

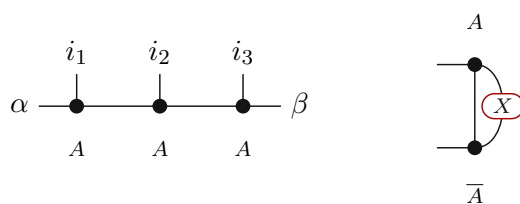
tikz-tensor-networks

Language manual

Declarative tensor-network diagrams in L^AT_EX

September 2026

manual for **tikz-tensor-networks** version 0.8.0



A picture chooses one layout; its options declare topology and policy; its body declares atoms, connections, regions, and annotations.

Contents

1	First Principles	2
2	The Choice of Layout	3
3	Tutorial	4
4	Practical Techniques	14
5	Catalogue of Constructs	18
6	Summary of the Grammar	32
7	Custom Atoms and Typographic Style	38
8	The Benchmark Suite	39
9	Recovery from Errors	40
10	References	41

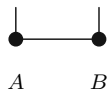
1 First Principles

Welcome to **tikz-tensor-networks**! The goal of this system is to make the creation of tensor-network diagrams in $\text{\TeX}$ as pleasant, natural, and mathematically rigorous as typesetting ordinary formulas. If you have ever tried to draw a matrix product state or a tensor contraction by placing low-level coordinates by hand, you know how quickly manual geometry obscures the algebraic meaning of the network. In **tikz-tensor-networks**, we declare the mathematics, and let $\text{\TeX}$ draw the ink.

To begin, you need only load the package and declare your first network:

```
\usepackage{tikz-tensor-networks}
\begin{tenkz}[cols=2, physical=up] \tn{A} & \tn{B} \end{tenkz}
```

which immediately produces the elementary two-site chain:



For a working repository checkout, simply add `tex/tenkz/` to your `TEXINPUTS` search path; for an archived article, ship `tikz-tensor-networks.sty` and its companion files beside the document source. $\text{\XeLaTeX}$ is our reference engine, both for high-quality print output and for the reproducible XDV-to-SVG web pipeline.

The golden rule of **tikz-tensor-networks** can be stated in a single sentence:

A picture chooses one layout; its options declare topology and policy; its body declares atoms, connections, regions, and annotations.

Whenever you encounter a `tikz-tensor-networks` picture—or sit down to write one of your own—proceed in this natural order. First, the environment establishes the placement grammar. Second, its optional arguments declare global facts and policies shared by the entire picture. Third, the commands within the body declare the mathematical entities themselves; they describe the *structure* of the network, rather than giving imperative painting instructions.

Five grammatical classes span the entire public vocabulary: *atoms*; *connections and closures*; *regions and annotations*; *composition*; and *setup and extension*. A command introduces an object belonging to one of these classes, while a key modifies the record being declared. This clean separation keeps the grammar delightfully compact and easy to read directly from source.

Scoping in `tikz-tensor-networks` is completely explicit. A declaration made via `\tnset` applies to the whole document; an option on an environment applies to that picture; an option on an atom applies to that single object; and a join or side-policy option governs that particular connection. Furthermore, a shared key name retains exactly one value type and one meaning across every environment where it appears.

Notice that open ink is genuine mathematical data: a matrix keeps its two virtual indices, an operator channel keeps its input and output indices, and a scalar keeps none at all. Closing an uncontracted bond merely for aesthetic “neatness” would change the mathematical object itself! To ensure that your diagrams always say what they mean, `tikz-tensor-networks` writes a companion `.tnlog` audit stream during compilation, recording the resolved model and boundary signature used for rendering. The log also records the type of every port: a contracted index carries its type in the record even though it strokes identically on the page, so that physical contractions and virtual bonds stroke alike without confusing the underlying physics.

2 The Choice of Layout

When you sit down to typeset a tensor network, your very first design decision is to select the right environment. A simple yet profound principle will guide you: **choose from topology, not appearance.**

Layout	Use it when	Recognition test
<code>tenkz</code>	The picture is a tensor network with regular contractions.	Rows are layers, columns are lattice sites, and neighbouring nodes contract.

In `tikz-tensor-networks`, the overwhelming majority of diagrams in quantum many-body physics begin with `tenkz`. A one-dimensional Matrix Product State (MPS), a Matrix Product Operator (MPO), a two-dimensional Projected Entangled Pair State (PEPS), or a double-layer norm network is not a separate, ad-hoc

layout: each is simply a `tenkz` picture equipped with an appropriate frame—such as the default 1D chain, the convenient `lattice=` and `planes` options for 2D quantum lattices, or a declared geometric basis for oblique projections.

Resist changing layouts for style. It is often tempting to switch environments whenever a diagram looks slightly different, but remember that the layout establishes the shared metric, node skins, semantic roles, labels, and contraction policies across every figure in your paper. An MPS chain does not cease to be an MPS merely because it runs vertically on the page! Choose the layout that reflects the underlying physical contraction geometry, and then declare your stylistic preferences through its public keys.

When none fits. What should you do if you encounter a diagram that seems to defy the standard grammar? Before reaching for raw TikZ drawing commands or inventing ad-hoc syntax, pause to identify the missing mathematical class. In the *tikz-tensor-networks* design philosophy, a new key requires at least three independent, manifested consumers in the literature, while a new command requires a genuinely new grammatical class. Until an extension clears that gate, treat the need as a capability request, rather than burying uncanonical drawing hacks inside your document.

3 Tutorial

This chapter is an invitation to learn *tikz-tensor-networks* by doing. We shall climb a gentle ladder of eight examples, starting from a single matrix product state (MPS) and culminating in the expectation value of a two-dimensional PEPS state. In the spirit of *The T_EXbook*, each step introduces exactly one new idea built upon the ones before it. Along the way, we shall meet the familiar objects of tensor-network theory—matrix product states, transfer operators, gauge transformations, 2D PEPS sheets, and double-layer norm networks—calling each by its standard mathematical name in the quantum information and condensed matter literature (e.g. arXiv:2011.12127) as it arrives.

The package loads the diagram language automatically. The examples use its current commands and keys throughout.

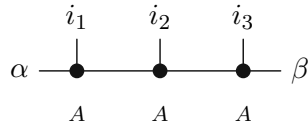
Every code example is complete: paste it into any document that loads *tikz-tensor-networks* and it will compile exactly as printed. As you read each example, look out for the little “danger-

ous bend” signs () in the margin: they flag technical subtleties and deeper design insights that you may safely skip on a first reading. At the end of each section, you will find short exercises to test your understanding; their answers await you at the end of the chapter. (Try to solve them before looking!)

3.1 A chain of tensors

Let us begin with the foundation of one-dimensional tensor networks: the matrix product state (MPS). Each lattice site carries one tensor A , with two horizontal virtual indices of bond dimension D (or χ) that contract with adjacent sites, and one vertical physical index of dimension d corresponding to the local Hilbert space. Here is a three-site segment:

EXAMPLE 3.1 · three sites of a matrix product state¹



```
% Formula: (A^{i_1} A^{i_2} A^{i_3})_{\alpha\beta}
% Ink: bonds contract; typed ports declare the open indices.
\begin{tenkz}[cols=3]
  \tn[ports={180:virtual:$\alpha$, 0:virtual, 90:physical:$i_1$}]{A} &
  \tn[ports={180:virtual, 0:virtual, 90:physical:$i_2$}]{A} &
  \tn[ports={180:virtual, 0:virtual:$\beta$, 90:physical:$i_3$}]{A}
\end{tenkz}
```

$$(A^{i_1} A^{i_2} A^{i_3})_{\alpha\beta}$$

¹ The horizontal bonds are summed virtual indices. The two stubs are the open matrix indices α and β ; each upward leg is one physical index.

kernel-boundary|signature=open:e,
open:w, phys:n, phys:n, phys:n

There! You have just typeset your first matrix product state contraction $(A^{i_1} A^{i_2} A^{i_3})_{\alpha\beta}$. Notice how cleanly the code expresses the underlying mathematics. The `tenkz` environment holds one typed tensor network in a single, coherent frame: `cols=3` establishes three lattice sites along the 1D chain, each `\tn` declares a local tensor at that site, and the familiar `&` alignment tab advances to the next site. Because adjacent sites contract by default—forming the internal virtual bonds—the only indices you need to declare explicitly are the open ones: the boundary virtual indices α, β and the physical legs i_1, i_2, i_3 .

The `ports=` key defines each port around an atom's perimeter and optionally attaches an index label. (A port is simply the geometric anchor where an index wire meets its tensor node.) A typed port that finds no partner cell to contract with remains an open physical or virtual index. Notice also the marginal annotation: during compilation, *tikz-tensor-networks* audits the picture and writes its resolved boundary signature into the `.tnlog` event stream: two virtual boundary stubs, three physical legs, and nothing you did not ask for.

Users of the *beamer* presentation package will be pleased to learn that *tikz-tensor-networks* pictures require no special fuss. Because *beamer* scans a frame's body before typesetting, an alignment tab `&` inside a diagram reaches *tikz-tensor-networks* as an ordinary token. The environment captures its body and interprets the tab directly, so your diagrams compile happily inside standard slides without needing the `[fragile]` option.

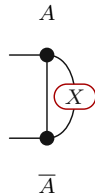


EXERCISE 3.1 Try drawing a single matrix M by itself—one tensor atom, both virtual indices open, with no physical leg—and determine what boundary signature *tikz-tensor-networks* writes to the log.

3.2 Layers and boundary policies

Now let us add a second idea to our repertoire: rows represent layers. Our object of interest is the workhorse of matrix-product algorithms (such as DMRG and TEBD), the transfer operator $\mathcal{E}_A(X) = \sum_i A^i X A^{i\dagger}$, which sandwiches an input matrix X on the virtual space between a ket layer above and a bra layer below.

EXAMPLE 3.2 · the channel applied to X ²



```
% Formula: E_A(X) = sum_i A^i X A^{i dagger}.
% Boundary: the input closes east; the matrix output stays
% open west.
\begin{tenkz}[rows={ket,bra}, cols=1, west=open, east={cup=$X$}]
  \tn[label pos=90]{A} \\\
  \tn[label pos=270]{\overline{A}}
\end{tenkz}
```

Here `rows=` declares a ket layer standing over a bra layer, and the frame automatically contracts their facing physical ports—producing the vertical bond that sums over the physical index i . Notice that complex conjugation is pure mathematics: $\overline{A}$ is an ordinary label name, not a cryptic internal switch!

The perimeter sides declare boundary policy from a tidy four-word alphabet: `open` mints legs and records them in the boundary signature; `none` (the default) leaves a side unadorned; `trace` loops a row back onto itself; and `cup` closes adjacent layers together. In this example, `east={cup=$X$}` closes the eastern pair through the matrix X , while the western pair remains open, yielding a linear map on virtual matrices—precisely matching the two-entry signature quoted in the margin.

A labelled cup is what computer scientists call “syntactic sugar”: behind the scenes, *tikz-tensor-networks* expands it into an ordinary `cup` wire plus a ring atom seated at its midpoint. The kernel record stream never contains an ad-hoc “labelled cup” entity, but only a wire named `cup-1-2` and an atom on it. Every convenience sugar row in the language registry defines its canonical expansion this way, and authors who insist on absolute purity can specify `\tnset{strict}` to reject sugar outright.



EXERCISE 3.2 Draw the full transfer matrix $E = \sum_i A^i \otimes \overline{A}^i$ of the state—the map on the D^2 -dimensional doubled virtual space: a ket layer over a bra layer, with the physical pair contracted and all four virtual ends left open.

EXERCISE 3.3 A contraction loop with no tensor on it—the spelling `\tnwire[closed]{(1,1)}{(2,1)}`—refuses to compile. Why?

$$\mathcal{E}_A(X) = \sum_i A^i X A^{i\dagger}$$

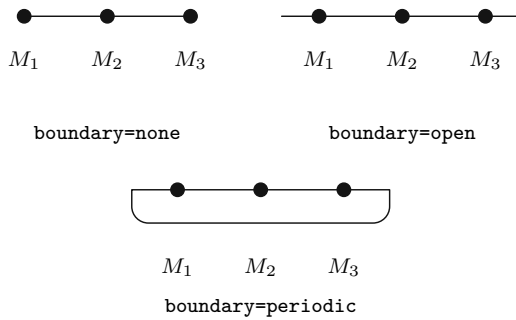
² X closes the east input pair. The west pair remains open because the output is a matrix; the vertical connection sums the physical index i .

`kernel-boundary|signature=open:w, open:w`

3.3 Boundary conditions: open, periodic, and closed

A single row of matrices can represent three distinct mathematical objects frequently encountered in tensor-network algorithms: an open matrix product $(M_1 M_2 M_3)_{\alpha\beta}$ with free virtual boundary bonds, a finite chain with trivial 1D boundary vectors where outer virtual bonds close without exposed ink, and the periodic trace $\text{tr}(M_1 M_2 M_3)$ representing a state with periodic boundary conditions (PBC). The convenient `boundary=` key sets both `west=` and `east=` policies simultaneously. Watch how easily its three settings produce all three objects from the very same row of tensors:

EXAMPLE 3.3 · one row, three boundary conditions³



```
% Formula: M_1 M_2 M_3 under three side policies.
\begin{tenkz}[rows={wire}, cols=3, variant]
  \tn{$M_1$} & \tn{$M_2$} & \tn{$M_3$}
\end{tenkz}
```

This is a delightful demonstration of “small multiples” (in Edward Tufte’s sense): all three panels share identical code, varying only in the boundary word beneath them. The setting `none` explicitly spells the default—a side that specifies no policy exposes no dangling ink, corresponding to a finite chain whose outer boundary closes with trivial 1D boundary vectors. The word `open` turns the row into an open matrix $(M_1 M_2 M_3)_{\alpha\beta}$ with exposed boundary virtual bonds, while `periodic` (convenient sugar for `west=trace`, `east=trace`) loops the virtual bond back upon itself to compute the trace. The atoms and internal bonds remain completely identical; only the boundary signature moves!

A boundary leg appears only when it is declared. The setting `none` states the default; use `open` when the diagram should expose an index.

EXERCISE 3.4 Draw the coefficient of a periodic matrix product state—the scalar $\text{tr}(M_1 M_2 M_3)$ —and state what boundary signature it produces.

3.4 Highlighting subsystems and regions

In real-space renormalization group (RG) arguments, DMRG site-blocking, or entanglement bipartition studies, one frequently groups several adjacent sites into an effective block tensor $A^{[3]}$ or defines a

$(M_1 M_2 M_3)_{\alpha\beta}$

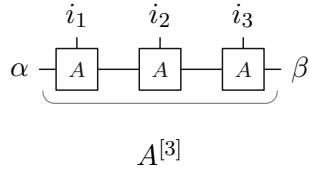
³ Left to right: the outer bonds close trivially and nothing is exposed; both ends open and the row is a matrix; the trace returns the row to itself and the picture is a scalar. The quoted signature is the open panel’s; the other two are empty.

`kernel-boundary|signature=open:e,`
`open:w`



subsystem A . The underlying tensor contraction does not change; instead, an annotation or “mark” speaks over the grouping.

EXAMPLE 3.4 · blocking three sites⁴



```
% Formula: the blocked word A^{[3]}; the mark adds no index.
\begin{tenkz}[cols=3]
  \tn[skin=box, ports={180:virtual:$\alpha$, 0:virtual, 90:physical:$i_1$}]{A} &
  \tn[skin=box, ports={180:virtual, 0:virtual, 90:physical:$i_2$}]{A} &
  \tn[skin=box, ports={180:virtual, 0:virtual:$\beta$, 90:physical:$i_3$}]{A} \\
  \tnmark[form=bracket]{(1,1) .. (1,3)}{\$A^{[3]}\$}
\end{tenkz}
```

The `\tnmark` command annotates a designated target without altering the underlying network topology. The target here is specified as a selector: $(1,1)$ names a cell in the frame, and the two-dot range $(1,1) \dots (1,3)$ selects all records spanning between the two cells in sequence. A bracket mark with no side specified naturally speaks from the south, where a horizontal brace under a formula belongs. Observe the margin note: the boundary signature is entirely identical to Example 3.1! Marks own annotations, never topology: if you delete every mark from a picture, not a single tensor contraction changes.

Notice that the span is written with two dots $(\dots)$, rather than a hyphen. This is because internal generated record names already use hyphens (for instance, a cluster atom is named `a-2-2`); using two dots keeps ranges visually unambiguous and easy for $\text{\TeX}$ to parse.

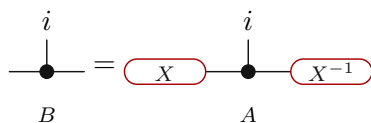


EXERCISE 3.5 Try restating the same blocked run using a tinted enclosure whose label is placed below the bounding contour.

3.5 Diagrammatic equations and verification

We now arrive at one of the crowning features of *tikz-tensor-networks*: composing multiple diagrams into an equation, and having $\text{\TeX}$ verify that index contractions remain balanced across equality signs! Our example is the fundamental gauge freedom of matrix product states: two sets of tensors A and B , related by an invertible gauge matrix X via $B^i = XA^iX^{-1}$, generate the exact same quantum state.

EXAMPLE 3.5 · a gauge identity, audited⁵



$$B^i = XA^iX^{-1}$$

⁵ The rings are matrices on the wire and add no indices. Both panels expose one west stub, one east stub, and one physical leg, so the audit records the relation as equal.

```
check|scope=1|relation=1|
result=equal|signature=open:e,
open:w, phys:n
```

```
% Formula:  $B^i = X A^i X^{-1}$ ; one relation, audited.
\[
\begin{tenkzeq}[check=signature]
\begin{tenkz}[cols=1]
\tn[ports={180:virtual, 0:virtual, 90:physical:$i$}]{B}
\end{tenkz}
=
\begin{tenkz}[cols=3]
\tn[skin=ring, ports={180:virtual, 0:virtual}]{X} &
\tn[ports={180:virtual, 0:virtual, 90:physical:$i$}]{A} &
\tn[skin=ring, ports={180:virtual, 0:virtual}]{X^{-1}}
\end{tenkz}
\end{tenkzeq}
\]
```

The `tenkzeq` environment neatly balances tensor diagrams on either side of standard mathematical relation symbols under a unified scale and baseline. Even better, it audits! In tensor network theory, a diagrammatic identity requires both sides of an equality to match in tensor rank and index structure: the same set of open physical and virtual indices. With `check=signature` active (which is the default setting), *tikz-tensor-networks* verifies this balance automatically, recording the `check` event in the `.tnlog` stream. This checks boundary index compatibility: both sides must carry matching physical and virtual open ports. The mathematical identity itself remains the author's responsibility.

What happens when an equation fails the audit? Let us deliberately omit the physical leg from tensor B , and see how the compiler responds:

A DELIBERATE REFUSAL 3.1 · the audit catches a dropped leg⁶

```
% Refused: B lost its physical leg, so the relation's two
% sides no longer expose the same boundary.
\[
\begin{tenkzeq}[check=signature]
\begin{tenkz}[cols=1]
\tn[ports={180:virtual, 0:virtual}]{B}
\end{tenkz}
=
\begin{tenkz}[cols=3]
\tn[skin=ring, ports={180:virtual, 0:virtual}]{X} &
\tn[ports={180:virtual, 0:virtual, 90:physical:$i$}]{A} &
\tn[skin=ring, ports={180:virtual, 0:virtual}]{X^{-1}}
\end{tenkz}
\end{tenkzeq}
\]
```

[TKZ-EQ-SIGNATURE] sides 1 and 2 expose unequal boundaries: (open:e, open:w) versus (open:e, open:w, phys:n).

Don't be alarmed by compiler refusals! Think of them as a vigilant proofreader standing by your shoulder. The diagnostic code names the exact check that failed, and prints both boundary signatures side by side so the missing index is immediately apparent. The remedy here is simply to restore the port `90:physical:$i$` to tensor B .

Mathematical relations are one of three joiner classes in `tenkzeq`. A summation also requires matching signatures; a product, by contrast, contracts facing cuts (eastern entries of the left factor cancelling western entries of the right) and exposes the remainder.

⁶ The left panel now exposes two indices, the right three. The correction is one token: restore the port `90:physical:$i$` on B .



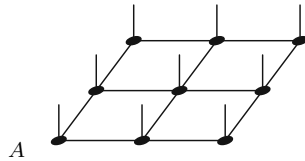
Should you ever need to state an equality where the two sides are legitimately unequal for reasons outside the diagram (such as an implicit projection), you can record an explicit opt-out using `off={<relation>: <reason>}`. The waiver is faithfully logged to the event stream, ensuring that the audit cannot be silenced covertly.

EXERCISE 3.6 Why does the spelling `check={signature=false}` trigger a refusal?

3.6 Two-dimensional lattices and PEPS

Now that we have explored one-dimensional chains, we are ready to step into the second dimension. Our model object is the projected entangled pair state (PEPS), which represents the wavefunction $|\psi(A)\rangle$ of a two-dimensional quantum lattice system (such as a 2D spin model): one five-index tensor A per site, four virtual indices of bond dimension D contracted with its four planar nearest neighbours, and one physical index of local Hilbert space dimension d standing perpendicular to the sheet.

EXAMPLE 3.6 · a PEPS on a 3×3 patch⁷



```
% Formula: the PEPS  $|\psi(A)\rangle$  generated by one tensor  $A$ .
% Ink: in-sheet bonds contract; each transverse leg is one
% physical index.
\begin{tenkz}[lattice={3x3}, frame=plane, physical=up]
  \tn{} & \tn{} & \tn{} \\\
  \tn{} & \tn{} & \tn{} \\\
  \tn[label pos=225]{A} & \tn{} & \tn{}
\end{tenkz}
```

$|\psi(A)\rangle$

⁷ Every dot is the same tensor A . The in-sheet bonds are the contracted virtual indices; each transverse leg is one physical index, and the signature lists the nine of them.

```
kernel-boundary|signature=phys:n,
phys:n, phys:n, phys:n, phys:n,
phys:n, phys:n, phys:n, phys:n
```

Just two concise options bring this two-dimensional sheet onto the page. The option `lattice={3x3}` is convenient shorthand for three rows of three sites each, whose cells contract with their neighbours just as chain cells do—these form the in-plane virtual bonds mediating 2D entanglement. Next, `frame=plane` projects the grid into perspective, while `physical=up` sprouts one physical leg per site along the plane’s transverse axis, pointing vertically on the page into the local Hilbert space. Because no boundary policy is stated on the perimeter, no virtual index is exposed: the patch has open boundary conditions, and its boundary signature lists exactly the nine physical legs of the state.

A projected plane possesses three spatial directions, but only two lie within the sheet itself. Specifying `ports={90:physical}` on a planar atom refers to the in-sheet north face, not the upward transverse leg; the transverse physical axis is governed through the picture policy. A PEPS tensor is described by four numeric pla-



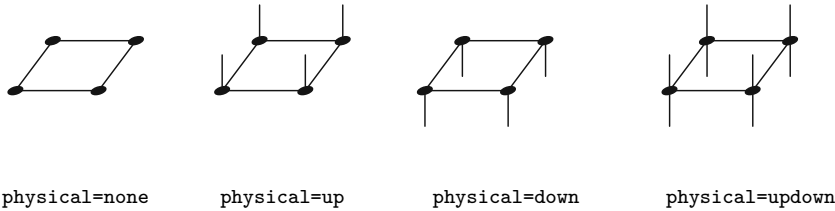
nar ports plus the frame policy—never by an artificial fifth angle attempting to fake the 3D projection.

EXERCISE 3.7 Try drawing a single PEPS tensor by itself—the five-index A_{uldr}^i : four open virtual ports labelled u, l, d, r in the sheet, and the physical leg pointing above.

3.7 Planar projections and viewing angles

The very same sheet of tensors can represent four fundamental concepts of the two-dimensional literature, selected by a single enumeration key: a fully contracted network computes a classical partition function Z or scalar overlap; physical legs pointing up define the quantum ket state $|\psi\rangle$; physical legs pointing down yield the conjugate bra $\langle\psi|$; and physical legs extending in both directions produce a Projected Entangled Pair Operator (PEPO) O representing a 2D Hamiltonian or density matrix.

EXAMPLE 3.7 · one sheet, four readings⁸



$Z, |\psi\rangle, \langle\psi|, O$

⁸ Left to right: no physical legs, a fully contracted scalar; legs up, the state; legs down, its conjugate; legs both ways, the PEPS operator. The quoted signature is the legs-up panel's.

kernel-boundary|signature=phys:n,
phys:n, phys:n, phys:n

```
% Formula: Z, |psi>, <psil, O -- one sheet, four leg policies.
\begin{tenkz}[lattice={2x2}, frame=plane, variant]
  \tn{} & \tn{} \\
  \tn{} & \tn{}
\end{tenkz}
```

Here again we see the elegance of small multiples: the four panels share every token of input except the policy key beneath them. Moreover, the boundary signature distinguishes a ket from a bra even before the diagram is rendered: a leg extending above the sheet logs **phys:n**, while one extending below logs **phys:s**. The event stream tells them apart immediately, and the PEPO panel exposes both lists simultaneously.

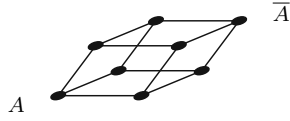
EXERCISE 3.8 A PEPS with periodic boundary conditions naturally lives on a torus. The single key **surface=torus** wraps all four sides of the patch—convenient sugar for **trace** on each of **west=**, **east=**, **north=**, and **south=**. What does this change in the boundary signature of Example 3.6, and what does it change in the network?

3.8 Double layers and expectation values

Virtually every expectation value or correlation function in the PEPS literature is computed within the “double layer”: a conjugate bra sheet laid atop the ket state, with each site’s physical index contracted with its conjugate partner to yield an effective 2D network

with squared bond dimension D^2 . The norm $\langle\psi|\psi\rangle$ is simply the double layer with no operator inserted between them:

EXAMPLE 3.8 · the norm as a double layer⁹



```
% Formula: <psi|psi> -- the bra sheet contracted onto the
% ket sheet through every physical index.
\begin{tenkz}[lattice={2x2}, planes, bonds=none]
  \tn[at=(2,1,1), label pos=225]{A}
  \tn[at=(1,2,2), label pos=45]{\overline{A}}
  % ket-sheet virtual bonds
  \tnwire{(1,1,1)}{(1,2,1)} \tnwire{(2,1,1)}{(2,2,1)}
  \tnwire{(1,1,1)}{(2,1,1)} \tnwire{(1,2,1)}{(2,2,1)}
  % bra-sheet virtual bonds
  \tnwire{(1,1,2)}{(1,2,2)} \tnwire{(2,1,2)}{(2,2,2)}
  \tnwire{(1,1,2)}{(2,1,2)} \tnwire{(1,2,2)}{(2,2,2)}
  % ket-bra physical pairings
  \tnwire{(1,1,1)}{(1,1,2)} \tnwire{(1,2,1)}{(1,2,2)}
  \tnwire{(2,1,1)}{(2,1,2)} \tnwire{(2,2,1)}{(2,2,2)}
\end{tenkz}
```

$\langle\psi|\psi\rangle$

⁹ Two 2×2 sheets in one frame: A marks the ket sheet, $\overline{A}$ the bra copy above it. Each short diagonal is one ket–bra physical contraction; nothing stays open, and the empty signature says scalar.

kernel-boundary|signature=

The key `planes` provides shorthand for a plane frame possessing a two-member basis: sheet 1 for the ket, and sheet 2 for the bra. Both sheets inhabit a single coordinate frame, and every site is addressed as `(r,c,sheet)`. Because a multi-member basis does not guess your wiring, we specify `bonds=none` and declare each bond explicitly in the body: eight in-sheet virtual bonds, followed by four vertical ket–bra contractions. Notice how cleanly *tikz-tensor-networks* handles these vertical wires: because a member address names a lattice position rather than an explicit port, the frame uses its knowledge of the transverse axis to type the wire as physical automatically.

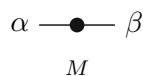
In projection, the two sheets naturally overlap, and several bond lines pass near vertices of the opposite sheet. Rest assured that nothing contracts by accident: geometric coincidence in $\text{T}_{\text{E}}\text{X}$ never creates an edge! The only contractions that exist are the twelve wires explicitly declared in the body.



EXERCISE 3.9 Now let us insert a local operator: the expectation value $\langle\psi|O|\psi\rangle$ places a ring atom O along one of the ket–bra pairing wires. Name that pairing wire, and seat the operator halfway along its length.

3.9 Answers to the exercises

ANSWER 3.1¹⁰

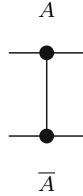


$M_{\alpha\beta}$

¹⁰ Two typed ports with no partners: the matrix keeps two open virtual indices and nothing else.

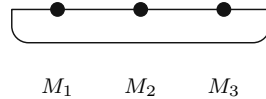
kernel-boundary|signature=open:e,
open:w

```
% Formula: the matrix M with both virtual indices open.
\begin{tenkz}[cols=1]
  \tn[ports={180:virtual:$\alpha$, 0:virtual:$\beta$}]{M}
\end{tenkz}
```

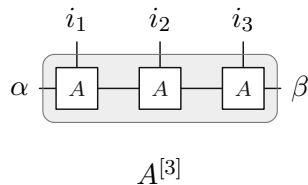
ANSWER 3.2¹¹

```
% Formula: E = sum_i A^i tensor conj(A^i).
% Labels: automatic placement -- every face of the ket atom except the
% north one carries ink, so its name stands there.
\begin{tenkz}[rows={ket, bra}, cols=1, west=open, east=open]
  \tn{A} \\\
  \tn{$\overline{A}$}
\end{tenkz}
```

ANSWER 3.3 A closed wire represents a smooth, self-contained loop, so it has no separate endpoints to specify; every open wire, by contrast, connects exactly two points. Specifying both at once is contradictory, and `tikz-tensor-networks` catches it with the informative diagnostic code [TKZ-LANG-WIRE-ARITY]: *a closed wire takes no positional ends; every other wire takes exactly two*. Use `closed` when you want a free loop, and specify two endpoints when you want a standard bond.

ANSWER 3.4¹²

```
% Formula: tr(M_1 M_2 M_3); sugar: boundary=periodic
% expands to west=trace, east=trace.
\begin{tenkz}[rows={wire}, cols=3, boundary=periodic]
  \tn{$M_1$} & \tn{$M_2$} & \tn{$M_3$}
\end{tenkz}
```

ANSWER 3.5¹³

```
% Formula: the blocked word inside one tinted contour.
\begin{tenkz}[cols=3]
  \tn[skin=box, ports={180:virtual:$\alpha$, 0:virtual, 90:physical:$i_1$}]{A} &
  \tn[skin=box, ports={180:virtual, 0:virtual, 90:physical:$i_2$}]{A} &
  \tn[skin=box, ports={180:virtual, 0:virtual:$\beta$, 90:physical:$i_3$}]{A}
  \tnmark[form=enclosure, tint,
    label pos=270]{(1,1) .. (1,3)}{$A^{[3]}$}
\end{tenkz}
```

$$E = \sum_i A^i \otimes \overline{A^i}$$

¹¹ The frame contracts the ket-bra physical pair; all four virtual ends stay open, so E is a map on the doubled space.

kernel-boundary|signature=open:e,
open:e, open:w, open:w

$$\text{tr}(M_1 M_2 M_3)$$

¹² The trace returns the row to itself; no index is left open, and the empty signature says scalar.

kernel-boundary|signature=

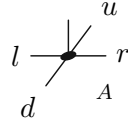
$$A^{[3]}$$

¹³ The enclosure strokes the offset hull of the selection and tint lays ink over the paper inside it; the physical legs pierce the contour, and the signature still does not move.

kernel-boundary|signature=open:e,
open:w, phys:n, phys:n, phys:n

ANSWER 3.6 The audit deliberately refuses unrecognized keywords rather than quietly ignoring them, because a check that fails silently is an audit you believe is running when it is not! The coded error is [TKZ-EQ-CHECK-WORD]: *the audit does not know ‘signature=false’; check= states the audit (signature) and its recorded opt-outs (off={<relation>: <reason>})*. The keyword **signature** names the audit itself; it is not a boolean flag, and therefore takes no value.

ANSWER 3.7 ¹⁴



```
% Formula: the five-index PEPS tensor A^i_{uldr}.
\begin{tenkz}[rows={wire}, cols=1, frame=plane,
  bonds=none, physical=up]
  \tn[ports={0:virtual:$r$, 90:virtual:$u$,
    180:virtual:$l$, 270:virtual:$d$},
    label pos=315]{A}
\end{tenkz}
```

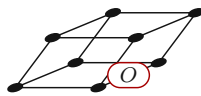
A^i_{uldr}

¹⁴ Four open in-sheet virtual indices and one transverse physical leg: the five-index PEPS tensor. The two receding faces expose their in-sheet bearings in the signature.

```
kernel-boundary|
signature=open:233.130102,
open:53.130102, open:e, open:w,
phys:n
```

ANSWER 3.8 The boundary signature does not change at all, but six new wrap wires appear in the diagram. Because each **trace** closes in-sheet virtual bonds that were never exposed on the perimeter, the boundary signature continues to record nine physical legs. Meanwhile, the network gains one wrap wire per row and per column (**wrap-1** through **wrap-3** and **wrap-col-1** through **wrap-col-3**), closing the flat patch into a topological torus. Always remember: the boundary signature records external connections, not internal topology! The open patch and the periodic state expose identical boundary lists, yet represent fundamentally different networks.

ANSWER 3.9 ¹⁵



```
% Formula: <psi|O|psi> -- O on one site's physical index.
\begin{tenkz}[lattice={2x2}, planes, bonds=none]
  \tnwire{(1,1,1)}{(1,2,1)} \tnwire{(2,1,1)}{(2,2,1)}
  \tnwire{(1,1,1)}{(2,1,1)} \tnwire{(1,2,1)}{(2,2,1)}
  \tnwire{(1,1,2)}{(1,2,2)} \tnwire{(2,1,2)}{(2,2,2)}
  \tnwire{(1,1,2)}{(2,1,2)} \tnwire{(1,2,2)}{(2,2,2)}
  \tnwire{(1,1,1)}{(1,1,2)} \tnwire{(1,2,1)}{(1,2,2)}
  \tnwire{(2,1,1)}{(2,1,2)}
  \tnwire[name=po]{(2,2,1)}{(2,2,2)}
  \tn[skin=ring, at=on po 0.5]{O}
\end{tenkz}
```

$\langle\psi|O|\psi\rangle$

¹⁵ The ring stands halfway along the named ket–bra pairing: a one-site operator between the state and its conjugate. The signature stays empty — the expectation value is a scalar.

```
kernel-boundary|signature=
```

4 Practical Techniques

The tutorial introduced the foundational grammar of **tikz-tensor-networks**; this chapter gathers a collection of practical techniques for the everyday situations you will encounter in research papers. Each section focuses on a single semantic task. When designing a diagram of

your own, start from the tutorial example that most closely matches your mathematical structure, and add only the specific policies your physics requires.

4.1 Open operators versus closed scalars

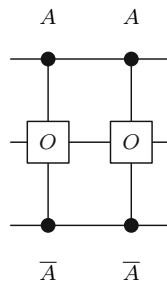
In tensor-network notation, an open perimeter denotes an operator map with free physical or virtual indices, whereas a closed perimeter represents a scalar amplitude (such as a wavefunction overlap $\langle\phi|\psi\rangle$ or a partition function Z). In *tikz-tensor-networks*, a perimeter side exposes indices only when explicitly instructed: an unstated side draws no dangling legs and carries an empty boundary signature. Consequently, an operator map keeps its boundary indices by stating **boundary=open**—or **west=open** for a single side—while **boundary=none** explicitly declares the default closed behaviour. The layout grid places the tensor nodes; it does not assume that every perimeter site carries a dangling index. That is why a quantum circuit pyramid or a single four-legged junction never sprouts unwanted legs.

The rest of the boundary alphabet is equally intuitive: use **boundary=periodic** to trace a row back into itself (periodic boundary conditions), and use **west=cup** or **east=cup** to connect adjacent layers (such as contracting a ket with a bra). Notice that two closures may happen to have the same number of open ends and yet represent entirely different mathematical operations!

4.2 Sandwiching operators: ket, mid, and bra

When calculating local expectation values $\langle\psi|O_1O_2|\psi\rangle$ or sandwiching an MPO between ket and bra states, setting **rows={ket,op,bra}** automatically establishes the three-tier layer geometry: vertical connections contract the physical indices between ket and operator, and between operator and bra, while open horizontal bonds expose the virtual boundary indices of the transfer operator:

EXAMPLE 4.1 · an expectation-value window¹



$$\sum_{s,s'} A^s \otimes O^{ss'} \otimes \bar{A}^{s'}$$

¹ The rows declare their roles.

Vertical connections contract physical indices; six horizontal stubs keep the window a map.

`kernel-boundary|signature=open:e,
open:e, open:e, open:w, open:w,
open:w`

```
% Formula: sum_{s,s'} A^s tensor O^{ss'} tensor conjugate(A^{s'}).
% Labels: automatic placement stands each name on its atom's free face --
% north for the ket row, south for the bra row.
\begin{tenkz}[rows={ket,op,bra}, cols=2, west=open, east=open]
  \tn{A} & \tn{A} \\
  \tn[skin=mpo]{O} & \tn[skin=mpo]{O} \\
  \tn{\overline{A}} & \tn{\overline{A}}
\end{tenkz}
```

4.3 Grouping tensors without altering topology

In coarse-graining schemes (such as block-spin renormalization, DMRG site-blocking, or MERA layers), a bracket mark over a cell range (`\tnmark` with `form=bracket`) declares a brace or measured grouping over existing tensors, while `skin=dots` declares a pass-through ellipsis cell for omitted sites. Neither construct alters the underlying contractions! A wide single tensor node is appropriate only after your formula has already replaced the run by a single blocked tensor.

4.4 Semantic roles versus persistent species

A good diagram distinguishes between structural roles and persistent species. Use `role=` for a tensor's fixed semantic role in an algorithm, such as `operator` or an active site being optimized. By contrast, declare `species=` at setup scope when your paper features a recurring family of tensors (such as isometries W , unitary disentanglers U , or projectors P) whose members should maintain a consistent hue across multiple figures. Always remember: labels distinguish objects of the same kind; color should never be the sole carrier of identity. A role may modify visual style, but never network topology.

4.5 Labels on external legs

An open leg is a wire like any other, and it carries an accessible name. The boundary policy names its legs after the side and the row or column they depart from—`open-west-1`, `open-south-col-3`—while the `physical=` policy names each leg after its face and home cell—`leg-n-1-1` and `leg-s-2-3`. Similarly, a user-authored wire takes whatever name its `\tnwire` declares. You can address a fractional station along that named wire to place a label or bead with complete precision:

```
\begin{tenkz}[rows={wire}, cols=2, west=open, east=open]
  \tn{A} & \tn{B}
  \tnmark[form=label, label pos=90]{on open-west-1 0.5}{\alpha$}
\end{tenkz}
```

The fractional station is measured along the path the wire follows. A leg has one end anchored to a glyph and one free tip; fraction zero is always the end where the wire began. A policy leg runs outward from the glyph, so `on leg-n-1-1 0` lies on the north face of the site in row 1, column 1, while `on leg-n-1-1 1` reaches its outer tip. Declaring a row's legs once with `physical=` is therefore especially convenient: the very same name serves both for line crossings and for label placement.

4.6 Highlighting specific bonds

A lattice draws its bonds automatically from its row and column policies, and every one of them is typeset as a standard black contraction. When a figure needs to highlight one specific edge—an

entanglement cut across a bipartition $A : B$, an SVD Schmidt spectrum cut, or a Hamiltonian interaction term $h_{i,j}$ on a particular link—you can restate that bond directly in the body. An index wire declared between two adjacent cells becomes that pair’s bond: `tikz-tensor-networks` creates no duplicate wire beneath it, and the wire’s species, stroke, and route become the bond’s own:

```
\tndecclare{species}{blocked-edge}{hue=source:red!65!black}
\begin{tenkz}[rows={wire,wire}, cols=3]
  \tn{} & \tn{} & \tn{} \\
  \tn{} & \tn{} & \tn{} \\
  \tnwire[species=blocked-edge]{(1,1)}{(1,2)}
\end{tenkz}
```

Because this restatement represents the same mathematical contraction, the count of internal bonds and the external boundary signature remain unchanged; only the visual ink reflects your styling. You can address the pair by cell coordinates as above, or by typed ports (`a.0` and `b.180`) on named atoms. (Specifying `bonds=none` and redrawing every bond by hand is needlessly tedious and completely unnecessary!) A travelling string is not a restatement: it carries an operator index, so it is drawn cleanly over the lattice without retiring the underlying bonds.

4.7 Directing physical indices

In tensor-network theory, each port carries a defined index type: virtual (mediating quantum entanglement between neighboring tensors) or physical (spanning the local Hilbert space of a site). The `ports=` key types each perimeter face individually, so `0:physical` directs a physical index eastward just as naturally as `0:virtual` directs a virtual one. A single-tensor linear map types its opposing faces accordingly:

```
\begin{tenkz}[rows={wire}, cols=1, bonds=none]
  \tn[at=(1,1), skin=box, ports={180:virtual:V, 0:physical:W}]{A}
\end{tenkz}
```

The `physical=` key is a picture-wide policy, not the sole way to create physical legs: it equips every cell atom with an outward physical port in the frame’s natural direction. Both mechanisms complement each other harmoniously. Picture policy shines on uniform rows: declare it once, and every site receives its physical leg, with any rare exception opting out via `physical=none`. Conversely, on a sparse row where only a single site among many carries a physical index, it is far cleaner to declare the port on that individual atom:

```
\begin{tenkz}[rows={wire}, cols=5]
  \tn{} & \tn{} & \tn[ports={180:virtual, 0:virtual, 90:physical:$i$}]{ } &
  \tn{} & \tn{}
\end{tenkz}
```

4.8 Physical indices in projected sheets

A PEPS tensor has five index directions: four virtual bonds mediating entanglement within the 2D lattice plane, and one physical index projecting out of the plane into the local Hilbert space. The four numeric ports below follow the projected planar axes, while the picture policy supplies the upward transverse direction:

```
\begin{tenkz}[rows={wire,wire,wire}, cols=3,
               frame=plane, bonds=none, physical=up]
  \tn[at=(2,2),
       ports={0:virtual,90:virtual,180:virtual,270:virtual}]{A}
\end{tenkz}
```

Do not attempt to add an artificial fifth numeric angle to straighten the transverse leg! A numeric angle always lives within the plane of the sheet, whereas `physical=up` follows the plane’s independent transverse normal.

Remember also that physical indices belong to lattice sites defined by frame addresses. A mark positioned at a midpoint, along a wire, or at a crossing is a geometric annotation, not a lattice site; hence, picture policies never sprout physical legs from marks. If an annotation represents an additional tensor with its own physical index, simply declare that port explicitly via `ports=`.

4.9 Fusion trees

When working with symmetric tensor networks (such as $SU(2)$ spin-rotation symmetry or abelian $U(1)$ particle conservation), tensors decompose via the Wigner-Eckart theorem into structural Clebsch-Gordan coefficients (represented by fusion trees) and reduced matrix elements. Use `\tntree` when you need to draw parenthesized fusion words and tree contractions. Commutative diagrams, by contrast, compose morphisms between mathematical objects rather than contracting tensor indices; they belong to the excellent `tikz-cd` package, outside the scope of `tikz-tensor-networks`. A `\tntree` atom is a self-contained box that embeds gracefully inside any `tikz-cd` cell whenever the two worlds meet.

4.10 Inline diagrams

A `tikz-tensor-networks` tensor network enters surrounding $\TeX$ as a displayed mathematical term. There is no special “inline” key: inline mathematical alignment belongs to the ambient equation or text scope, not to an internal property of individual diagram records.

5 Catalogue of Constructs

The generated tables in Chapter 6 name every environment, command, and key. Naming is not showing. This chapter draws one specimen per family the tutorial and the recipes leave unpictured,

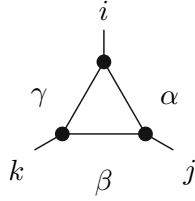
in the order a source is read: the frame first, then the atoms it places, then the connections between them, then the annotations over them, and last the setup that extends the alphabet.

Each specimen is modelled on a benchmark case or a blueprint figure, so what you read is the spelling the corpus writes. Where a family has words the corpus does not write, the entry says so rather than inventing a use.

5.1 Alternative frames

A picture's frame is its coordinate system. `frame=flat` is the default and the whole tutorial rides it; `plane` projects a sheet. The third word is `circle`, which seats the columns of one row at equal angles on a ring and turns each glyph by its station's tangent. The object it draws is a network whose stations stand on a cycle, and the contraction that closes the cycle is written rather than seated.

EXAMPLE 5.1 · three tensors contracted around a cycle¹



```
% Formula: psi_ijk = sum A^i_{gamma alpha} B^j_{alpha beta}
%           C^k_{beta gamma}.
% Ink: the circle frame seats the three sites and carries each
%      physical port out along its station's radius.
\begin{tenkz}[rows={wire}, cols=3, frame=circle, bonds=none]
  \tn[at=(1,1), name=a,
    ports={0:virtual, 180:virtual, 90:physical:$i$}]{ }
  \tn[at=(1,2), name=b,
    ports={0:virtual, 180:virtual, 90:physical:$j$}]{ }
  \tn[at=(1,3), name=c,
    ports={0:virtual, 180:virtual, 90:physical:$k$}]{ }
  \tnbond[name=al]{a.0}{b.180}
  \tnbond[name=be]{b.0}{c.180}
  \tnbond[name=ga]{c.0}{a.180}
  \tnmark[form=label, label pos=e]{on al 0.5}{\alpha$}
  \tnmark[form=label, label pos=s]{on be 0.5}{\beta$}
  \tnmark[form=label, label pos=w]{on ga 0.5}{\gamma$}
\end{tenkz}
```

Three facts of the ring are visible at once. A station's local faces are its own: 0 is the direction the glyph at that station calls east, not the page's east, so the one wire `a.0` to `b.180` means the same thing at every station and the three bonds are written alike. The physical port declared at 90 leaves along the station's outward radius, because the row's north is the ring's outward normal. And the names read level, because a glyph turns and its label does not.

`bonds=none` appears here for the reason it appears in almost every ring in the corpus: the frame's adjacent-cell contraction would join the columns in a line, which on a ring is one bond short of the cycle. Write the cycle's edges and the picture says what it means.

$$\psi_{ijk} = \sum_{\alpha\beta\gamma} A^i_{\gamma\alpha} B^j_{\alpha\beta} C^k_{\beta\gamma}$$

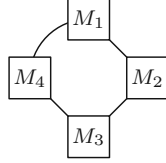
¹ Each station carries one physical spoke, and the three virtual bonds close the triangle. The signature records each spoke at the bearing its station transported it to, not at the face it was declared on.

```
kernel-boundary|
signature=phys:210, phys:330,
phys:n
```



A ring whose closure is the trace itself, with no tensor on the return, is common enough to have a word. `ring= $\langle n \rangle$` is sugar for `rows={wire}, cols= $\langle n \rangle$ , frame=circle, west=trace, east=trace`: the stations of a circle, closed to themselves.

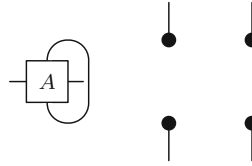
EXAMPLE 5.2 · the traced word, drawn as a ring²



```
% Formula: tr(M_1 M_2 M_3 M_4); the ring word is sugar for
%   rows={wire}, cols=4, frame=circle, west=trace, east=trace.
\begin{tenkz}[ring=4]
  \tn[skin=box]{M_1} & \tn[skin=box]{M_2} &
  \tn[skin=box]{M_3} & \tn[skin=box]{M_4}
\end{tenkz}
```

The four side policies describe the boundary of a rectangular frame. Opening the transverse sides exposes the north and south indices; tracing a side instead joins its boundary sites through the frame. A physical trace is different: it closes the applicable pair of physical ports at each site.

EXAMPLE 5.3 · a physical trace and two open transverse sides³



```
% Ink: trace=physical closes the site's two physical ports.
% north=open and south=open expose the frame boundary.
\begin{tenkz}[rows={wire}, cols=1, bonds=none,
             west=open, east=open, physical=updown,
             trace=physical]
  \tn[skin=box]{A}
\end{tenkz}
\quad
\begin{tenkz}[lattice={2x2}, bonds=none, west=none, east=none,
             north=open, south=open]
  \tn{} & \tn{} \\
  \tn{} & \tn{}
\end{tenkz}
```

Identifying opposite sides of a square gives a torus. A horizontal segment and a vertical segment become closed cycles after this identification. They represent winding classes $(1, 0)$ and $(0, 1)$ and meet transversely once.

$\text{tr}(M_1 M_2 M_3 M_4)$

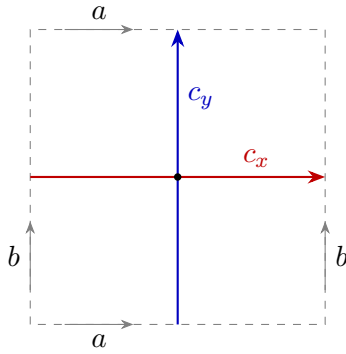
² The trace follows the frame's arc rather than the chord between the two stations it joins. No index survives, so the picture is a scalar. The names are inscribed: a label outside a glyph on a ring is placed toward the centre, where four of them would meet.

kernel-boundary|signature=

$\text{tr}_{\text{phys}}(A)$ and $A: V_S \rightarrow V_N$

³ Left: the upper and lower physical ports close to one another, while the virtual ends remain open. Right: the north and south frame policies expose one virtual index per boundary site. The signature below records the left panel.

kernel-boundary|signature=open:e, open:w

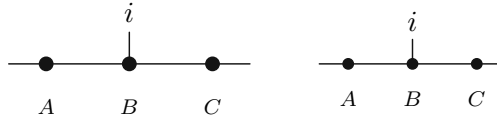


A fundamental square: identify equally labeled edges in the arrow direction. The dot marks the intersection of c_x and c_y , not a tensor insertion.

This topological schematic is drawn with TikZ. The current `wind=` renderer uses a projected curve that does not reliably display the two fundamental cycles; it should not be used as a verified drawing of this construction.

The metric profile rescales every pitch-relative length together. It does not introduce a second table of lengths: `\tnset` changes the single base pitch, and `metrics=` selects the one named ratio applied to it.

EXAMPLE 5.4 · one tensor word at two metric scales ⁴



⁴ The two panels carry the same three-site network. The right panel starts from a larger document pitch and then applies the compact profile; all pitch-relative distances move together.

`kernel-boundary|signature=open:e, open:w, phys:n`

```
% Ink: tnset changes the one base pitch; metrics=compact scales
% every pitch-relative metric from that base.
\begin{tenkz}[rows={wire}, cols=3, west=open, east=open]
  \tn{A} & \tn[ports={90:physical:$i$}]{B} & \tn{C}
\end{tenkz}
\quad
\begin{tenkz}[metrics=compact, rows={wire}, cols=3,
  west=open, east=open]
  \tn{A} & \tn[ports={90:physical:$i$}]{B} & \tn{C}
\end{tenkz}
```

Within an equation, `align=` names the row whose axis is the common mathematical baseline. The policy belongs to the equation and is inherited by its panels, so both sides below align on their operator row.

EXAMPLE 5.5 · equation panels aligned on their second row ⁵

$$\begin{array}{ccc}
 \bullet & & \bullet \\
 A & & A \\
 \boxed{O} & = & \boxed{O} \\
 \bullet & & \bullet \\
 \overline{A} & & \overline{A}
 \end{array}$$

$$AO\overline{A} = AO\overline{A}$$

⁵ The ket, operator, and bra rows have different silhouettes. The equation nevertheless places both panels on the operator axis because `align=2` is shared by the composition.

`kernel-boundary|signature=`

```
% Layout: align=2 makes the operator row the equation baseline.
\begin{tenkzeq}[align=2, check={signature}]
  \begin{tenkz}[rows={ket,op,bra}, cols=1, bonds=none]
    \tn[at=(1,1)]{A} \tn[at=(2,1), skin=mpo]{0}
    \tn[at=(3,1)]{\overline{A}}
  \end{tenkz}
  =
  \begin{tenkz}[rows={ket,op,bra}, cols=1, bonds=none]
    \tn[at=(1,1)]{A} \tn[at=(2,1), skin=mpo]{0}
    \tn[at=(3,1)]{\overline{A}}
  \end{tenkz}
\end{tenkzeq}
```

The remaining frame spelling is the explicit one. `basis=` lists ordered row-kind members at quarter-pitch offsets from the origin, and every cell address then gains a member coordinate: (r,c,k) is the k -th member's copy of cell (r,c) . The `planes` word of the tutorial's double layer is one two-member basis under a name; writing the basis out places the members where the mathematics wants them.

EXAMPLE 5.6 · three basis members in one frame⁶



```
% Ink: an explicit three-member basis; the member index is the
%   third coordinate of every address.
\begin{tenkz}[rows={wire}, cols=1, bonds=none,
  frame={flat, basis={wire at (0,0), wire at (2,0),
    wire at (1,2)}}]
  \tn[at=(1,1,1), skin=box]{A}
  \tn[at=(1,1,2), skin=box]{B}
  \tn[at=(1,1,3), skin=box]{C}
\end{tenkz}
```

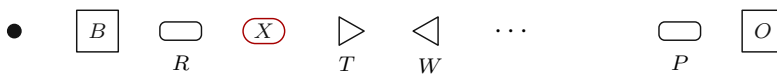
An explicit basis belongs to a picture-level `flat` or `plane` frame. A circle frame refuses one, because its members would need a tangent-offset contract the language does not state, and a group frame refuses one because a basis is picture-scoped. An atom has no frame of its own at all: neither `frame=` nor `basis=` is an atom key, so a site that writes either is turned away as an unknown word rather than accepted and restricted.



5.2 Atom declarations

An atom's silhouette and its type are separate data. `skin=` names the silhouette alone; the stock vocabulary is eight words, and the standard prelude declares two more.

EXAMPLE 5.7 · the stock silhouettes, and the two declared by the prelude⁷



⁶ One cell, three members. The offsets are quarter-pitch steps from the origin member, and each member's copy of the cell is addressed by its index.

⁷ Left to right: dot, box, roundrect, ring, tri, triwest, dots, none, and the prelude's pill and mpo. The eighth cell is empty because `none` draws neither glyph nor label. None of the ten says anything about what a record contracts; that is a declared skin's `pairings=`, which takes ports and appears with the declarations at the end of the chapter.

kernel-boundary|signature=

```
% Ink: one atom per skin; bonds=none, so nothing here contracts.
% The none cell is deliberately blank -- that is what it draws.
\begin{tenkz}[rows={wire}, cols=10, bonds=none]
  \tn[skin=dot]{} & \tn[skin=box]{B} & \tn[skin=roundrect]{R} &
  \tn[skin=ring]{X} & \tn[skin=tri]{T} & \tn[skin=triwest]{W} &
  \tn[skin=dots]{} & \tn[skin=none]{} & \tn[skin=pill]{P} &
  \tn[skin=mpo]{O}
\end{tenkz}
```

Two of these are structural rather than decorative. `dots` is the ellipsis cell of a chain whose middle is elided, and its wires pass through it; `none` draws no glyph at all, which is what a corner junction wants when the wire must turn at a place that carries no tensor. Both appear in the traced word below.

A glyph's extent is `size=`, and a record that stands for a block of records is `cluster=`: an $R \times C$ group of addressable sub-atoms named after the host.

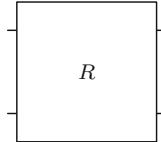
EXAMPLE 5.8 · three size classes, and one clustered site⁸



```
% Ink: size= chooses a glyph's extent from the metric table;
% cluster= expands one declaration into an addressable group.
\begin{tenkz}[rows={wire}, cols=4, bonds=none]
  \tn[at=(1,1), skin=dot, size=s]{} &
  \tn[at=(1,2), skin=dot, size=m]{} &
  \tn[at=(1,3), skin=dot, size=l]{} &
  \tn[at=(1,4), name=q, cluster=2x2]{}
\end{tenkz}
```

The two span counts belong to the atom rather than to the frame. `wide=` counts columns and `wires=` counts rows; together they place one tensor over the rectangular block of sites it represents.

EXAMPLE 5.9 · one tensor spanning two rows and two columns⁹



```
% Ink: wide=2 and wires=2 give one atom a two-by-two host rectangle.
\begin{tenkz}[rows={wire,wire}, cols=2, bonds=none]
  \tn[at=(1,1), skin=box, wide=2, wires=2, name=R,
    ports={180@1:virtual, 180@2:virtual,
      0@1:virtual, 0@2:virtual}]{R}
  \tnwire{open w}{R.180@1} \tnwire{open w}{R.180@2}
  \tnwire{R.0@1}{open e} \tnwire{R.0@2}{open e}
\end{tenkz}
```

A declared skin may carry more than one internal pairing. When two such routes meet, `pairing cross=` identifies the pairing instance and fixes its order at the crossing, just as `cross=` does for an authored wire.

EXAMPLE 5.10 · two declared pairings with a chosen crossing order¹⁰



⁸ The first three are one skin at the three metric classes. The fourth is a single declaration standing for four addressable sub-atoms, reached as q-1-1 through q-2-2.

kernel-boundary|signature=

$$R: V^{\otimes 2} \longrightarrow V^{\otimes 2}$$

⁹ The rectangle is one atom, not four coincident sites. Its slotted west and east ports expose the two virtual indices carried by the rows.

kernel-boundary|signature=open:e, open:e, open:w, open:w

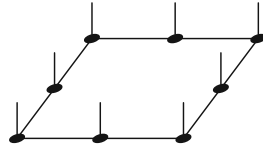
¹⁰ The horizontal and vertical routes belong to their skins. The horizontal host declares that its first pairing passes under the first pairing of the vertical host.

kernel-boundary|signature=

```
% Ink: pairing cross= orders two skin-owned routes
%   at their intersection.
\tndecclare{species}{warm}{hue=source:red}
\tndecclare{species}{cool}{hue=source:blue}
\tndecclare{skin}{crossed-horizontal}{
  base=box, pairings={w@1 > e@1 : warm}}
\tndecclare{skin}{crossed-vertical}{
  base=box, pairings={n@1 > s@1 : cool}}
\begin{tenkz}[size=1, rows={wire}, cols=1, bonds=none]
  \tn[at=(1,1), name=H, skin=crossed-horizontal, size=1,
    pairing cross=
      1: under at crossing of self and pairing 1 of V]]{}
  \tn[at=0 e of H, name=V, skin=crossed-vertical, size=1]{}
\end{tenkz}
```

A site that is not there is `void=`. The two words differ in how much they take away. `sealed` removes the site, the bonds that reached it, and the leg the row's physical policy would have given it. `open` removes the glyph alone: the bonds still meet at the station and the leg still leaves it, so the picture loses a tensor's ink and not one entry of its boundary. The glyph goes because a void site's silhouette falls to `none`, and that is a default rather than an override: a site that names its own `skin=` draws it.

EXAMPLE 5.11 · a sealed site in a sheet ¹¹



```
% Ink: void=sealed removes the site and the bonds that reached
%   it; void=open would keep both and drop only the glyph.
\begin{tenkz}[lattice={3x3}, frame=plane, physical=up]
  \tn{} & \tn{} & \tn{} \\
  \tn{} & \tn[void=sealed]{} & \tn{} \\
  \tn{} & \tn{} & \tn{}
\end{tenkz}
```

EXERCISE 5.1 The eight physical legs above are the eight surviving sites. How many entries does the signature carry when the centre is written `void=open` instead, and which of them are virtual?

5.3 Connections

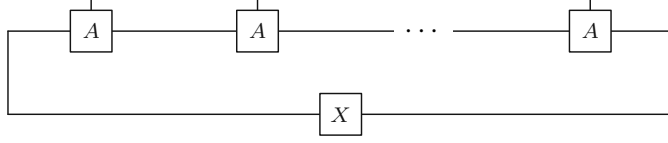
Every connection is a `\tnwire`. Its `kind=` says which kind: `index` is a contraction, the default, and `\tnbond` is its one-word spelling; `string` travels over the picture carrying an operator index; `pairing` is a declared skin's own internal route, and is never authored at a call site.

The path a wire takes is `route=`. `straight` is the default and joins the two ends directly. A rectangular periodic return uses straight segments and four corner junctions, as below. Each junction pairs two virtual ports and carries no tensor.

$|\psi\rangle$ with one site removed

¹¹ The centre site and its four bonds are gone, and the sheet's remaining sites keep their physical legs. A sealed hole exposes nothing, so the signature loses exactly the removed site's leg.

```
kernel-boundary|signature=phys:n,
phys:n, phys:n, phys:n, phys:n,
phys:n, phys:n, phys:n
```

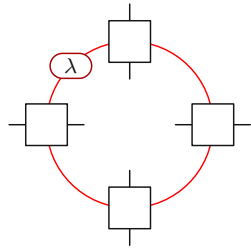
EXAMPLE 5.12 · a periodic word with a symmetric return ¹²

```
% Formula:  $|\psi(X)\rangle = \sum \text{tr}(X A^{s_1} \dots A^{s_N}) |s\rangle$ .
% Ink: one virtual loop with four tensor-free corners.
% X is centered below the uniformly spaced tensor row.
\begin{tenkz}[rows={wire,wire}, cols=9, bonds=none]
  \tn[at=(1,2), skin=box, name=a1,
    ports={0:virtual, 180:virtual, 90:physical}]{A}
  \tn[at=(1,4), skin=box, name=a2,
    ports={0:virtual, 180:virtual, 90:physical}]{A}
  \tn[at=(1,6), skin=dots, name=d,
    ports={0:virtual, 180:virtual}]{\dots}
  \tn[at=(1,8), skin=box, name=a4,
    ports={0:virtual, 180:virtual, 90:physical}]{A}
  \tn[at=(2,5), skin=box, name=x,
    ports={0:virtual, 180:virtual}]{X}
  \tnwire{a1.0}{a2.180} \tnwire{a2.0}{d.180}
  \tnwire{d.0}{a4.180}
  \tn[at=(1,1), skin=none, name=nw, ports={0:virtual,270:virtual}]{\nwarrow}
  \tn[at=(2,1), skin=none, name=sw, ports={0:virtual,90:virtual}]{\swarrow}
  \tn[at=(1,9), skin=none, name=ne, ports={180:virtual,270:virtual}]{\nearrow}
  \tn[at=(2,9), skin=none, name=se, ports={180:virtual,90:virtual}]{\searrow}
  \tnwire{a4.0}{ne.180} \tnwire{ne.270}{se.90}
  \tnwire{se.180}{x.0} \tnwire{x.180}{sw.0}
  \tnwire{sw.90}{nw.270} \tnwire{nw.0}{a1.180}
\end{tenkz}
```

The corner junctions make the two sides of the return equal. Every segment is a contraction, so all virtual indices close through X and only the three physical indices remain open.

Use `arc` for a curved connection: it leaves and enters along its ends' faces. When neither end names a face, it follows the straight chord. The next example uses four arcs to close a plaquette.

The same family draws a closed cycle of tensors laid out as a plaquette, and a matrix inserted on one of its edges is an ordinary ring atom addressed by a fraction along the named wire.

EXAMPLE 5.13 · four arcs, and a bead on one of them ¹³

```
% Ink: four arcs close the plaquette; at=on <wire> <fraction>
%   stands the inserted matrix halfway along one of them.
\nddeclare{species}{op}{hue=source:red}
\begin{tenkz}[rows={wire,wire,wire}, cols=5, bonds=none]
  \tn[skin=mpo, at=(1,3), name=N,
    ports={0:virtual, 90:virtual, 180:virtual, 270:virtual}]{\text{N}}
  \tn[skin=mpo, at=(2,4), name=E,
    ports={0:virtual, 90:virtual, 180:virtual, 270:virtual}]{\text{E}}
```

$$|\psi(X)\rangle = \sum_{s_1 \dots s_N} \text{tr}(X A^{s_1} \dots A^{s_N}) |s\rangle$$

¹² The four corner junctions carry no tensor, and the elision cell passes its wires through. Every virtual index closes through X ; each drawn site keeps its physical leg.

kernel-boundary|signature=phys:n, phys:n

λ on one edge

¹³ The four operator-species arcs form the closed ring; the ring atom stands halfway along the named north-west arc. Each tensor spends two of its four faces on the ring and leaves two open, so eight virtual ends reach the boundary.

kernel-boundary|signature=open:e, open:e, open:n, open:n, open:s, open:s, open:w, open:w

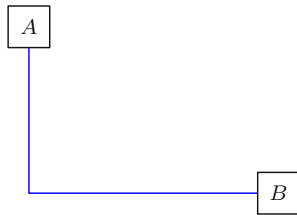
```

\tn[skin=mpo, at=(3,3), name=S,
  ports={0:virtual, 90:virtual, 180:virtual, 270:virtual}]{}
\tn[skin=mpo, at=(2,2), name=W,
  ports={0:virtual, 90:virtual, 180:virtual, 270:virtual}]{}
\tnwire[route=arc, species=op, name=lam]{W.90}{N.180}
\tnwire[route=arc, species=op]{N.0}{E.90}
\tnwire[route=arc, species=op]{E.270}{S.0}
\tnwire[route=arc, species=op]{S.180}{W.270}
\tn[skin=ring, at=on lam 0.5]{\lambda}
\end{tenkz}

```

The third route word is **orth**, which turns at right angles. It is the spelling for a string that must reach around the picture rather than across it, and it takes its corners from **via=**, a list of addresses the path passes through.

EXAMPLE 5.14 · a string routed around the picture¹⁴



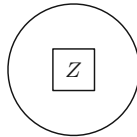
```

% Ink: route=orth turns at right angles; via= names the cells
%   the path passes through.
\tndeclare{species}{gauge}{hue=source:blue}
\begin{tenkz}[rows={wire,wire,wire}, cols=4, bonds=none]
  \tn[at=(1,1), name=a, skin=box]{A}
  \tn[at=(3,4), name=b, skin=box]{B}
  \tnwire[kind=string, species=gauge, route=orth,
    via={(3,1)}]{a.270}{b.180}
\end{tenkz}

```

A string that returns to itself is **closed**, and a closed wire takes no positional ends at all: it is a cycle, so it has none to give. Its shape comes from its route: a loop that rings a selection travels the hull, **route={all of z}**, and a freer shape names its waypoints with **via=**.

EXAMPLE 5.15 · a closed string around a site¹⁵



```

% Ink: a closed string takes no ends; its route rings the named selection.
\begin{tenkz}[rows={wire}, cols=2, bonds=none]
  \tn[at=(1,1), name=z, skin=box]{Z}
  \tnwire[kind=string, closed, route={all of z}]
\end{tenkz}

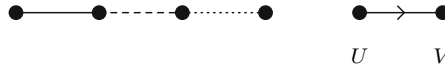
```

Three further wire keys decorate the rail rather than change its path. **stroke=** is the rail's pattern, and the corpus draws only three: **solid** for a contraction, **dashed** for a line the figure shows but its contraction does not use, **dotted** for a lattice lying under the sheet being drawn. Those are readings, and the pattern is

¹⁴ The waypoint is an ordinary cell address. The path leaves *A* southward, turns at the named cell, and enters *B* from the west.
kernel-boundary|signature=

¹⁵ The loop carries an operator index and contracts nothing. It exposes no end, so it adds no entry to the signature.
kernel-boundary|signature=

ink alone: a dashed wire keeps the default `kind=index`, and what decides whether a drawn line contracts is what its two ends are. A wire between two ports takes them both, whatever its stroke; a wire between two atoms named as wholes leaves their ports where they were. That is how the dashed diagonals of the benchmark's dual lattice recall an edge without contracting one: their ends are junctions with no ports to take. `dir=` marks a directed index against its dual. Multiplicity is mathematics carried by the index label, not a second field on the rail.

EXAMPLE 5.16 · the three strokes and a direction¹⁶

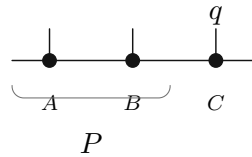
```
% Ink: stroke= is the rail's pattern; dir= marks the direction.
\begin{tenkz}[rows={wire}, cols=4, bonds=none]
  \tn[at=(1,1), name=p]{} \tn[at=(1,2), name=q]{}
  \tn[at=(1,3), name=r]{} \tn[at=(1,4), name=s]{}
  \tnwire[stroke=solid]{p.0}{q.180}
  \tnwire[stroke=dashed]{q.0}{r.180}
  \tnwire[stroke=dotted]{r.0}{s.180}
\end{tenkz}
\qqquad
\begin{tenkz}[rows={wire}, cols=2, bonds=none]
  \tn[at=(1,1), name=u]{U} \tn[at=(1,2), name=v]{V}
  \tnwire[dir=to]{u.0}{v.180}
\end{tenkz}
```

¹⁶ Left: the three declared rail patterns. The four sites declare no ports, so the three rails differ in pattern and in nothing else. Right: one rail carrying the direction mark that distinguishes a space from its dual.

kernel-boundary|signature=

5.4 Annotations

A mark speaks over a picture and owns none of it. Its `form=` is the shape of the speech, and the alphabet is four words. Three put ink on the page: `label` stands text at a target, `bracket` measures a span from outside it, and `enclosure` strokes the offset hull of a selection. The fourth, `prose`, is accepted and recorded without ink. This section draws the three, shows what a spelling the alphabet has lost answers now, and reads `prose` out of the record stream.

EXAMPLE 5.17 · a bracket over a span, a label on a site¹⁷

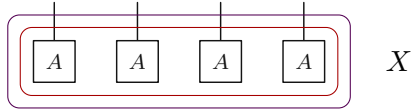
```
% Ink: two mark forms over one row; no contraction changes.
\begin{tenkz}[rows={wire}, cols=3, west=open, east=open,
  physical=up]
  \tn{A} & \tn{B} & \tn{C}
  \tnmark[form=bracket]{(1,1) .. (1,2)}{P}
  \tnmark[form=label, label pos=n]{(1,3)}{q}
\end{tenkz}
```

Contours nest by themselves. A second enclosure over one selection stands one clearance outside the first because it was declared second, which is how a figure draws a boundary operator acting on the open indices of a region it also has to show.

¹⁷ The bracket names the range of two cells; the label names one. Neither touches the contraction, and the signature is the row's.

kernel-boundary|signature=open:e,
open:w, phys:n, phys:n, phys:n

EXAMPLE 5.18 · a region, and an operator on its boundary ¹⁸



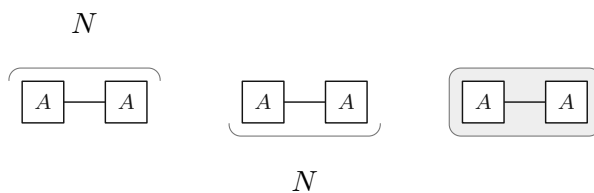
```
% Ink: two semantically tinted contours over one selection;
%   concentric order steps the second declared clear of the first.
\begin{tenkz}[rows={wire}, cols=4, bonds=none, physical=up]
  \tn[at=(1,1), skin=box]{A} & \tn[at=(1,2), skin=box]{A} &
  \tn[at=(1,3), skin=box]{A} & \tn[at=(1,4), skin=box]{A}
  \tnmark[form=enclosure, species=selected]{(1,1) .. (1,4)}{ }
  \tnmark[form=enclosure, species=collar, label pos=e]
    {(1,1) .. (1,4)}{X$}
\end{tenkz}
```

Nothing in the source states that order. Contours are ranked by containment, ties going to the earlier declaration, and each rank stands one clearance further out than the rank inside it. A bracket drawn inside a region window is ranked the same way and clears it instead of sharing its support.

species= tints the contour, as it tints every other record: the five region words – **selected**, **secondary**, **complement**, **collar**, **neutral** – are prelude species carrying the house region palette, and **complement** also dashes the contour it names, the grey rest of a picture under discussion. A contour naming no species is passive. **tint** lays ink over the paper inside a contour rather than only stroking it; the tutorial’s blocked run shows the tinted form, and so does the band below.

Use **bracket** to mark a span. Its **label pos=** names the side: 90 places it above and the default places it below. Use **enclosure** with **tint** for a shaded region around a selection.

EXAMPLE 5.19 · brackets above and below, and a tinted enclosure ¹⁹



```
% Ink: brackets on two sides and one tinted enclosure.
\begin{tenkz}[rows={wire}, cols=2]
  \tn[skin=box]{A} & \tn[skin=box]{A}
  \tnmark[form=bracket, label pos=90]{(1,1) .. (1,2)}{N$}
\end{tenkz}
\quad
\begin{tenkz}[rows={wire}, cols=2]
  \tn[skin=box]{A} & \tn[skin=box]{A}
  \tnmark[form=bracket]{(1,1) .. (1,2)}{N$}
\end{tenkz}
\quad
\begin{tenkz}[rows={wire}, cols=2]
  \tn[skin=box]{A} & \tn[skin=box]{A}
  \tnmark[form=enclosure, tint]{(1,1) .. (1,2)}{ }
\end{tenkz}
```

$|\psi(X)\rangle$

¹⁸ The inner contour is the retained block; the outer one, declared second and therefore one clearance further out, is the operator acting on the indices the block leaves open. Their region species distinguish the selected block from its collar by the house palette.

`kernel-boundary|signature=phys:n, phys:n, phys:n`

¹⁹ One bracket named on two sides, and one tinted enclosure. A bracket’s arc follows its name, so **label pos=90** carries both to the north and the default carries both to the south. The enclosure’s **tint** shades the selected region.

`kernel-boundary|signature=`

`prose` is the one form that is accepted without ink, and it is accepted because it asks for none. `\tnprose` states, in the picture's own record stream, that a step of an argument was not drawn. Nothing appears on the page and one line appears in the event stream:

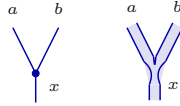
```
mark[id=mark-1|form=prose|label=by isometry|target=panel
```

On a checked relation that line is a hard error without a recorded opt-out, because `prose` is not a boundary signature and cannot be compared with one.

5.5 Fusion

Two atoms are structured rather than atomic. `\tntree` takes a parenthesized fusion word and draws the tree that parses it; the strands are drawn as lines by default, and `tree style=ribbon` draws the same parsed tree as a regular ribbon neighbourhood, one band per simple object.

EXAMPLE 5.20 · one fusion tree in both strand styles²⁰

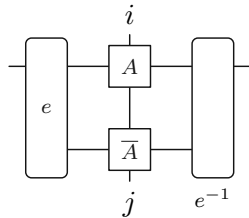


```
% Ink: one parsed fusion word, drawn in both strand styles.
\nddeclare{species}{anyon}{}
$\tntree[tree style=wire, species=anyon]{(a\,b)_x}
\quad
\tntree[tree style=ribbon, species=anyon]{(a\,b)_x}$
```

A tree is a self-contained box and stands wherever mathematics stands: in a display, in an alignment cell, in a `tikz-cd` cell. It is the one construct here that does not need a picture around it.

`\tnfuse` is the other. It is a fusion atom, a triangle whose split side the frame bonds to the rows it spans, and it is how a doubled-layer object is written as one tensor.

EXAMPLE 5.21 · a fuser, the doubled layer, and its inverse²¹



```
% Formula: M^{ij} = \sum_k e (A^{(i,k)} \otimes \overline{A}^{(j,k)}) e^{-1}
% conj(A)^{(j,k)} e^{-1}; the fusers combine the ket and bra
% virtual indices into one, and the bond between the rows is
% the ancillary index k.
\begin{tenkz}[rows={op,op}, cols=3]
\tntfuse[at=(1,1), name=e, skin=pill]{e}
\tn[at=(1,2), skin=box, ports={90:physical:$i$}]{A}
\tn[at=(2,2), skin=box, ports={270:physical:$j$}]{\overline{A}}
```

$$m_{ab}^x: a \otimes b \rightarrow x$$

²⁰ Both pictures are the same chosen fusion morphism. The wire style meets three lines at one trivalent vertex; the ribbon style gives each simple object a band and joins them at a vertex patch.

$$M^{ij} = \sum_k e (A^{(i,k)} \otimes \overline{A}^{(j,k)}) e^{-1}$$

²¹ The two fusion atoms span both rows and combine the ket and bra virtual indices into one; between them the two layers carry the physical indices i and j . The frame's bond between the rows is the ancillary index k the sum runs over.

kernel-boundary|signature=open:e,
open:w, phys:n, phys:s

```
\tnfuse[at=(1,3), name=ei, skin=pill]{e^{-1}}
\tnwire{open w}{e.180} \tnwire{ei.0}{open e}
\end{tenkz}
```

`\tnfuse` is sugar for `\tn[skin=tri, wires=2, ...]`, and authored keys follow the preset, so the `skin=pill` above overrides the triangle the preset would have drawn. Under `\tnset{strict}` the sugar is refused and the kernel spelling is required.

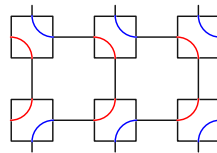


5.6 Setup and extension

`\tndeclare` is the one extension door, and it has three classes. A *species* is a named hue a paper's family of objects keeps across figures. A *skin* is a base silhouette plus `pairings=`, its own internal routes between its own ports. An *atom* is a new typed spelling in the atom class.

The skin class is the one that changes what a glyph means. A tensor whose two port pairs are joined inside the box is drawn once as a declaration and used as a word:

EXAMPLE 5.22 · two declared skins, each with its own internal routes²²

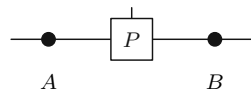


```
% Ink: a declared skin is a base silhouette plus its own
% internal port pairings, tinted by declared species.
\tndeclare{species}{right}{hue=source:blue}
\tndeclare{species}{left}{hue=source:red}
\tndeclare{skin}{shift-right}{
  base=box, pairings={n@1 > e@1 : right, s@1 > w@1 : left}}
\tndeclare{skin}{shift-left}{
  base=box, pairings={n@1 > w@1 : left, s@1 > e@1 : right}}
\begin{tenkz}[rows={wire,wire}, cols=3, physical=updown]
  \tn[skin=shift-right]{} & \tn[skin=shift-right]{} &
  \tn[skin=shift-right]{} & \
  \tn[skin=shift-left]{} & \tn[skin=shift-left]{} &
  \tn[skin=shift-left]{}
\end{tenkz}
```

A pairing is a wire whose two ends are the skin's own ports, so it is reusable topology: declare it once and every site drawn with that skin carries it. `base=` may name another declared skin, so a family of related glyphs is written as a chain of small declarations rather than one long one. What the chain carries down is the silhouette alone: a derived skin resolves its base to a primitive and inherits none of its base's pairings, so a route the derived glyph wants is written again in its own declaration.

The atom class has a two-argument spelling, `\tndeclareatom`, for the common case of a new command with a skin and a port list.

EXAMPLE 5.23 · a declared atom command²³



$$U = U_{\rightarrow} U_{\leftarrow}$$

²² The upper row turns its north input east and its south input west; the lower row reverses both. The pairings belong to the skin, so every site drawn with it carries them. The row policy grows a leg on both faces of every site, and the frame's bond between the rows takes the inner one at each, so six legs reach the boundary and not twelve.

kernel-boundary|signature=phys:n, phys:n, phys:s, phys:s, phys:s

²³ The declaration adds one word to the atom class; the picture then spells it like any other atom. The port list is complete: an anchor without a type is not an extension contract.

kernel-boundary|signature=open:e, open:w, phys:n

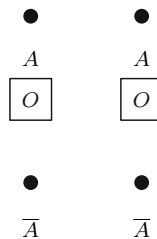
```
% Ink: tndeclareatom names faces west|east:virtual and
%   up|down:physical; tndeclare{atom} also accepts angles.
\tndeclareatom{\tnprojector}{skin=box,
  ports={west:virtual, east:virtual, up:physical}}
\begin{tenkz}[rows={wire}, cols=3, west=open, east=open]
  \tn{A} & \tnprojector{P} & \tn{B}
\end{tenkz}
```

The two spellings do not accept the same vocabulary, in ports or in skins. `\tndeclareatom` takes the four named faces only — `west`, `east` typed `virtual`, and `up`, `down` typed `physical` — and answers anything else with [TKZ-ATOM-INVALID-PORT]. Its `skin=` is four words as well, `dot`, `box`, `pill` and `mpo`; a fifth is refused as an unknown choice and the declaration falls back to `dot`. The three-argument `\tndeclare{atom}` also takes compass letters, numeric angles, and every silhouette, so a glyph with an off-axis face or a `ring` skin is declared there.

A declaration is setup, and setup happens outside a composition form. Writing one among an equation's panels is refused: it would be a document-wide act performed in the middle of an assertion, ordered by where it happens to sit and read once for each time the equation is measured. Write it before the equation, where it says the same thing once.

`\tnset` carries the same policy at document scope. Two keys live there: `pitch=`, the one base metric every named ratio follows, and `strict`, which rejects every sugar spelling in the registry. The house semantic ink binds when the package loads. A palette taken from a cited source reaches ink through a species declaration with `hue=`. No benchmark case sets `strict`. The canonical cases write sugar freely, and `physical=` alone stands in fifty-one of the hundred and thirty of them; the sources that turn `strict` on are the five refusal fixtures that prove it refuses, together with the refusal below, which is a sixth of the same kind. What the checks hold is the sugar itself: eleven sugar spellings compile beside their kernel expansions and must emit the same events and the same picture, so a sugar word is a shorter way to write a canonical case and never a second meaning.

EXAMPLE 5.24 · the sandwich sugar and its strict kernel spelling²⁴



$\langle A|O|A \rangle$

²⁴ Both panels draw the same ket-operator-bra stack. The left uses the `sandwich` preset. In the locally strict panel on the right, the three rows are written explicitly and therefore need no sugar.

`kernel-boundary|signature=`

```
% Ink: sandwich expands to rows={ket,op,bra}; strict accepts the
%   explicit kernel spelling and would refuse the sugar word.
\begin{tenkz}[sandwich, cols=1, bonds=none]
  \tn[at=(1,1)]{A} \tn[at=(2,1), skin=mpo]{O}
  \tn[at=(3,1)]{\overline{A}}
\end{tenkz}
\quad
{\tnset{strict}}
```

```
\begin{tenkz}[rows={ket,op,bra}, cols=1, bonds=none]
  \tn[at=(1,1)]{A} \tn[at=(2,1), skin=mpo]{0}
  \tn[at=(3,1)]{\overline{A}}
\end{tenkz}
```

A DELIBERATE REFUSAL 5.1 · strict setup rejects a sugar spelling²⁵

```
% Refused: strict rejects sugar. The kernel spelling of this
% picture is rows={ket,op,bra}.
\tnset{strict}
\begin{tenkz}[sandwich, cols=1]
  \tn{A} \ \ \tn[skin=mpo]{0} \ \ \tn{\overline{A}}
\end{tenkz}
```

[TKZ-LANG-STRICT] 'sandwich' is sugar and \tnset{strict} is active.

One composition word closes the alphabet. `\tngroup` transforms a sub-diagram as one object, and the records inside it keep the names they were given, so the rest of the picture can still address them.

EXAMPLE 5.25 · a grouped sub-diagram²⁶



```
% Ink: a group is a sub-frame of its picture, and the names
% declared inside it stay addressable outside it.
\begin{tenkz}[rows={ket}, cols=1, bonds=none]
  \tngroup[frame=flat]{ \tn[skin=box, name=g]{G} }
\end{tenkz}
```

A group's options are picture keys, and `frame=` is the one the specimen writes, a sub-frame being what a group is for. A group shares its picture's metric, size class, and audit, so the three keys that would set those over again are refused at group scope rather than accepted and left inert: `size=`, `metrics=`, and `check=`. A `basis=` inside a group's frame is refused with them.

5.7 Answers to the exercises

ANSWER 5.1 Nine entries, none of them virtual, which is the signature of the sheet with no hole in it at all. `open` takes away the glyph and nothing else: the four bonds still meet at the empty station, and the centre still carries the leg its row's policy gives it, so a reader who wants the eight-leg boundary above must write `sealed`. Neither word turns a bond into an exposed index. What `open` buys is the picture: a station where the sheet contracts and no tensor is drawn.

6 Summary of the Grammar

The executable language registry generates this chapter. It inventories environments, commands, keys, scopes, value types, defaults, diagnostics, and standalone examples. The registry also drives parser registration, linting, and the public-symbol census.

²⁵ `sandwich` is sugar for `rows={ket,op,bra}`. Under `strict` the expansion is not performed; the spelling is refused and named.

²⁶ The group is one object of its picture. Its member keeps the name `g` and is reachable from outside the group by that name. `kernel-boundary|signature=`

The tables name the vocabulary; they draw none of it. For a construct's picture, read Chapter 5, which sets one drawn specimen per family beside the source that produced it.

Command	Class	Why this is a command
<code>\tn</code>	atoms	An atom changes the object record, so it is a command rather than a key.
<code>\tn [object keys]{label}</code>		
Example: <code>tex/tenkz/examples/api/tn.tex</code> . Representative failure: An unknown object key is a hard error.		
<code>\tnfuse</code>	atoms	A fusion atom changes wire arity.
<code>\tnfuse [object keys]{label}</code>		
Example: <code>tex/tenkz/examples/api/tnfuse.tex</code> . Representative failure: A nonpositive span is a hard error.		
<code>\tntree</code>	atoms	A fusion tree is one structured atom.
<code>\tntree [keys]{word}</code>		
Example: <code>tex/tenkz/examples/api/tntree.tex</code> . Representative failure: An unbalanced fusion word is a hard error.		
<code>\tnset</code>	setup-extension	Setup changes document or group policy outside any one picture.
<code>\tnset {setup keys}</code>		
Example: <code>tex/tenkz/examples/api/tnset.tex</code> . Representative failure: An unknown setup key or enum value is a hard error.		
<code>\tndeclareatom</code>	setup-extension	A declaration adds a typed atom spelling to the grammar. The four face words are its whole port vocabulary; angles, slots, and port labels belong to the three-argument door.
<code>\tndeclareatom {\command }{skin=dot box pill mpo, ports={west east:virtual, up down:physical}}</code>		
Example: <code>tex/tenkz/examples/api/tndeclareatom.tex</code> . Representative failure: An unknown face, type, skin, or duplicate command is a hard error.		
<code>\tnwire</code>	connections-closures	A wire is one typed index line; bonds and strings are its kinds.
<code>\tnwire [wire keys]{end}{end}</code>		
Example: <code>tex/tenkz/examples/api/tnwire.tex</code> . Representative failure: A closed wire with positional ends is a hard error.		
<code>\tnmark</code>	regions-annotations	A mark annotates without owning topology.
<code>\tnmark [mark keys]{target}{label}</code>		
Example: <code>tex/tenkz/examples/api/tnmark.tex</code> . Representative failure: An unknown mark form is a hard error.		
<code>\tngroup</code>	composition	A group transforms a sub-diagram as one object.
<code>\tngroup [picture keys]{body}</code>		
Example: <code>tex/tenkz/examples/api/tngroup.tex</code> . Representative failure: An unknown picture key is a hard error.		
<code>\tndeclare</code>	setup-extension	The one extension door: atom, skin, or species.
<code>\tndeclare {class}{name}{keys}</code>		
Example: <code>tex/tenkz/examples/api/tndeclare.tex</code> . Representative failure: An unknown declaration class is a hard error.		
<code>\tnbond</code>	connections-closures	Sugar: a bond is a wire of kind index.
<code>\tnbond [wire keys]{end}{end}</code>		

Example: `tex/tenkz/examples/api/tnbond.tex`. Representative failure:
An unknown wire key is a hard error.

`\tnprose` regions- Sugar: prose in place of a
 annotations diagram. The text enters the
 record stream and no ink reaches
 the page.

`\tnprose {text}`

Example: `tex/tenkz/examples/api/tnprose.tex`. Representative failure:
Prose on a checked relation without an opt-out is a hard error.

Scope	Key	Type	Default	Meaning
object	role	semantic-role	empty	Semantic theme role.
object	species	declared-name	empty	Semantic species.
object	tree style	enum(wire ribbon)	wire	Fusion-tree strand style.
atom-declaration	skin	silhouette-name	dot	Declared atom silhouette.
atom-declaration	ports	typed-port-list	empty	Declared atom ports.
kernel-picture	rows	row-list	wire	Chain row topology; non-wire rows populate.
kernel-picture	cols	positive-integer	3	Cells per row.
kernel-picture	frame	frame-spec	flat	Closed flat, plane, or circle frame word.
kernel-frame	basis	basis-spec	origin singleton	Ordered row-kind members at east/north quarter-pitch offsets.
kernel-picture	west	side-policy	none	West side policy.
kernel-picture	east	side-policy	none	East side policy.
kernel-picture	north	side-policy	none	North side policy.
kernel-picture	south	side-policy	none	South side policy.
kernel-picture	trace	trace-spec	empty	Cells or physical ports closed to themselves.
kernel-picture	open	cell-set	empty	Cells opened by declaration.
kernel-picture	bonds	bond-policy	grid	Frame-generated adjacent contraction.
kernel-picture	align	row	midline	The wire axis panels align on.
kernel-picture	size	size-class	from math style	Size class of the metric context.

kernel-picture	metrics	enum(compact)	empty	Named metric profile of the picture's pitch.
kernel-picture	check	check-spec	signature	The tenkzeq audit: selection, equivalence, recorded opt-outs.
kernel-picture	sandwich	flag	false	Ket-over-bra row preset.
kernel-picture	boundary	enum(open none periodic)	none	All-side boundary word.
kernel-picture	lattice	pair	empty	RxC wire-row lattice preset.
kernel-picture	ring	integer	empty	Circle-frame ring preset.
kernel-picture	surface	enum(torus)	empty	Torus closure preset.
kernel-picture	planes	flag	false	Plane frame with ket and bra basis members.
kernel-picture	physical	physical-policy	none	Expander: frame-owned outward physical port per frame-cell atom; geometric overlays are excluded.
kernel-atom	skin	silhouette-name	theme default	Declared silhouette.
kernel-atom	wide	positive-integer	1	Cells spanned.
kernel-atom	wires	positive-integer	1	Wire rows spanned.
kernel-atom	at	address	next chain cell	Placement address.
kernel-atom	name	identifier	none	Addressable name; an unnamed atom answers only to its cell.
kernel-atom	ports	typed-port-list	from skin	Typed faces and slots.
kernel-atom	species	declared-name	empty	Semantic species.
kernel-atom	pairing cross	indexed-crossing-list	empty	Per-instance crossings of generated skin pairings.
kernel-atom	size	size-class	m	Per-atom metric size class.
kernel-atom	label pos	angle	auto	Label quadrant; auto takes the first free face in the order s, n, e, w, and south when every face carries ink.

kernel-atom	void	void-policy	unset	Hole or removed site.
kernel-atom	physical	physical-policy	unset	Site's own answer to the picture's physical policy.
kernel-atom	cluster	pair	empty	Expander: RxC addressable sub-atom group.
kernel-atom	role	semantic-role	empty	Prelude species spelling.
kernel-wire	kind	enum(index string)	index	Bond or travelling string; a declared skin's pairing arrives generated, never authored.
kernel-wire	route	route-spec	straight	straight, orth, arc (honours the ends' faces), or the hull route <side> of <selector>.
kernel-wire	via	address-list	empty	Waypoints of a string; an index bond draws its chord and reads none.
kernel-wire	species	declared-name	empty	Semantic species.
kernel-wire	wind	pair	zero	Winding pair on traced frames; no certified quotient geometry.
kernel-wire	cross	crossing-list	empty	Declared over and under crossings.
kernel-wire	crossing	enum(over under alternate alternate=over alternate=under)	empty	Default order for the crossings a hull route derives.
kernel-wire	dir	direction-policy	none	Direction mark of a directed index.
kernel-wire	stroke	enum(solid dashed dotted)	solid	Declared stroke pattern of the rail.
kernel-wire	name	identifier	generated	Addressable name.

kernel-wire	closed	flag	false	A closed cycle, smooth unless route=orth squares its turns, which a wind=cycle ignores; takes no ends.
kernel-mark	form	enum(bracket enclosure label prose)	label	Mark form; the prose form records its text and draws nothing.
kernel-mark	species	declared-name	empty	Semantic species.
kernel-mark	label pos	angle	auto	Label bearing: a compass word or a degree value.
kernel-mark	name	identifier	generated	Recorded name; no address grammar resolves a mark yet.
kernel-mark	tint	flag	false	Interior tint under an enclosure's contour; a bracket takes none.
kernel-setup	pitch	length	11mm	Base metric; the default is a fixed length, not an em-relative one.
kernel-setup	strict	flag	false	Rejects sugar; only five refusal fixtures enable it.
kernel-declare	hue	hue-source	house cycle	Species ink; the source: prefix records provenance and strips to the colour it names.
kernel-declare	base	identifier	empty	Skin the declaration extends.
kernel-declare	pairings	port-pair-list	empty	Skin-internal wires between own ports.

Prelude class	Name	Declaration
skin	mpo	base=box
skin	pill	base=roundrect

The tables list the commands and keys for new documents.

7 Custom Atoms and Typographic Style

Good typography depends upon a disciplined visual vocabulary. In *tikz-tensor-networks*, the standard tensor nodes are shared consistently across all layout frames: a `dot` represents an unnamed tensor site; a `box` or `pill` encloses an inscribed tensor label; an `mpo` provides facing physical input and output ports for local operators or MPO nodes; and a `ring` denotes a single-site operator or gauge matrix seated upon an existing wire. Always remember that a tensor’s semantic kind, its typed ports (physical vs. virtual), and its visual skin are three independent concepts.

7.1 Declaring a custom atom

When should you extend the node vocabulary? The command `\tndeclare` exists for those occasions when the built-in skins cannot express a specialized quantum many-body entity (such as a multi-legged projector P , an isometry W , or a disentangler U):

```
\tndeclare{atom}{\langle command \rangle}
{skin=\langle dot/box/pill/mpo \rangle,
 ports={\langle face:type \rangle,...}}


---


\tndeclare{atom}{\tnprojector}{
  skin=box,
  ports={180:virtual, 0:virtual, 90:physical}
}
\begin{tenkz}[cols=1] \tnprojector{P} \end{tenkz}
```

Each port face is specified as an angle in the atom’s internal coordinate axes: 180 and 0 are the western and eastern ends of the wire, while 90 points north and 270 south. The allowed types are `virtual` and `physical`. The declaration defines a one-label atom command that accepts optional per-cell keys. Notice that the port specification must be exhaustive: in *tikz-tensor-networks*, a geometric anchor without a mathematical type does not constitute a valid extension contract.

¹

To abbreviate repeated structures in a paper, use ordinary TeX definitions whose expansions produce the public *tikz-tensor-networks* syntax. Keep benchmark examples expanded so that each source states the topology it draws.

7.2 Criteria for language extensions

To protect the language from creeping bloat, *tikz-tensor-networks* enforces a strict gate for new features. A proposed key must demonstrate at least three distinct, manifested consumers in the published literature; a new command must represent a genuinely new grammatical class. Furthermore, every accepted feature requires a registry entry, a standalone example, a coded diagnostic, a negative test case, and a teachability review.

¹ Why a command, not a key?

Declaring an atom introduces a new mathematical entity and therefore a new word in the atom grammatical class. By contrast, changing the skin or role of an existing atom is done through options.

7.3 House style and typography

Consistent style is an art that rewards restraint. Use `\tnset` in your preamble to declare document-wide metrics or semantic themes, and use picture options for local policies. Document themes may adjust colors and typography, but they never alter topology. Use semantic roles to convey function, and textual labels to establish mathematical identity. Never resort to arbitrary rainbow colors merely to tell identical tensors apart!

In *tikz-tensor-networks*, labels dwell in reserved bands outside measured silhouettes. If a diagram feels crowded, increase the documented metric spacing or layer separation. Manual coordinate nudging is an emergency measure of last resort, and has no place in canonical benchmark code. The layout engine measures occupied ink automatically, but authored routes and label stations can still overlap. Inspect the resulting figure and adjust the documented placement options when needed.

8 The Benchmark Suite

The manual teaches the language. The separate *tenkz RMP benchmark* stress-tests it against 130 agreed targets: 92 published targets in paper order and 38 author-workbench targets in source order. It is built from `docs/tenkz/rmp-benchmark.tex`; this manual does not duplicate its target catalogue.

Each target lives in one preamble-free source file under `tests/tenkz/rmp/`. That exact file is wrapped for standalone regression, input by the book, rendered by the web/SVG build, and printed verbatim on the target's spread. No source body is copied into a driver, manifest, or manual.

Every target header states its formula, ink, boundary, source anchors, and capabilities. The verso page prints that contract and complete line-numbered source. The recto page gives the vector rendering, audited boundary signature, dimensions, digests, and print/web verdict. Digest changes make a prior verdict stale automatically.

One script drives every check:

```
python3 scripts/tenkz_rmp.py check --id <target>
python3 scripts/tenkz_rmp.py check --section <section>
python3 scripts/tenkz_rmp.py check --all
python3 scripts/tenkz_rmp.py book --all
python3 scripts/tenkz_rmp.py render --all
python3 scripts/tenkz_rmp.py compare --all --source-root /absolute/path/to/author-source-tree
```

The benchmark style controls typography only. It does not extend *tenkz*. The separate author-source tree is not part of this repository. Reference comparisons are compiled temporarily from its mapped source blocks; prebuilt PDFs are not design evidence. Targets without author drawing source do not pretend that a reference implementation exists. They carry the status `paper-context-only`.

9 Recovery from Errors

9.1 Interpreting compiler diagnostics

When you encounter an unexpected rendering or a compiler refusal, do not despair! $\text{\TeX}$ tools are designed to assist you, and in *tikz-tensor-networks*, the golden rule of debugging is simple: **read meaning before ink**.

Begin by opening the job’s companion `.tnlog` file and locating the picture’s coded diagnostic or boundary event. Ask yourself: What mathematical object should this tensor network represent? A state $|\psi\rangle$ has open physical indices; a matrix window has two open virtual indices; and a doubled transfer window has four. A fully contracted expectation value or partition function has an empty boundary signature. When inspecting an equation, compare the boundary signatures of each side term by term, ensuring that all physical and virtual indices balance across the equality.

Symptom	First correction
An operator rendered as a scalar	State <code>boundary=open</code> , or type the open ports; an unstated side draws no leg.
A periodic trace looks like a cup	Use <code>boundary=periodic</code> for same-row closure; use side <code>cup</code> only between adjacent layers.
A label touches ink	Restore a documented metric or label band; do not hide the problem with an undocumented coordinate offset.
Requested ink disappears	Treat it as an implementation defect. Unsupported ink must be a hard coded error, never a warning followed by omission.

A few simple habits will save you from the most common syntactic pitfalls: Always enclose in curly braces any key value containing a comma or an equals sign (for instance, `east={cup=$X$}`). Remember that complex conjugation is mathematical label notation and wire direction, not an internal atom flag. When creating a hole in a lattice, distinguish between `void=open` (where incident indices survive around the missing site) and `void=sealed` (which removes the site and its incident bonds entirely).

9.2 The internal pipeline

Behind the scenes, *tikz-tensor-networks* translates your declarations into vector ink through seven strictly sequenced stages:

```
language -> model -> style/atoms -> geometry -> kernel
          layout -> rendering -> events.
```

Each stage owns a distinct responsibility: Language parses public declarations and emits coded diagnostics. Model normalizes those declarations into canonical records. Style manages metrics, atom descriptors, label bands, and silhouettes. Geometry computes frames, daylight separation, wire routing, and crossings. Layout handles genre-specific grid placement. Rendering draws ink faithfully without parsing syntax or altering topology. Finally, Events serializes the resolved model for auditing.

The top-level file `tikz-tensor-networks.sty` is a logic-free load map. Every internal source file opens with a five-line contract declaring its consumed inputs, promised outputs, owned private state, invariants, and downstream consumer. Private control sequences follow a strict owner-prefixed convention, guarded by an automated symbol census.

Full build and debugging workflows live in `HACKING.md`, while the underlying ownership contracts are detailed in `ARCHITECTURE.md`. Always keep the maintainer’s motto in mind: after every meaningful change, compile, lint, audit the versioned event stream, and inspect the rendered pages. Exit codes verify compilation, but only your eyes can inspect the beauty of a diagram!

10 References

The `tikz-tensor-networks` package was developed for `TNLEAN`, a Lean 4 formalization of the mathematical theory of tensor networks, and its diagram grammar reflects the structural insights established across more than three decades of quantum many-body research. In `tikz-tensor-networks`, a diagram is not merely placed ink; it is a typed mathematical record whose ports, contractions, and boundary signatures respect the algebraic reality of tensor networks.

The references below gather the foundational and modern tensor-network literature that forms the theoretical backbone of the language, with particular emphasis on the seminal works of J. I. Cirac and collaborators on Matrix Product States (MPS), Matrix Product Density Operators (MPDO), Matrix Product Unitaries (MPU), and Projected Entangled Pair States (PEPS), alongside standard reviews, diagrammatic lecture courses, and machine-checked formalizations.

- [1] J. I. Cirac, D. Pérez-García, N. Schuch, and F. Verstraete. Matrix product states and projected entangled pair states: Concepts, symmetries, theorems. *Reviews of Modern Physics*, 93(4):045003, 2021. [doi:10.1103/RevModPhys.93.045003](https://doi.org/10.1103/RevModPhys.93.045003). [arXiv:2011.12127](https://arxiv.org/abs/2011.12127).
- [2] D. Pérez-García, F. Verstraete, M. M. Wolf, and J. I. Cirac. Matrix product state representations. *Quantum Information & Computation*, 7(5):401–430, 2007. [arXiv:quant-ph/0608197](https://arxiv.org/abs/quant-ph/0608197).
- [3] F. Verstraete and J. I. Cirac. Renormalization algorithms for quantum-many body systems in two and higher dimensions. 2004. [arXiv:cond-mat/0407066](https://arxiv.org/abs/cond-mat/0407066).
- [4] J. I. Cirac, D. Pérez-García, N. Schuch, and F. Verstraete. Matrix product density operators: Renormalization fixed points and boundary theories. *Annals of Physics*, 378:100–149, 2017. [doi:10.1016/j.aop.2016.12.030](https://doi.org/10.1016/j.aop.2016.12.030). [arXiv:1606.00608](https://arxiv.org/abs/1606.00608).
- [5] J. I. Cirac, D. Pérez-García, N. Schuch, and F. Verstraete. Matrix product unitaries: Structure, symmetries, and topological invariants. *Journal of Statistical Mechanics: Theory and Experiment*, 2017(8):083105, 2017. [doi:10.1088/1742-5468/aa7e3b](https://doi.org/10.1088/1742-5468/aa7e3b). [arXiv:1703.09188](https://arxiv.org/abs/1703.09188).

- [6] A. Molnár, J. Garre-Rubio, D. Pérez-García, N. Schuch, and J. I. Cirac. Normal projected entangled pair states generating the same state. *New Journal of Physics*, 20(11):113017, 2018. doi:[10.1088/1367-2630/aae94f](https://doi.org/10.1088/1367-2630/aae94f). arXiv:[1804.04964](https://arxiv.org/abs/1804.04964).
- [7] M. Sanz, D. Pérez-García, M. M. Wolf, and J. I. Cirac. A quantum version of Wielandt’s inequality. *IEEE Transactions on Information Theory*, 56(9):4668–4673, 2010. doi:[10.1109/TIT.2010.2054562](https://doi.org/10.1109/TIT.2010.2054562). arXiv:[0909.5347](https://arxiv.org/abs/0909.5347).
- [8] D. Pérez-García, M. M. Wolf, M. Sanz, F. Verstraete, and J. I. Cirac. String order and symmetries in quantum spin lattices. *Physical Review Letters*, 100(16):167202, 2008. doi:[10.1103/PhysRevLett.100.167202](https://doi.org/10.1103/PhysRevLett.100.167202). arXiv:[0802.0447](https://arxiv.org/abs/0802.0447).
- [9] N. Schuch, I. Cirac, and D. Pérez-García. PEPS as ground states: Degeneracy and topology. *Annals of Physics*, 325(10):2153–2192, 2010. doi:[10.1016/j.aop.2010.05.008](https://doi.org/10.1016/j.aop.2010.05.008). arXiv:[1001.3807](https://arxiv.org/abs/1001.3807).
- [10] G. De las Cuevas, J. I. Cirac, N. Schuch, and D. Pérez-García. Irreducible forms of matrix product states: Theory and applications. *Journal of Mathematical Physics*, 58(12):121901, 2017. doi:[10.1063/1.4998639](https://doi.org/10.1063/1.4998639). arXiv:[1708.00029](https://arxiv.org/abs/1708.00029).
- [11] J. I. Cirac, J. Garre-Rubio, and D. Pérez-García. Mathematical open problems in projected entangled pair states. *Revista Matemática Complutense*, 32(3):579–599, 2019. doi:[10.1007/s13163-019-00305-6](https://doi.org/10.1007/s13163-019-00305-6). arXiv:[1903.09439](https://arxiv.org/abs/1903.09439).
- [12] S. Lu, E. Tjoa, and J. I. Cirac. Multi-agent autoformalization of tensor network theory. 2026. arXiv:[2607.07857](https://arxiv.org/abs/2607.07857).
- [13] M. Fannes, B. Nachtergaele, and R. F. Werner. Finitely correlated states on quantum spin chains. *Communications in Mathematical Physics*, 144(3):443–490, 1992. doi:[10.1007/BF02099178](https://doi.org/10.1007/BF02099178).
- [14] U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011. doi:[10.1016/j.aop.2010.09.012](https://doi.org/10.1016/j.aop.2010.09.012). arXiv:[1008.3477](https://arxiv.org/abs/1008.3477).
- [15] R. Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014. doi:[10.1016/j.aop.2014.06.013](https://doi.org/10.1016/j.aop.2014.06.013). arXiv:[1306.2164](https://arxiv.org/abs/1306.2164).
- [16] J. C. Bridgeman and C. T. Chubb. Hand-waving and interpretive dance: An introductory course on tensor networks. *Journal of Physics A: Mathematical and Theoretical*, 50(22):223001, 2017. doi:[10.1088/1751-8121/aa6dc3](https://doi.org/10.1088/1751-8121/aa6dc3). arXiv:[1603.03039](https://arxiv.org/abs/1603.03039).
- [17] M. M. Wolf. Quantum channels & operations: Guided tour. Lecture notes, Technical University of Munich, 2012. <https://mediatum.ub.tum.de/doc/1099654/1099654.pdf>.
- [18] The TNLean Project Contributors. TNLEAN: A Lean 4 formalization of the mathematical theory of tensor networks. 2026. <https://github.com/LionSR/TNLean>.
- [19] S. Lu. *tikz-tensor-networks*: Declarative tensor-network diagrams in L^AT_EX. L^AT_EX package, version 0.8.0, 2026. <https://github.com/LionSR/tenkz>.

THE TENKZ LANGUAGE AT A GLANCE

Layout and frame

tenkz — one typed tensor network in one frame

The five grammatical classes

atoms — `tn`, `tnfuse`, `tntree`

connections — `tnwire`, `tnbond`, implicit bonds, cups, traces

annotations — `tnmark`, labels

composition — TeX equations, alignment, `tngroup`

setup — `tnset`, `tndeclare`, themes

Boundary conditions

boundary=open — mints the side legs and puts them in the signature; an unstated side draws no leg

boundary=none — the explicit spelling of the default

boundary=periodic — returns each row to itself

west=cup / **east=cup** — connects adjacent layers, and outranks `boundary=` on its own side

.tnlog — records the resolved boundary signature

Stylistic conventions

one layout — choose from topology, not silhouette

explicit scope — document, picture, object, connection

typed ports — virtual joins virtual; physical joins physical

role/species — semantic style, never topology

route=arc — leaves and enters along its ends' faces; the family is straight, orth, arc

Verification and tools

xelatex — compile the exact source

tenkz_audit.py — audit the versioned semantic event stream

tenkz_lint.py — reject non-canonical source

pdftocairo — render and inspect the affected pages

tenkz_rmp.py — build the separate 130-target benchmark book