"asymmetrical"` otherwise (read-only).

`@n_props` Number of props: sum of all throw heights divided by the period (read-only).

`@can_throw` `TRUE` if thrown and landing prop counts balance at every (slot, hand) pair within one cycle (read-only).

`@satisfies_average_theorem` `TRUE` if `n_props` is a whole number (read-only).

`@valid` `TRUE` if both `can_throw` and `satisfies_average_theorem` are `TRUE` (read-only).

### See Also

The visualisation functions `timeline()`, `ladder()` and `throw_data()`.

Other siteswap constructors: `multiplexSiteswap()`, `passingSiteswap()`, `siteswap()`, `synchronousSiteswap()`, `vanillaSiteswap()`

### Examples

```
synchronousMultiplexSiteswap("(2,4)([4x4],2x)")
synchronousMultiplexSiteswap("(4,[42x])*")

s <- synchronousMultiplexSiteswap("(2,4)([4x4],2x)")
s@n_props
s@valid
```

---

## synchronousSiteswap *Synchronous siteswap*

---

### Description

Creates a synchronous siteswap object from a two-handed simultaneous-throw sequence. Synchronous notation describes patterns where both hands throw at the same time, written as pairs of throw heights in parentheses such as `"(4,4)"` or `"(4,2x)*"`. An `x` suffix marks a crossing throw; a trailing `*` indicates the pattern alternates between two mirrored versions.

### Usage

```
synchronousSiteswap(sequence = character(0))
```

## Arguments

| | |
|---|---|
| sequence | A single character string of synchronous siteswap notation, e.g. `"(4,4)"` or `"(4,2x)*"`. |

## Value

A `synchronousSiteswap` S7 object.

## Additional properties

`@type` Always `"synchronous"` (read-only).

`@full_sequence` The expanded sequence with the `*` shorthand resolved (read-only).

`@throws` Character vector of all individual throw values across one expanded cycle (read-only).

`@throws_by_hand` Named list with elements `hand_1` and `hand_2`, each a character vector of throws for that hand per slot (read-only).

`@period` Number of throw slots per full cycle (counts both hands per simultaneous beat, so always even) (read-only).

`@symmetry` `"symmetrical"` if the pattern is its own mirror image; `"asymmetrical"` otherwise (read-only).

`@n_props` Mean of the slide sequence, equal to the number of props (read-only).

`@can_throw` `TRUE` if no collisions occur in the slide sequence (read-only).

`@satisfies_average_theorem` `TRUE` if `n_props` is a whole number (read-only).

`@valid` `TRUE` if both `can_throw` and `satisfies_average_theorem` are `TRUE` (read-only).

## See Also

The visualisation functions `timeline()`, `ladder()` and `throw_data()`.

Other siteswap constructors: `multiplexSiteswap()`, `passingSiteswap()`, `siteswap()`, `synchronousMultiplexSiteswa`
`vanillaSiteswap()`

## Examples

```
synchronousSiteswap("(4,4)")
synchronousSiteswap("(4,2x)*")

s <- synchronousSiteswap("(4,2x)*")
s@n_props
s@valid
```

`throw_data` *Get throw data for a siteswap*

## Description

Returns a data frame describing each throw in the simulated pattern, including which beat and hand each throw originates from, where it lands, and which prop is thrown. This underpins `timeline()` and `ladder()`, and can be used directly for custom visualisations.

## Usage

```
throw_data(siteswap, n_cycles = 3, ...)
```

## Arguments

| | |
|---|---|
| `siteswap` | Any siteswap object: a `vanillaSiteswap`, `synchronousSiteswap`, `multiplexSiteswap`, `synchronousMultiplexSiteswap`, or `passingSiteswap`. |
| `n_cycles` | Number of complete cycles to simulate (default 3). Increase for patterns with many props to ensure all appear in the data. Setting `n_cycles` >= `period` * `n_props` * 2 guarantees each prop is thrown at least twice. |
| `...` | Additional arguments passed to methods. |

## Value

A data frame with one row per throw and columns:

- `beat`: throw beat index
- `hand`: throwing hand (0 or 1)
- `throw`: throw height (integer)
- `catch_beat`: beat on which the prop lands
- `catch_hand`: hand that catches (0 or 1)
- `prop`: prop identifier (integer)

Synchronous types add an `is_crossing` column. Passing types instead include `juggler`, `is_pass`, and `catch_juggler` columns identifying which juggler throws and catches each prop.

## See Also

The siteswap constructors, e.g. `siteswap()` and `vanillaSiteswap()`.

Other siteswap visualisation: `ladder()`, `timeline()`

## Examples

```
s <- vanillaSiteswap("531")
throw_data(s)
```

`timeline` *Plot a timeline diagram for a siteswap*

## Description

Produces a timeline (arc) diagram showing the trajectory of each prop across beats. Each arc represents a throw, with height proportional to the throw value and coloured by prop using the colour-blind-friendly Okabe-Ito palette (a ggplot2 default scale is used beyond seven props). The x-axis labels show the throw heights from the siteswap sequence.

## Usage

```
timeline(siteswap, n_cycles = 3, title = TRUE, subtitle = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `siteswap` | A vanillaSiteswap, synchronousSiteswap, multiplexSiteswap, synchronousMultiplexSiteswap, or passingSiteswap object. |
| `n_cycles` | Number of complete cycles to simulate (default 3). A warning is issued if not all props appear within the simulated range. |
| `title` | Logical. If `TRUE` (default), adds a title showing the siteswap sequence. Set to `FALSE` to suppress; override with `ggplot2::labs()` on the returned plot. |
| `subtitle` | Logical. If `TRUE` (default), adds a subtitle showing the siteswap type and number of props. Set to `FALSE` to suppress; override with `ggplot2::labs()` on the returned plot. |
| `...` | Additional arguments passed to methods. |

## Details

`timeline()` is an S7 generic. There are methods available for the following classes:

- `jugglr::multiplexSiteswap`
- `jugglr::passingSiteswap`
- `jugglr::synchronousMultiplexSiteswap`
- `jugglr::synchronousSiteswap`
- `jugglr::vanillaSiteswap`

See **Behaviour by siteswap type** section below for details on the differences between the resulting plots.

The result is a standard ggplot2 object, so beyond the `title` and `subtitle` toggles you can modify it by adding any ggplot2 function, for example `ggplot2::labs()` or `ggplot2::scale_colour_manual()`.

## Value

A ggplot2 object.

### Behaviour by siteswap type

- **Vanilla and multiplex** patterns are drawn single-sided, with all arcs above a single baseline. The x-axis labels are the throw heights, and multiplex labels keep their slot brackets (for example `[54]`).
- **Synchronous and synchronous multiplex** patterns are drawn two-sided: one hand's arcs sit above a faint centre line and the other hand's arcs sit below it. Each hand gets its own x-axis labels, and the subtitle notes the two-sided layout.
- **Multiplex** patterns (synchronous or not) that throw two or more identical props on the same beat fan those arcs to slightly different peak heights, so otherwise-overlapping throws render as concentric parabolas.
- **Passing** patterns are drawn with one lane per juggler; passes arc between lanes. Fractional notation offsets the beat positions accordingly.

## See Also

The siteswap constructors, e.g. `siteswap()` and `vanillaSiteswap()`.

Other siteswap visualisation: `ladder()`, `throw_data()`

## Examples

```
# Vanilla: single-sided arcs
timeline(siteswap("423"))

# Synchronous: two-sided, hands mirrored about a centre line
timeline(siteswap("(4,2x)(2x,4)"))
timeline(siteswap("(2,6x)([6x4x],2x)"))

# Multiplex: identical simultaneous throws fan to distinct heights
timeline(siteswap("[54]24"))
timeline(siteswap("[33]"))

# Passing: one lane per juggler, with passes arcing between them
timeline(siteswap("<3p 3|3p 3>"))
timeline(siteswap("<4.5 3 3 | 3 4 3.5>"))

# Increasing n_cycles helpful for sequences with short period
timeline(siteswap("3"), n_cycles = 6)

# ggplot2 object: customise like any other plot
timeline(siteswap("423")) + ggplot2::labs(title = "423 or W")
```

---

`vanillaSiteswap` *Vanilla siteswap*

---

## Description

Creates a vanilla siteswap object from an alphanumeric siteswap sequence such as `"531"` or `"97531"`. Vanilla siteswap describes solo juggling patterns where one prop is thrown per beat, alternating hands.

## Usage

```
vanillaSiteswap(sequence = character(0))
```

## Arguments

| | |
|---|---|
| `sequence` | A single character string of vanilla siteswap notation (digits and letters only, e.g. `"531"`). |

## Value

A `vanillaSiteswap` S7 object.

## Additional properties

`@type` Always `"vanilla"` (read-only).

`@throws` Integer vector of throw heights for one cycle (read-only).

`@period` Number of throws per cycle (length of `throws`) (read-only).

`@symmetry` `"symmetrical"` when period is odd (pattern repeats with swapped hands); `"asymmetrical"` when period is even (read-only).

`@n_props` Mean throw height, equal to the number of props required (read-only).

`@can_throw` `TRUE` if no two throws land on the same beat (no collisions) (read-only).

`@satisfies_average_theorem` `TRUE` if `n_props` is a whole number (read-only).

`@valid` `TRUE` if both `can_throw` and `satisfies_average_theorem` are `TRUE` (read-only).

## See Also

The visualisation functions `timeline()`, `ladder()` and `throw_data()`.

Other siteswap constructors: `multiplexSiteswap()`, `passingSiteswap()`, `siteswap()`, `synchronousMultiplexSiteswap()`, `synchronousSiteswap()`

## Examples

```
vanillaSiteswap("531")
vanillaSiteswap("97531")

s <- vanillaSiteswap("531")
s@n_props
s@valid
```

# Index