Supplementary Methods:
VSM (Visual Syntax Method) in detail

Below follows a full description of VSM. Following up on the main document's concise overview, this Supplementary text adds more details, additional perspectives, and justifications for VSM's design choices, intended for interested readers, but also for experts in current knowledge representation (KR) methods.

i. Goal

With VSM, we aim to develop a method that enables people to consider any unit of information, and reformulate it into a semantically precise, computable form that clarifies its full meaning and context. We aim for VSM to be applicable to the widest range of use-cases, so it works for diverse, heterogeneous, and detail-rich information. At the same time, we aim for VSM to be easily usable by many people with diverse skills and expertise, which requires that it keeps the inherent complexity of real-world information manageable. Like this, VSM enables humans to manually create meaningful **computable information**, with high **ease of use**, and high **expressive power**.

VSM includes selected strengths from existing methods to represent knowledge: natural language, controlled languages[1], ontologies[2], and RDF triples[3]. But in order to meet the requirements above, VSM also intentionally differs in several ways, and defines a fresh set of foundational principles.

One can compare VSM with how a high-level programming language[4] enables people to make an algorithm explicit by encoding it in a way that both humans and computers can manage. Similarly, VSM is a tool to make an idea, thought, or piece of information explicit, by encoding it with intuitive, elemental components, in a way that both humans and computers can readily process and manage.

ii. Design

VSM is the combination of a new **semantic model** (or procedure, or *method*) and the design for a supporting **user interface** that together enable people to produce and read computable information.

As a Semantic Model, VSM is like a language. It has a simple, small set of rules for constructing information in the form of a so-called *VSM-sentence*, which is a semi-linear statement form[5] that is flexible and intuitive for humans to work with. Similar to how a natural language consists of words and a grammar, VSM has two building blocks: *VSM-terms* and *VSM-connectors*. In addition, the three VSM Principles describe how terms and connectors can be meaningfully combined into virtually limitless combinations, in order to unambiguously represent essentially any type of complex information.

As a User Interface (UI), VSM defines the design for a software input-component that can be embedded in other software, for instance a web page. It enables people to enter (i.e. make explicit to the computer) a possibly complex piece of information, one per input-component. This component is called a *VSM-box*.

[1] https://en.wikipedia.org/wiki/Controlled_natural_language, and …/Formal_language.
[2] https://…/wiki/Ontology_(information_science), and …/Open_Biomedical_Ontologies.
[3] https://en.wikipedia.org/wiki/Resource_Description_Framework.
[4] In contrast to low-level, bare-metal *assembly languages* that offer little to no assisting abstractions.
[5] This *statement* form is inspired by controlled languages. But unlike them, VSM avoids intricate rules on word order or fixed keywords. VSM uses a simpler, generalized way to clarify structure.

iii. Current status

We have built a prototype user-interface for curation[6] based on VSM, for the purpose of demonstrating VSM's utility. It is embedded in the accompanying webpages[7] for interactive examples, and we have been using it for several years at our lab for a specific biocuration project[8], to test and experience its use in real life. This exercise has given us many insights that are now included in this paper, in which we focus on VSM's full design and principles.
Recently, we have also built a production-ready (well-designed, well-documented, and easily reusable) VSM-box web-component, available in the *Vsmjs* organization[9] on GitHub. This core user-interface component and its supporting modules result from a large software-design and programming effort, and their implementation will be detailed in a separate publication[10].
We hope that this work may be the start of a potentially large ecosystem of supporting and derived modules: of tools that facilitate specific or general use of VSM as an interface for communicating diverse and context-rich knowledge with computers. We therefore welcome the community to contribute in Vsmjs on GitHub, or to create depending projects.[11]

iv. Caveat

The design of VSM incorporates several ideas from existing technologies, but it also includes significant differences. As such, we have repeatedly experienced that experts with backgrounds in related areas are prone to make initial assumptions that cause misunderstanding of key concepts of VSM. In what follows, we hope to preempt such assumptions through extensive clarifications, yet we would like to advise readers familiar with related KR technologies to be especially mindful of the intentional differences.[12]

v. Approach of this text

VSM is a practical way of constructing meaning, inspired by human thinking; i.e. it combines a focus on human usability, with a particular view on semantics. For most of VSM's intended audience, the prime interest will likely be its usability aspect. Therefore, we have written this VSM paper mostly from that perspective. Each of the semantic principles that make VSM work, are introduced starting from the perspective of usability, inspired by how humans/curators are used to think. Further elaboration on VSM's semantics from a mathematical or formal-logic perspective, technical implementations of its user interface, etc., will be described in specialized publications.

While VSM was initially designed to solve a problem in the biosciences, we believe that VSM is much more generally applicable. Therefore, in what follows, we often use and start from domain-neutral examples.

---

[6] Knowledge *Curation* = the manual (or semi-automatic) conversion of information from unstructured text etc. into a precise, computer-understandable (i.e. computer-actionable) form.
[7] At https://vsmjs.github.io/vsm-pages/vsm and https://vsmjs.github.io/vsm-pages/examples.
[8] Described in draft form at https://vsmjs.github.io/vsm-pages/scicura but not part of this paper.
[9] At https://github.com/vsmjs : several modules that culminate in the *vsm-box* web-component.
[10] Ref.: "**VSM-box**: general-purpose interface for biocuration and knowledge representation", Preprint at https://doi.org/10.20944/preprints202007.0557.v1 (2020).
[11] E.g. https://github.com/**UniBioDicts** : modules that provide bioscience terminology to a vsm-box; and https://github.com/MI2CAST/**causalBuilder** : uses it for curating molecular causal interactions; (UBD doi:10.20944/preprints202007.0586.v1; causalBldr. doi:10.20944/preprints202007.0622.v1).
[12] Still, a mapping between VSM and existing technologies is one of our active research interests. As a first connection, see a proposed conversion to RDF: https://github.com/vsmjs/**vsm-to-rdf**.

1. VSM-terms

We will use a *VSM-box* as the setting in which we introduce both the semantic model and the user-interaction design of VSM.

A VSM-box, the user-interface of VSM, supports the entering of *VSM-terms* via an autocomplete function, by way of a drop-down list that lets users select the specific term that matches the concept they want to define (Fig. S1a). Each shown, entered **VSM-term** (blue rectangle with text, as a user-interface element) represents a single **concept**[13], or a *unit of meaning*, represented by a **term** (i.e. term-string). A term can consist of one or more words, for instance "homo sapiens", "is located in", or "binds to". A term is in fact a human-friendly representation of a unique **identifier** (ID): a number or code that computers work with. Each term and its linked ID ideally come from an authoritative list, whose purpose it is to unambiguously define the meaning of each of its member terms, and to give it a unique ID together with known synonyms and other information. Such a list can be a domain ontology (e.g. Gene Ontology[14]), a controlled vocabulary (CV) (e.g. PSI-MI[15]), a list of entities (e.g. genes, NCBI Gene[16]), or a curator-built list (e.g. via PubDictionaries[17]). A VSM-box UI-component – which is meant to be embedded in other software – needs to be linked to one or more external term-lists or *dictionaries* like these, to give the user access to available terminology and IDs.

It can often happen that a same term is present in multiple lists and therefore linked to multiple IDs. The autocomplete function therefore helps the user to disambiguate between the available concepts or IDs. For example, for a human, a mouse, and a chicken gene with the same name, the autocomplete selection panel would suggest all three names together with the species of each. Once the intended term+ID couple is selected, only the term remains visible to the user (by default), while the ID is stored in the background, by the software or curation system. The UI may still show full information when mouse-hovering a term, or it could display a clarifying icon, or use other customizations.[18]

In addition to supporting the above-mentioned terms with multiple meanings, or **homonyms**, a VSM UI also supports **synonyms**, i.e.: different terms may be used to represent a same concept. This enhances user-friendliness in several ways.[19] First, it enables VSM-sentences to correspond better to the text of the publication that is curated, because for example: gene and ontology terms often have many synonyms that are frequently used in the biological literature. When curating, synonyms (linked to the correct ID) make it easier to mentally map a long curated VSM-sentence onto terminology used in the publication, especially for other curators who may double-check the VSM-sentence. If needed, one can still add a UI-option to switch to displaying terms by their main, official name. – Second, some official terms may be excessively long (e.g. in GO[20]), and curators may prefer to use an available

[13] https://en.wikipedia.org/wiki/Concept

[14] http://geneontology.org : "The Gene Ontology Resource: 20 years and still GOing strong", NAR, 2019.

[15] https://www.ebi.ac.uk/ols/ontologies/mi : Molecular Interactions Controlled Vocabulary

[16] https://www.ncbi.nlm.nih.gov/gene

[17] http://pubdictionaries.org . Ref.: "Open Agile Text Mining for Bioinformatics: The PubAnnotation Ecosystem", Bioinformatics, 2019.

[18] In this document we discuss features for the VSM-box UI that are either implemented or that are ideas for customization options or extensions. We refer to the Vsmjs organization on GitHub to follow up on which features are available, and to future publications on this topic.

[19] Ref.: "Jointly creating digital abstracts: dealing with synonymy and polysemy", BMC Res Notes, 2012.

[20] Gene Ontology includes a dictionary (controlled vocabulary) of terms about gene functionality etc. – Example of a long term: "positive regulation of transcription from RNA polymerase II promoter", and

abbreviation to keep produced statements readable. – And third, it enables to make VSM-sentences look more like natural language sentences. For example, consider a single relation concept that has two synonyms: the verb-form “is-located-in” and the preposition “in”, so that both represent the same ID. Then the VSM-sentence “A activates B in C” is more natural to read when one can use the preposition “in” instead of the verb form (Fig. S1b vs S1c). This support for synonymous word-forms and also conjugations (as also shown in many following examples), is a key factor in enabling the creation of VSM-sentences that are easily human-readable.

In summary: VSM’s support for synonyms and homonyms enables a curator to write easily readable information, with reduced cause for confusion; and once specified, each VSM-term in a statement represents one specific ID.

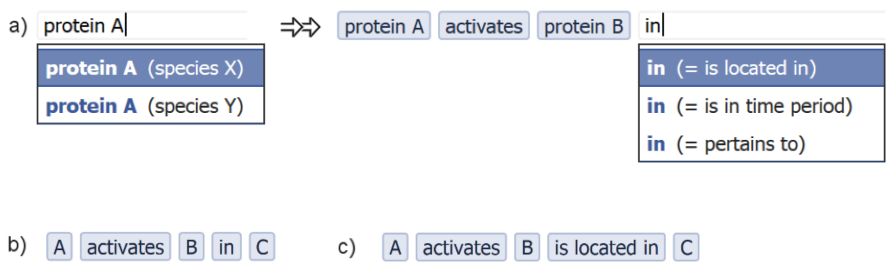

Fig. S1 | **Entering VSM-terms. a**, A curator searches and enters a sequence of terms, each linked to an identifier that represents a particular *concept* or meaning (two of five searches shown). Each selected term+ID combination results in the creation of a VSM-term user-interface element, representing that term+ID couple. **b**, The full sequence of VSM-terms. This example is just as easily readable as a natural language sentence, among others because the user was able to use the preposition “in” for the relation “is located in”, both of which would here represent the same identifier. **c**, This sequence represents the same five IDs but illustrates a less natural choice of terms.

Note: in text that follows, we enclose example VSM-sentences and VSM-terms in **double quotes**, and replace spaces in multi-word terms by **dashes**, like in this example VSM-sentence: “X is-located-in Y”. When we give examples as natural language phrases, we use ‘**single quotes**’.

many are longer. This length originates from GO being designed as a label-based classification system for biological concepts, resulting in a deep tree structure of ever-more-specific terminology.

2. VSM-connectors

2.1. Indicating structure with the trident

A sequence of entered VSM-terms may initially look like a natural language statement, in which words have been disambiguated. But a VSM-sentence is typically not copied verbatim from a paper[21]. We have often experienced that this needs to be emphasized, so we declare it as:

VSM Principle 1. A VSM-sentence is a curator's interpretation, or concise rephrasing, of some information's essence[22]. Curators need to reflect this understanding of knowledge that they acquired from somewhere (like literature) by entering terms in such a way that a syntax (i.e. structure of how VSM-terms link together to create composed meaning) can be specified with VSM's syntax connectors. These *VSM-connectors*[23] are added through a user interface and are available in three main types: the *trident* (incl. three *bident* subtypes), the *list-connector* (of any length), and the *co-reference*.

In what follows, we explain how VSM-connectors are used to syntactically specify the meaning of VSM-sentences. We start with simple statements, and next progress toward statements with complex internal structure that convey rich contextual information.

We start with the **trident**. A curator can add structure to a statement by identifying triples, or subunits consisting of three VSM-terms. At first glance these triples look similar to RDF-triples[24], yet we elaborate on that further below. The UI enables a user to specify triples by linking terms with a trident connector. A trident is added via three consecutive clicks, one above each term according to the role it has in the triple, in the order of: subject, relation, and direct object[25]. The corresponding, attaching parts of the trident, which we call its **legs**, are: a simple line, a line decorated with an up-pointing filled triangle, and a down-pointing arrow, respectively. Figure S2a shows a simple example one-triple statement that is specified with a trident, and that expresses that a "John" "eats" a "chicken".

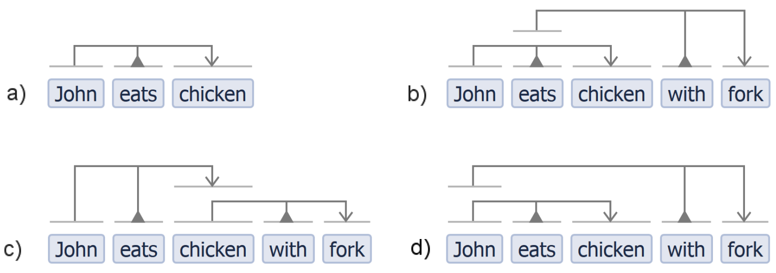

Fig. S2 | **Specifying triples with tridents. a**-**b**, The user adds trident VSM-connectors that specify triples (subject-relation-object triplets) of *individual terms* (as indicated by leg positions and leg *foot*-widths); as subunits defining the intended syntax. The resulting composite semantics is explained in the main text. **c**-**d**, Meaning changes when the rightmost connector's subject-leg is attached to a different term.

[21] In fact, we are able to make it *resemble* natural language, because we are *able/allowed* to make VSM-terms *appear* as conjugated verbs, nouns, prepositions, etc. That ensures that we can keep complex information (i.e. longer sentences) easily readable for us, humans, too. Computer do not understand conjugations etc. reliably well, so we need to clarify the structure of a VSM-sentence in another way.

[22] Examples of concise paraphrasings (applied to free text): https://en.wikipedia.org/wiki/Concision .

[23] As a result, VSM-terms' text can show a readable *formulation*, while VSM-terms' IDs and the VSM-connectors show the *conceptualization* structure that exists in the information, intuitively and clearly.

[24] E.g. triples represented in RDF Turtle: https://www.w3.org/TR/turtle/#simple-triples .

[25] An indirect object must be captured via an extra relation; e.g. 'Ann gives Eve X' as 'Ann gives X to Eve'.

Next we will show by example how to use multiple tridents, after which we can explain VSM Principles 2 and 3. Figure S2b shows the addition of more terms or context via a second trident, again attaching to *individual VSM-terms* (n.b.: <u>not</u> to any previous subunit as a whole). This specifies a second triple subunit that expresses: this "eating" happens "with" (or "using"), a "fork". We represented this second triple's relation by a preposition-type synonym, for natural-language-like readability. In contrast, Fig. S2c shows an alternative version in which a chicken is using a fork instead. This illustrates that attaching a connector leg to a different term fundamentally changes the meaning. Similarly, Fig. S2d states that John uses a fork, although it does not say he eats chicken with it (he could be eating chicken with his hand, while using the fork for another purpose). Only by connecting the second trident to the "eating" term, one captures the correct meaning: 'the eating (by John, of chicken), is by use of a fork'.

This example introduces an elementary way of thinking, as a basis that can be scaled up to knowledge of any complexity. This is further elaborated in:

🕮 **VSM Principle 2.** A connector leg always attaches to one particular term, and never to an entire other triple/subunit. This is even true when a leg attaches to a term that is used as a relation in another triple; and this makes VSM different from the typical approach in RDF.[26] This difference is intentional, and lies at the basis of making the syntax-definition process flexible, scalable, and intuitive, following the thinking of the curator. It is rooted in VSM's specific approach to *treat all VSM-terms equally*: both so-called entities and relations are first-class citizens. In VSM, *any term is only viewed as a relation instead of an entity, within a particular subunit, under a particular trident* that connects to it with its relation-leg. Because of that, any term that happens to function as a relation under one trident can function as a subject or object (i.e. an entity) when viewed under another connecting trident.

For instance, in Fig. S1b the verb term "activates" can (and should) equally be thought of as a noun: 'activation'. In Fig. S2b the verb term "eats" is in meaning identical to its noun-form 'the eating'. If both lexical alternatives are available as terms in a dictionary for VSM, they should carry the same ID.

In summary: *every VSM-term should primarily be thought of as an entity – i.e. a referable thing*.

<u>Note</u>: the VSM Principles are not verbatim definitions. They may be rephrased using various wordings, for instance depending on the intended readers' background or on a chosen format of explaining VSM. This can offer additional perspectives that put these principles into proper context, and that together better convey the core idea. See e.g. footnote [27].

[26] In RDF one typically makes a *reification* construct in order to point to a relation; i.e. one creates an additional *entity* (a *blank node*) that represents that *relation* (or in fact, its entire triple), that one can then point to. Because RDF treats *relations* as fundamentally different things than *entities*. – (For an overview of four approaches or proposals for referring to relations in RDF, see ref: "Hernández D, et al. Reifying RDF: What works well with Wikidata?, ISWC, 2015"). – In fact Principle 2, pertaining to relations, is similar to the *n-ary relations* approach in RDF, if that would be applied consistently to all relations (except for a few *primitive* relations like has-subject/object/etc., which is what the VSM-connectors actually represent).

[27] Principle 2 is crucial, so we repeat it once more in other words:
Only when some trident attaches to a VSM-term with its relation-leg, only then is the term made to be seen as a relation. And then only so, under that one trident, i.e. locally in that one triple subunit.

What RDF (or formal logic) typically view as a relation (or *predicate*), VSM views only **locally as a relation** (under a particular connector, in one subunit) and views it elsewhere as an entity / as 'reified' / as a noun. (The RDF phrase 're-ified', based on Latin, means 'thing-ified' or 'made into an object').

This was an essential, necessary step for designing VSM's expressivity and simplicity in what will follow: think of **all** VSM-terms as nouns / 'thingified' concepts / something that one can always refer to or mentally point to again. Like this, one can immediately refer to any VSM-term, always in the same way.

2.2. Scaling up to more complex information

In order to scale this method up to statements with many more terms, one must keep in mind a third key principle. It provides a way of thinking that will be instrumental for constructing longer and valid VSM-sentences, and for unambiguously interpreting their meaning.[28]

🕮 **VSM Principle 3.** When a trident (or any VSM-connector except coreference, see later) is added, each of its connected terms receives extra '*context meaning*' from the other connected terms. In other words: a VSM-term does not just represent a particular entity (or thing, or instance); instead it represents a particular entity, in a particular situation (or state, or *local context*) that is specified by all the VSM-terms connected to it, directly but also indirectly. For example, in Fig. S2a the first trident makes the VSM-term "eats" (which without connections would just represent a particular but not further specified 'eating') now represent 'the eating, of a chicken, by John'; i.e. some contextual details are now *embedded* into this individual VSM-term's meaning. This applies to all three VSM-terms, so "chicken" becomes 'the chicken eaten by John', and also "John" becomes a more specific concept.

Next, when one connects a second trident to any of these three terms, one in fact refers to that term's full, *context-enriched meaning*, as created by its connection under the first trident. So with the second trident in Fig. S2b, one expresses: "the-eating-of-chicken-by-John happens-with fork". By induction, all five terms now carry an enriched meaning, acquired or composed from the other four. Next, each of these terms is individually referable again, in the same way. Continuing like this, *one can keep adding further connections and thus further context details to every single VSM-term*, recursively.

For instance, to express 'John eats a chicken, occasionally using a fork', one would add two terms and then connect a new trident to "using", to create a new triple that states: "using (of fork, by eating of chicken by John) has-frequency occasionally". (See Bidents (section 2.3) for how to intuitively add adverbs). All added detail to the meaning, delivered by connections, form what we call a VSM-term's *local context*. So each VSM-term is coupled with its own, local, or embedded context in the VSM sense.

Additional perspective is given in footnotes [29] and [30] on the notion of context in VSM.

It is essential to realize that a trident in itself carries no meaning. This should be evident from the fact that *a connector never connects to another connector, but only connects to <u>individual terms</u>*. In fact, VSM-connectors only act as *operators* on individual VSM-terms' meanings. They specify or narrow down the meaning of a term. They make some of its specifics (which make the term a particular instance of something) explicit. For example, in Fig. S2a the particular "eating" is not specified to be done in any specific way; that context is not given, so as far as we know it could be done with hands,

[28] In particular, Principle 3 will support the creation of clear VSM-sentences in which an entity is described in detail in multiple situations (see section 2.5).

[29] **The notion of *local context* in VSM** differs from *context* in e.g. Conceptual Graphs (CG) or RDF Named Graphs, which take a *formal logic*-based starting point for representing meaning. Their *context* can be seen as a *separate* object that is *shared* among a group of concepts, in order to manage the scope of logical quantifiers [CG: Sowa JF, "Conceptual Structures: Information Processing in Mind and Machine", 1984; RDF: Caroll JJ, "Named Graphs", 2005]. In contrast, VSM starts from an association-based view-point, which is instrumental for prioritizing different objectives. A *formal* definition is a future prospect.

[30] While Principle 2 defines the nature or meaning of individual VSM-terms, Principle 3 defines how the meaning of each of them changes by attaching connectors to them. Note that at this point, Principle 3 may be intuitive or even seem trivial for simple cases like Fig. S2. But it provides a crucial way of thinking for handling cases that need the coreference connector (sections 2.5 and 2.6).

a stick, fork and knife etc. But in Fig. S2b the meaning is narrowed down: by specifying the triple "eats with fork", it is made explicit what is used for John's action of eating. It narrows down "eats"'s meaning by anchoring down at least one more aspect of how it happens. Meanwhile, other unknowns remain: where John eats, if he eats alone or together with someone, how quickly he eats, etc.

Figure S3 shows longer VSM-sentences created in this way, representing information curated from scientific publications. For example, in Fig. S3a the second trident from the left declares a triple expressing "the-being-twisted-of-leaf-lamina pertains-to Oryza-that-underexpresses-YAB3-by-RNA-interference".[31] Here, the term "in" will have been linked by the curator to the concept "pertaining-to" rather than "is-located-in". Each term represents an ID from a controlled vocabulary, e.g. "leaf-lamina" is provided by PO (Plant Ontology), and "twisted" by PATO (Phenotype And Trait Ontology).[32]

We want to accentuate that any additional context, like the biological details captured here, helps to increase the quality and value of the information. Added detail could for instance resolve two seemingly contradictory 'bare triple' statements through reconciliatory contextual details.[33] Added detail is also vital when the collected information needs to be filtered down; for instance based on various quality measures determined by various use cases' requirements.

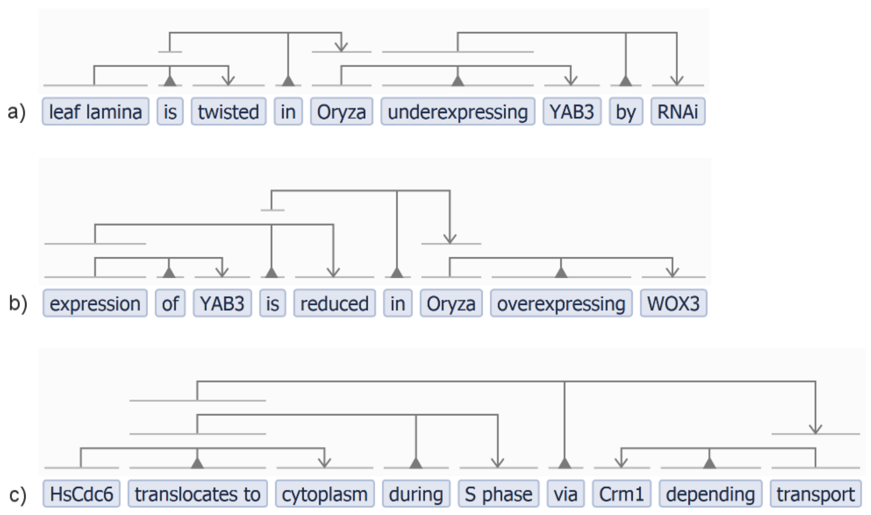

Fig. S3 | VSM-sentences based on scientific papers. The VSM-sentences are structured, reformulated versions of information from life science publications: (**a**-**b**), from [34]; (**c**), from [35]. Some of their terms come from ontologies and CVs, others are yet to be added to official CVs, especially relation terms.

We point out that all connections to a VSM-term together define its contextualized meaning; i.e. all connections are *applied simultaneously*: they add meaning in a way that order does not matter[36]. Therefore, the vertical stacking order in which connectors are added or displayed is irrelevant for the

[31] For non-biologists: this means that plant leaves are deformed in a particular way in a rice plant, that had the activity of one of its genes turned down, using some technique for achieving that.
[32] PO: ref.: "The plant ontology as a tool for comparative plant anatomy and genomic analyses", Plant Cell Physiol, 2013. – PATO: https://www.ebi.ac.uk/ols/ontologies/pato .
[33] For example both 'bare triple' statements "cat is alive" and "cat is dead" can be true. But they are true in their own specific context. We can make this context explicit by expanding the statements to e.g. "cat is alive in 2020" and "cat is dead in 2100".
[34] Ref.: "A WUSCHEL-LIKE HOMEOBOX gene represses a YABBY gene expression required for rice leaf development", Plant Physiol, 2007.
[35] Ref.: "Multistep regulation of DNA replication by Cdk phosphorylation of HsCdc6", PNAS, 1999.
[36] For cases where the semantics requires a deviation from this, see section 2.6.

connection structure (topology) and its resulting semantics. Still, some stacking order may look more intuitive than others, for example based on which subunit or term one thinks of as central in the statement (see *Head* later). Various visual sort algorithms may be developed for this UI aspect. We already implemented one that automatically rearranges the stacking of connectors as shown in Fig. S3, no matter in what order they were added.

Note that although a VSM-sentence can be mapped to RDF, either directly when two non-relation legs connect to a term, or via RDF-reification when a non-relation leg attaches to a term also used as a relation, such RDF graph renditions quickly become unintelligible to a curator (see later in Fig. S10).

A consequence of Principle 3's definition of VSM's semantics is that tridents (together with bidents and list-connectors, see later) must **never create loops**. A loop exists when one can follow a path of connections from one term that leads back to that term. (A loop is called a *cycle* in graph theory).

To illustrate: consider for instance a VSM-sentence that expresses "John eating apple calls Eve buying apple", and let us leave aside for a moment whether it is the same apple or not. The trident that connects "Eve buying apple" must always connect to a second "apple" VSM-term (Fig. S4b). It must not 'reuse' the first "apple" that would already be connected in "John eating apple" (Fig. S4a). Because if it did, it would make it impossible to interpret the meaning (following VSM Principle 3): the VSM-sentence would be read as: 'John, eating an apple, --> that is being bought by Eve, who is called by John, eating an apple, --> …'. It would have to be read like that, because *every trident-connection a VSM-term has, must be followed (i.e. applied) when interpreting its meaning.* Furthermore, if 'apple' would have additional VSM-terms attached (e.g. specifying its color), then this loop would create several interpretation possibilities, i.e.: from which of the two 'directions of encounter' in the loop would the color apply: for the situation where John eats it, or the reverse, the situation where Eve buys it (i.e. 'John calls Eve, buying apple, …')? This means that a connection loop creates context ambiguity. So in conclusion: with tridents (and bidents and list-connectors; later), one must only create hierarchical structures.

It may already be clear intuitively that when two distinct 'apple' entities (physical objects) are at play, then two "apple" VSM-terms (separate, referable concepts) are needed.[37] However, also if a sentence would mention one and the same apple twice, then the above shows that two VSM-terms must be used: one for each context. For that, a coreference connector will be needed (see later, section 2.5).

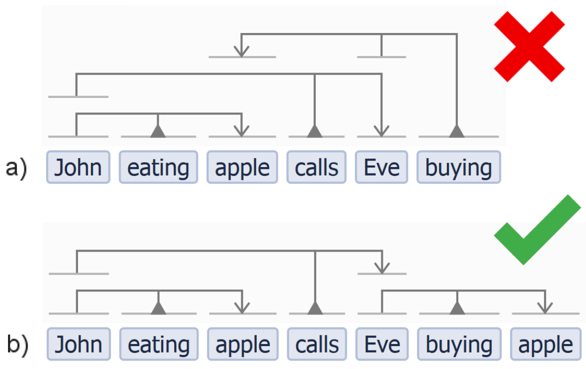

Fig. S4 | Tridents must never create connection loops. a, An incorrectly constructed VSM-sentence. Here, the interpretation of meaning, performed by following connectors that further specify terms' meanings, would result in an infinite loop and set a basis for context ambiguity, as described in the main text. **b**, A correct version that can be properly interpreted. (See Fig. S7a for the case with one apple).

[37] I.e. two *instances* of "apple" are needed. Each one would have its own instance-ID, plus a *class*-ID for the general concept it is an instance of; yet such implementation details are outside this paper's scope.

2.3. A variation on the trident: bidents

Some information is unnatural to express with triples. For instance, some verbs have no object, as in "plant flowers". Although this can be captured with a triple (the typical approach in RDF) like "plant belongs-to class-of-flowering-things", the task of stretching and reformulating such information to fit a triple structure is a burden that is best not imposed on the curator. VSM is focused on usability for the curator, and therefore the VSM UI supports tridents where one leg can be left away: **bidents**. The remaining two legs still appoint the same semantics, i.e. they assign the same role to the two terms in their subunit, as they would for a full trident.

In the example above, no object needs to be defined and the object leg is omitted (Fig. S5a). A user can create this **first type** of bident just like a trident, but with the third mouse click outside the VSM-box component, or by pressing Escape instead of the third mouse click.

Bidents also make VSM's knowledge representation *structurally consistent*. For instance, compare the statements "X stimulates: A activates B", and "X stimulates activation-of B", where the latter uses a **second type** of bident: one that omits the subject leg (Fig. S5b-c). Apart from using the appropriate synonym for the "activation" relation (with same ID), the principal difference is that the activator is unspecified in the latter case. Appropriately, the structure of both statements is similar, thanks to the bident. This consistency would not be present when using a second triple "activation having-object B" with an artificial relation inserted. This bident is also useful for capturing passive constructs where a subject is irrelevant, as could be the case in "X accelerates destruction-of B" or "to-write(=writing-of) text is rewarding". One starts this bident with two clicks above the relation.

A **third type** of bident omits the relation leg. Again, this helps usability for the curator, in this case by allowing statements to better reflect a basic structure also used in natural language: the attribute. Attributes include adjectives, adverbs, and numbers. For example, a statement about a 'white mouse' would require a "mouse having-color white" when using only the triple structure. However, if the term "white" is already classified as a "color" in its vocabulary or ontology, then making curators explicitly insert the relation "having-color" would be a waste of effort, since it can be inferred automatically. Then, a bident structure can be used, as in Fig. S5d-e. The relation is omitted via two clicks above the object term. This bident in fact indicates an **implicit relation**, most generally seen as: 'is specified by'.

However, in order to keep the semantics with relation-omitting bidents clear, consider this:

• In general, a category of terms that can be used as attributes, may need to be associated with its own default, more specific implicit-relation; e.g.: numbers (".. [has-count] 5"), measures (".. [has-size] big"), or posttranslational modifications (".. [has-state] phosphorylated"); see also Fig. S5f-g.

• In other cases, this could be associated with the attribute's linked concept as well. For example: "<tree-x> leaf" would mean "leaf [from-plant-of-species] <tree-x>", e.g. "oak leaf". Or "<gene-x> expression" would mean "expression [of] <gene-x>", e.g. "Cas9 expression".

• However, for attribute constructions as in natural language where the implicit relation is not yet automatically inferable to something more specific, one must use either an explicit triple or a single term. For example, 'actin filament' would then need to be represented as "filament composed-of actin" or as "actin-filament".

The above demonstrates that knowledge representation in VSM more closely follows the thinking of the curator, or the conceptualization structure in the human mind. This is one key aspect that makes VSM more intuitive.

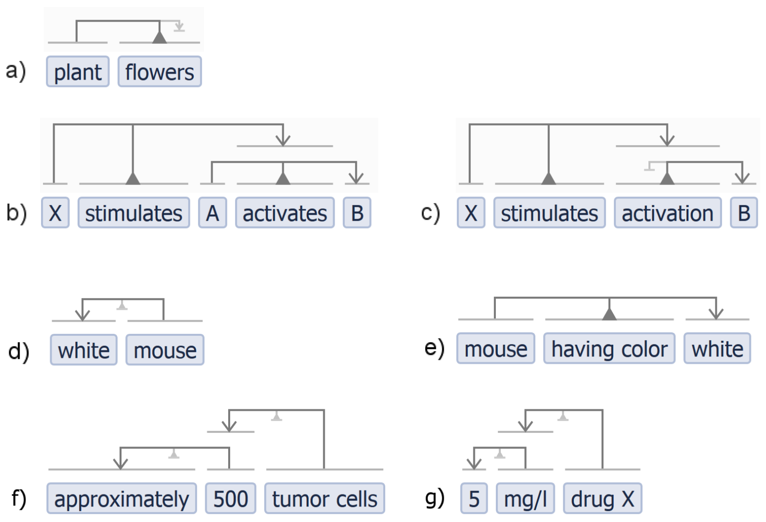

Fig. S5 | Bidents. The user can add a trident and omit any one of its three legs, i.e. create a bident, for ease of use and structural consistency as explained in the main text. **a**, The *object-omitting bident* (or *oobi*) accommodates a verb without an object. **b**-**c**, The *subject-omitting bident* (or *sobi*) specifies a unit with an unknown or unstated subject, here "activation(-of) B". The bident enables representing this information in a form structurally similar to the case where a subject "A" would be known and can be attached to a full trident's subject-leg. The availability of bidents makes this kind of structural consistency straightforward to achieve (which is not necessarily the case in RDF, see later in Fig. S10). **d**-**e**, The *relation-omitting bident* (or *robi*) implies an implicit relation 'is specified by'. It frees the curator from the obligation to explicitly define a more specific relation that could be computationally inferred. **f**-**g**, Various attributes can be connected via a robi bident. Here, the more specific implicit-relations would be inferable to be: "has-precision", "has-count", and "has-concentration".[38] Bidents are drawn with a tiny, light-colored indicator for the omitted leg type, to remind the user of their overall semantic equivalence with tridents, regarding their remaining two legs.

[38] This figure also shows that **numbers can be represented with VSM-terms**. This makes sense because one "5" can be conceptually different from another "5"; because just like other VSM-terms, a numeric concept can be embedded in a particular context. For example, one could be an "approximately 5", another a "5 ± 2", and another could be contextually specified or modified to mean "at-least 5".

For this, the autocomplete system needs to support the entry of numeric concepts. But there are infinitely many of them possible, so they cannot be stored in any prepared vocabulary. So the autocomplete will need to automatically generate an ID + term-label for any of them as needed, based on some pattern and on what the user is typing. – This was not yet implemented in the prototype (currently still embedded in some accompanying web pages), but is now supported via Vsmjs (*vsm-dictionary*) modules on GitHub (https://github.com/vsmjs), which support *vsm-box*.

Other numeric concepts like dates or timestamps could similarly be provided to the user. For this, additional dictionary software-modules that generate such concepts should be easy to implement.

2.4. Group types that combine items: the list-connector

Consider a statement like ‘A, B, and C react to D’. In order to build this with triples only, as in RDF, one would first make an “A and B”, then connect their “and” to “and C”, and then connect this second “and” to “react-to D”. However, it may be unknown in which order reactants combine. Then this construct with triples is artificial, as it declares that A and B combine into a distinct group, which then combines with C (i.e. the first “and” represents the existence of a ‘combined A+B’). In addition, the longer the list that needs to be made, the more burdensome all the artificial triples would become to the curator.

With VSM’s **list-connector**, one can express that any number of **list-item** terms all come together as one group, in a way specified by the meaning of a **list-relation** term. For example, Fig. S6a states that reactants all together react to D, as a group with unspecified internal order. The fact that group order is unspecified would be embedded in the meaning (and ID) of the chosen list-relation “and” (here: “and-(as-unordered-set)”). The curator is free to choose other list-relations as well: for instance a term “and-(as-ordered-list)”; or “sum-of”; or “either-or”, which is used in Fig. S6b to state that ‘either A, B, or C bind to D’. – This term “either-or” is also an example of a single concept that is formulated with two separated parts in natural language, and that must be captured with a single term in VSM.

The position of a list-relation among its list-items does not matter for the resulting meaning, although some placing can make a sentence easier to read; like placing “and” second-to-last in Fig. S6a. Just like with tridents, only the resulting connection- or graph-structure matters. The only difference here is: the horizontal order in which list-items are placed matters when the list-relation attaches meaning to this order; e.g. it would matter for “and-(as-ordered-list)”.

As a semantic operator, a list-connector makes its list-relation VSM-term represent the specific concept of a group of items combined in the way defined by this list-relation. Similar to how tridents operate, the list-connectors in Fig. S6a-b contextualize their list-relation term into a specific “the-A-B-C-group”, and (for lack of a less artificial way of phrasing) “the-eitherOr-union-of-A-B-C”, respectively. Then once again, another connector can link to this term as it represents the specific, whole group.

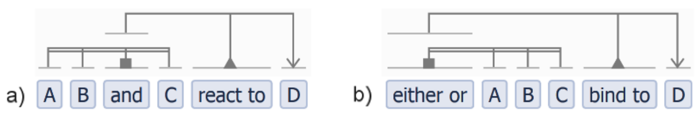

Fig. S6 | **List-connectors. a**, A list-connector assigns “and” as a list-relation that groups a number of list-elements. The connected list-relation “and” represents the combination of the list-elements *as a whole*, and as such can represent the left side of a reaction. **b**, Other terms can be used as a list-relation. Here, a list-connector contextualizes the VSM-term “either-or” to make it represent the phrase ‘either A, B, or C’.

The UI in Fig. S6 distinguishes list-connectors by drawing them with a double *backbone*, a leg ending in a filled square that connects to the list-relation, and undecorated legs that connect to list-items. The UI needs to support switching between trident and list-connector creation modes, for instance by holding the Shift button down while first clicking above the list-relation; and then clicking above all list-items. If order is important for the list-relation, then it is not the order of clicking above terms, but the order of terms as they appear in the VSM-sentence that counts.

2.5. Referring and further specifying: the co-reference

The **coreference** connector provides essential functionality that makes VSM work, both in terms of usability and in terms of semantics. It performs two related functions.

Its first, basic function is to let users build semantically correct, easily readable sentences with a term that refers to another term, for instance an "it" as in Fig. S7. We call this "it" the **child** term. The child term is a placeholder that receives meaning from its **parent** term: the term it refers to. A child term can have any label ("that", "them", etc.), and represents the same thing (i.e. entity) as its parent. For example in Fig. S7a, "it" refers to the same "apple" as its parent term (in contrast with Fig. S4b earlier, where two distinct "apple"s were mentioned). Note again that the no-loops rule (that followed from Principle 3) does not apply to coreferences, as will become clearer later. In Fig. S7, the UI draws the coreference in a dashed line, with an undecorated leg connecting to the child, and a down-pointing empty triangle as arrow connecting to the parent. The child term, drawn with a dashed blue rectangle, is a distinct VSM-term type that is entered via autocomplete, when users type e.g. 'it'. Users may add a coreference by first Ctrl+clicking above the child term, and next above the parent term.

The second function of the coreference is that it enables us to think about **two distinct concepts**. Although the parent and child term refer to one and the same *thing* (=instance of something general) in the world, they refer to that thing as it occurs in two different situations, perspectives, or *context*s.

Consider for example "Bob activates device in evening causes it beeps in morning" (Fig. S7c). The meaning embedded into the parent concept "device" is that it is being activated by someone, but should not be that it is beeping yet. Therefore (and because of no-loops), we should not connect "device" directly to "beeps" with a bident. Instead, we create a coreference-linked child concept "it", which **inherits the context** of 'having been activated in the evening' from its parent, and in addition **receives the extra context** of "beeping in morning" from its own connection environment. These two concepts can then be separately enriched with more context-details or be referred to later on (from other VSM-sentences), e.g. "Jane charges it (#1)" and "Bob silences it (#2)". A biological example could involve a biomolecule being chemically tagged in one context, e.g. in some cell cycle phase or location, which enables it to form a bond with another molecule in a further context. See also Fig. S7b, or read the extra perspective in this footnote: [39].

All this is made possible by Principle 3 that defines what a VSM-term really represents. In this light, we can also rephrase its gist the following ways: *a VSM-term represents a particular entity, in a particular situation (state, context, etc.) that is detailed (made explicit) by its connections to other VSM-terms*; a VSM-term represents a particular entity, with a particular set of available details that are given or known about it, at a particular point in a discourse; *a VSM-term represents a particular state of knowledge about a particular entity, determined by that term's own connections (primarily) and by its parent term*.

[39] In other words: the two VSM-terms refer to a same entity, whereby each one describes the entity as how it is in a distinct state/situation/context, e.g.: before vs. after an event; or a conditional state that causes an effect that eventually gives it a new state; or its known real state vs. a hypothetical state (which may be future but also present) as in "*Eve* pretends *she* likes Cat". The last example shows that the two VSM-terms do not need to represent the entity at two distinct timepoints; rather, they represent the entity in some explicitly stated context of what is known or declared about it. This can cover other aspects than time. Thus, a VSM-term does not make an implicit semantic commitment to time.

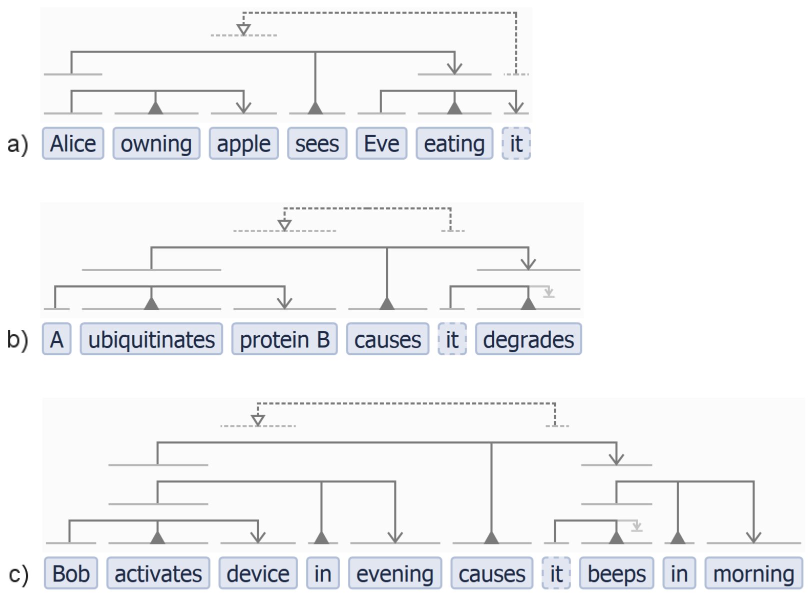

Fig. S7 | The co-reference connector. a-c, The coreference combines two functions. First, it allows to build VSM-sentences in which a same entity is referred to twice (here: same apple/protein/device) in a way that is natural to read (via a placeholder with label like "it"), and that creates no loop (of tridents, bidents, lists). Second, it enables one to work with two distinct concepts, each at a different stage of specification: in (**c**) the first "device" is not yet beeping, while the second occurrence of the device, represented by the child-term "it", not only was 'activated in evening' but also is 'beeping in morning'. A curator could still add further specifics about the device, in both of its local contexts. Similarly in (**a**), the parent-term "apple" is not yet described as being eaten by Eve; and in (**b**), the first occurrence of "protein B" is not specified to be degrading.

Now we can clarify why the no-loops rule, which followed from Principle 3, only applies to tridents, bidents, and lists-connectors, but not to coreferences. Unlike the other connectors, a coreference does not enrich the meaning of the parent term, and so it should not be followed when interpreting the parent's meaning. As a semantic operator, the coreference works only on the child concept; and also here it does not enrich an existing concept. It creates a new copy of the parent's context-embedded meaning and places that at the position of the child, where it can be further contextualized.

The two distinct concepts (parent and child) are not necessarily about past versus present, but about how much context is present in each part of the discourse. Hereby it is helpful to think of the coreference as a unidirectional weathervane or 'wind arrow' that points to where context is 'flowing from', as in 'inherited from'. This notion is crucial to understand the following issue: the case where (trident/bident/list-)connector attachments are non-interchangeable, as explained in the next section.

2.6. Non-interchangeable connections

When connecting multiple tridents or bidents to a VSM-term, we use one assumption: that the order of connecting them does not matter. This is usually true. In natural language one can test this by observing that the meaning of some statement does not change when switching attribute positions; e.g. 'big white mouse' has the same meaning as 'white big mouse'. Likewise in VSM, each new connection simply adds context in an interchangeable, additive way: in Fig. S8a, "mouse" is specified to be both "big" *and* "white" at the same time.

However, in some cases order does matter, as e.g. in the statement "half-of black dogs". By connecting "half-of" and "black" to "dogs", one can construct three different meanings (see the sets in Fig. S8b1-3):

1. 'some half of the group of dogs, that consists of only black ones': here "dogs" is specified to "half-of" and "black" at the same time (while still being half-of in total);
2. 'some half of: black ones of the dogs';
3. 'black ones of: some half of the dogs'.

In **case 1**, the attributes are applied (i.e. add context meaning) simultaneously, while in cases 2 and 3 they are applied in one of the two possible orders.

What happens when order matters, conceptually, is that we first compose an idea, and then we isolate it and refer to it, without letting the further added context mix in with the original idea. This is in fact exactly how also the coreference connector works: in e.g. **case 2** in Fig. S8b2 one can first construct a parent concept "black dogs", and then create a placeholder child concept, e.g. labeled "them"[40], that points with a coreference connector to the parent. This isolates the parent's meaning and transfers it onto the child, unidirectionally. Finally, one can use this child term "them" for building the rest of the VSM-sentence, e.g. by connecting it to "half-of" and "escape". The coreference connector points from the further-enriched child to its isolated, meaning-providing parent.

In **case 3**, Fig. S8b3, the same reasoning is applied: here one first constructs an isolated "half-of dogs", then one limits that conceptual unit of dogs to those being black, and specifies that they escape. Note that one can still further detail the isolated part, as in e.g.: 'black ones of: some female half-of dogs'.

Notes (which may go a bit too deep for a first reading) :

• The implicit relation between "dogs" and "black" would be automatically inferable to be "limited-to-all-those-having-color" (and not just "having-color"), because black would be classified as a color, and dogs represents not just one dog, but a group of dogs.[41] Alternatively, this relation could be made explicit, but for this example's brevity and focus we used an implicit relation. Similarly, "half-of" pertains to the group of dogs, so it does not create the meaning of half of one dog.

• Each "half-of" creates an unspecified subset that is some half of the group. So in case 1, Fig. S8b1, where "black" also specifies that this half subset consists of only black dogs, and where more than half of the dogs are black, the final subset remains partly unspecified. Therefore, we had to choose an

[40] "them" is a somewhat arbitrarily chosen label for the referring (child) term. It makes the text quite nice to read in Fig. S8b2, though less so in S8b3. Any other label can be used too, like "it" or "these".
[41] A note on how to create the meaning of some 'group of dogs', starting from a single "dog" concept. Consider a bident-based construct like "11 dog", or stated explicitly: "dog <has-amount> 11". The association with the relation "has-amount" is what converts the meaning of "dog" into a group. That is how that relation would work, i.e. how it would have to be semantically interpreted.

arbitrary subset of black dogs in Fig. S8b1. This is consistent with the other two cases, where the 'half' (dark blue line) is also not further specified and thus had to be chosen arbitrarily in Fig. S8b2-3.

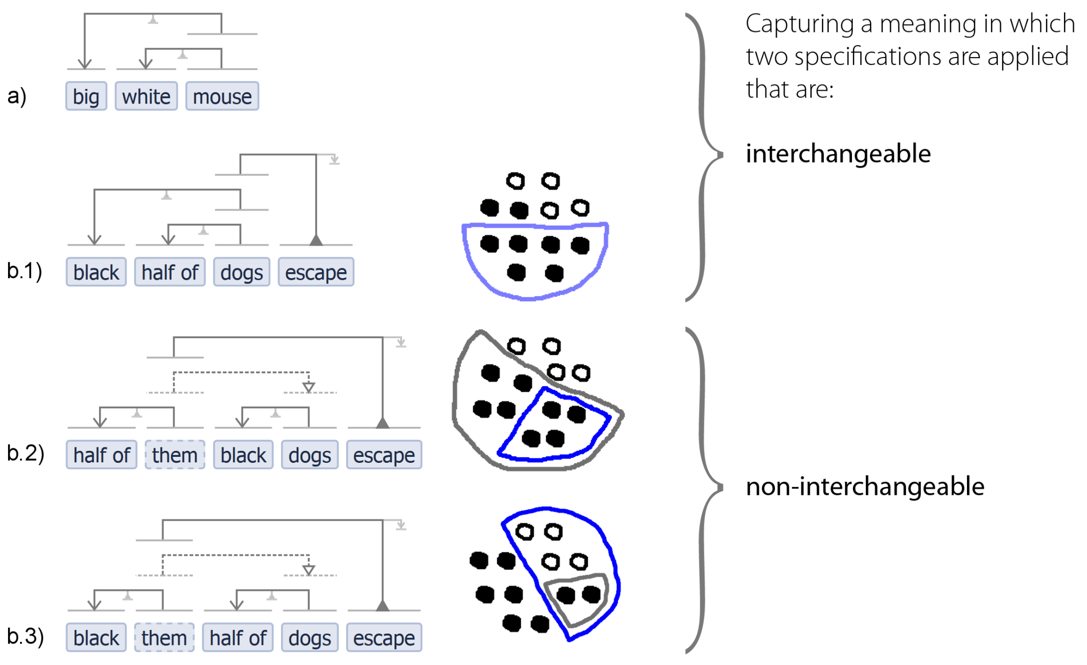

Fig. S8 | Interchangeable vs. non-interchangeable attributes. In (**a**), attribute order does not matter. But in special cases, the order in which multiple connections are added to a term can affect meaning. Here, the attributes "half-of" and "black" can be applied either (**b1**) simultaneously, or (**b2-3**) in a particular order. In the latter cases, one applies the first connection to the term "dogs", then isolates that concept as parent for a coreferencing child placeholder, that is e.g. labeled 'them', and one then further connects this child.

• In a natural language sentence one can imply multiple pieces of information simultaneously. For example, a phrase like '*the* black half of dogs escapes' would not only express the information of case 1, but also tell that (an other) "half-of dogs *not* has-color black" [42] [43], which would need to be captured with a second VSM-sentence in addition.

• Footnote: [44].

[42] Here "has-color" would be attached (as subject) with a relation-less bident to "not" (as object).

[43] [Advanced note]: One might think of representing this as "*exactly* half-of dogs has-color black", instead, inspired by a similar expression in natural language. However, natural language is ambiguous.
- The primary meaning of "exactly" is: exact, as in not more nor less than the stated amount; so: "some precise half"; (opposite of "around", which defines uncertainty about the exactness of the amount).
- A secondary meaning would imply, or intend to incorporate a meaning like: for the other part of the group, the opposite of what is said is true. This is what "exactly"'s meaning/ID would have to be linked to in that sentence. (And from that, an algorithm could infer that "half-of dogs not has-color black", which is the additional information we proposed in the main text already).

It is up to a VSM-box's term-provider whether or not to include such intricate, compacted meanings.

In any case, when dealing with concepts that represent groups, curators will need to learn to pay attention to ambiguities in natural language (or use curation software that could flag such pitfalls). They should at least interpret expressions like "exactly half-of" literally, i.e. as guided by Principle 3. With the primary meaning, it would quite simply express "half-of <has-precision> exact", i.e. '*a/some* exact half'.

[44] The challenge of non-interchangeable connections arose when the designer of VSM heard someone talk about their 'old new me' vs. 'new new me'. Also here, the 'new me' is first contextually isolated, before being referred to again with the extra adjective. | Fig. S8b inspiration: MK's black dog Boomer.

3. Additional functionality

While VSM-terms and -connectors provide the core functionality of VSM, some additional features further increase both the usability and expressivity of VSM. We built some of these functions into the vsm-box UI already, while some others are initial ideas that could be further developed.

3.1. Templates

While VSM enables flexible information capture, some curation tasks involve the repetitive entry of similarly structured facts. To streamline a routine entry of facts, one could use a **VSM-template**. This is a predefined structure of connectors, and a series of VSM-terms, including empty fields that need to be filled out with a term+ID by the user. – In a typical curation project, template designers would accompany a template with set of curation guidelines. These describe what type of term is expected in empty fields, in addition to how to interpret text in relevant scientific literature (which is what requires the most effort). Note that VSM-sentences (including VSM-templates) can closely resemble natural language, which makes it clear for the curator to know what to fill in where, and essentially explicates parts of the adopted curation guidelines. Like this, VSM makes the effort it takes to interact with the information entry software minimal.

Figure S9 shows several templates. Each empty field can be associated with one or more preferred dictionaries, e.g. genes or cell types. This can help the autocomplete function to either rank these terms easily accessible toward the top of the list, or to limit entry to these terms only. Curators need not be restricted to capture only what a template was designed for: because a template is also just an (uncompleted) VSM-sentence, they can still insert additional terms and connect them to the rest of the structure. For broader curation tasks, a curation platform could provide access to several VSM-templates in a menu, each dedicated to a particular information type that the curator may want to capture from a paper. The software may even enable users to design new templates.

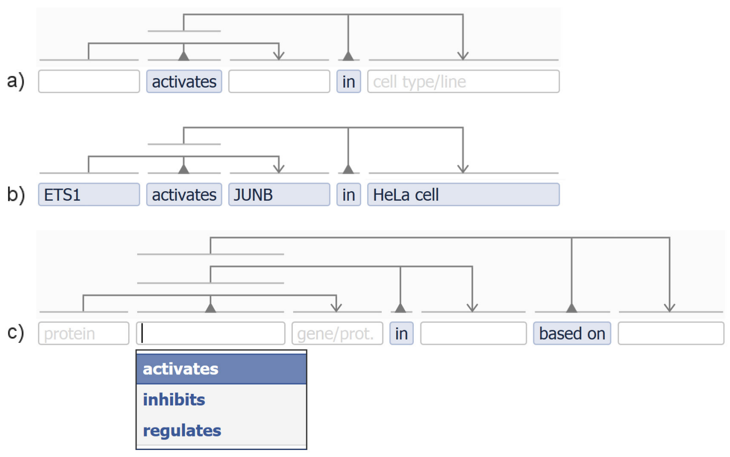

Fig. S9 | VSM-templates. a, A VSM-template in which the first two empty fields would be associated with the lookup of protein terms. The third field expects cell types and cell lines, which this template also clarifies in grey placeholder text. **b**, An example of that template, filled in. The result is a normal VSM-sentence. **c**, A template in which the focused, empty field features a list of easily selectable, one-click terms (pre-associated term+ID couples for that field). The last empty field would limit its term-lookup to a controlled vocabulary of experiment types.

3.2. External coreference

In natural language, one can build a story by referring to any concept that was conveyed in an earlier sentence (e.g. 'John buys cheese'; 'One hour after *that*, a mouse eats *it*'). Because each term in a VSM-sentence is a specific concept, one can refer back to any of them; and not only from within a VSM-sentence but from other VSM-sentences as well. Software for managing multiple VSM-sentences could support such coreferencing between sentences, visually or through internal IDs. This could for instance be used for unambiguously defining the steps of biological experiment protocols.

3.3. Head

As explained in Principle 3, each term receives extra context-meaning from all its connected terms. As a consequence, every term actually represents the entire content of the VSM-sentence, although from its own *perspective*: e.g. in Fig. S2b the fifth term can be read as 'the fork used for eating of chicken by John'. In natural language though, a sentence conveys information only from one particular perspective; e.g. in Fig. S2b, the sentence implies focus on 'the eating'. This perspective or intended main meaning is embedded in the term "eats"; and this term is also the one that subsequent sentences would refer to when stating e.g. "that was observed in 2020". The VSM UI could enable a user to assign this *focal term* or *head* in a VSM-sentence via an Alt+click on or above the VSM-term, and could draw dashes above the term and under the connectors. Alternatively, software could automatically detect this head by analyzing the word-forms chosen for VSM-terms (e.g. a verb, vs. preposition or noun).

Next, when one wants to refer to a sentence as a whole, e.g. for adding meta-information, software could automatically or by default make an external coreference toward this intended term.

3.4. General and Data VSM-terms

VSM-terms (as shown until now) each represent a specific concept, embedded in a particular context. Ontologies, however, define relations between *classes* (categories, sets that generalize over particular concepts), which may also be viewed as general concepts that are not bound to a particular context. In addition, RDF supports *literal* text-strings or data, which do not really represent any concept or ID. In order to support these distinctly different and useful types too, a VSM UI may enable setting a VSM-term's type to one of: Instance (specific), Referring Instance (specific, referring child), Class (general), or Literal (raw data or text); and show these in a distinct rectangle color or style (blue, dashed-border blue, yellow, and red, resp.). A VSM-sentence can then express e.g. "duck [in-general] is-a-type-of [specific] bird [general] according-to [specific] ontology-version-X [specific]", or "chicken [general] has-alias [specific] hen [literal-string]", or "proteinX has-sequence MEEPQSDPSVEPPLS [literal]".

Note that in case one would use VSM as an interface for generating RDF or OWL[45] models, VSM does not impose the use of either OWL Instances or OWL Classes. Still, it may make sense to translate the default, specific VSM-terms to OWL Instances, as bioscience findings are typically reported in the context of particular experimental or biological conditions.

A further treatise on connecting the meaning of VSM concepts to e.g. formal logic is outside the scope of this paper. This paper presents VSM as a user-oriented method for meaningful construction of contextualized knowledge. It explains VSM's human-thought inspired semantics primarily in a curator-oriented form, and describes it together with the enabling, intuitive, general-purpose UI design.

[45] OWL Web Ontology Language: https://www.w3.org/TR/owl-ref .

Supplementary Discussion

i. Novelty

VSM is an entirely new and simplified approach for representing and capturing knowledge. It makes it easier for curators to make the shift from thinking in natural language to formulating in structured statements. We described how to build VSM-sentences: diverse statements that can contain highly contextualized information, yet in a form that is both easily readable by humans and meaningfully processable by algorithms. VSM-terms enable a user to disambiguate concepts with the help of ontologies and identifiers. VSM-connectors enable a user to disambiguate the internal structure of information, and support a uniform and intuitive method for creating statements with full structural transparency. Tridents, bidents, and lists cover many use cases; coreferences support also specialized cases. The three VSM Principles provide conceptual descriptions for working with terms and connectors. Templates make VSM as easy as filling out spreadsheets or forms, or easier because of autocomplete and CV support. Some additional features make VSM even more expressive.

VSM thus forms a bridge between computers and human understanding. This forms the foundation for precise computational comprehension and meaningful processing of extremely diverse, human-readable quasi-sentences. It provides a solution to a long-standing problem for curation in the biosciences, and likely beyond.

VSM is most closely related to: controlled languages, RDF, and form-based entry methods. The table in main-document Fig. 5 compares VSM with several knowledge representation or analysis methods, on aspects like ease-of-use and semantic internals. Figure S10 details a further comparison with RDF.

ii. Continued implementation

To demonstrate our ideas for an interactive VSM UI, we first implemented a small prototype. We used it among others for creating figures with VSM-sentences in this text, and for interactive examples on accompanying web pages. Since then, we already rebuilt much of this work into well-designed software modules that can be reused in any curation project (this implementation will be detailed elsewhere)[46]. These modules have a modern design that now supports community-based co-development of the many more possible features, and of supporting and downstream modules.
A fully usable software suite should not only provide an interface for entering VSM-sentences, but also provide support for e.g. storing and querying them. Storage could be done in several ways: for instance as JSON-based graphs, or directly into a graph database. In addition, a VSM-to-RDF mapper that automatically produces RDF graphs (which was now done manually in Fig. S10) will have the advantage

[46] The code is in the Vsmjs organization on GitHub, and is open to contributions. Currently its three most central modules are: vsm-dictionary, vsm-autocomplete, and vsm-box.

of making VSM-sentences readily queryable with SPARQL[47] [48]. Also, a tool for building queries that directly use VSM-connectors or VSM structures as graph patterns for querying, could help users avoid the complexity of RDF reifications.

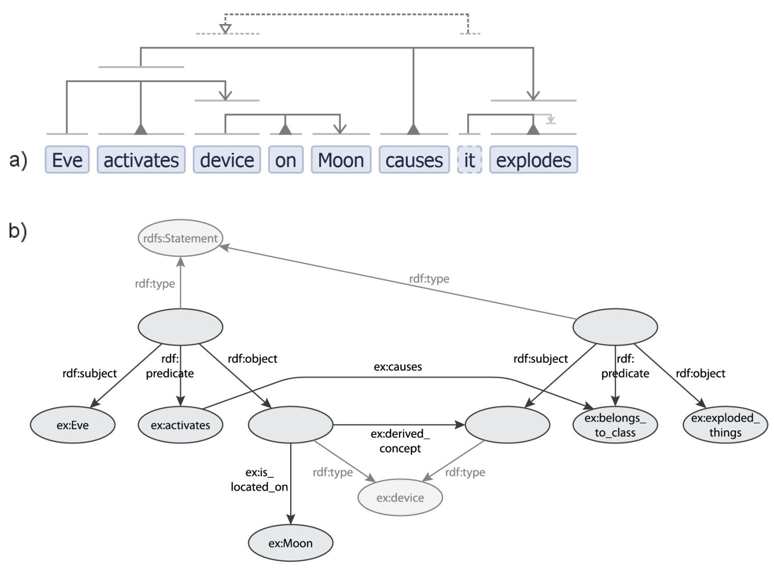

Fig. S10 | Example conversion from VSM to RDF: a VSM-sentence is easier to understand than RDF. The VSM-sentence in (**a**) was manually converted to (**b**): a possible graph representation in RDF, using an example controlled-vocabulary 'ex'. The relation 'activates' is used as the subject of another triple and thus needs e.g. a *reification* construction in RDF. Here, both representations show the two distinct 'device' concepts: one only located on the Moon and being activated but not yet exploding, the other one exploding as well. The bident-part "it explodes" needs to be expanded into a triple in RDF. The relation 'ex:causes' could (as a choice) either connect the nodes 'ex:activates' and 'ex:belongs_to_class', or connect their reified *blank nodes*. Note that RDF mixes together the supporting or *primitive* relations ('rdf:…') and many (but not consistently all) relations that are part of the actual or expressed information ('ex:…'); while VSM keeps these apart as connectors and terms resp., which is a form of elegance. For most people, the VSM-sentence is much easier to read than the RDF graph.

iii. Initial adoption: use with templates

For curators. Learning new things always poses some sort of threshold. As a first introduction to VSM, a curation project could start off by having new curators use only templates. As most people are familiar with spreadsheets or forms, the use of VSM-templates does not require a steep learning curve: the curation process uses readable statements that are essentially self-explanatory as to what belongs in which empty field, and it is comfortably augmented with ontology lookup and term auto-completion. More experienced curators who encounter the need to capture information that goes beyond a template, may gradually venture into learning more about VSM and add and connect extra terms and structure. In a pilot curation project, we witnessed exactly this. Early adopters also appreciated VSM for 'making the biological information's inherent complexity much easier to handle'.

[47] A proposed straightforward mapping to RDF is already being developed in the Vsmjs organization.
[48] SPARQL Protocol and RDF Query Language: https://www.w3.org/TR/sparql11-overview.
Ref.: Prud'hommeaux E, Seaborne A. "SPARQL query language for RDF", 2006.

For IT experts. Templates are also a good start for being able to write queries in the standard way, which is: relying on a data-schema that specifies precisely what kind of information is available for querying. Still, as a curator captures information that exceeds the template (e.g. extra details about a protein), a query could return a term, plus its attached subgraph. This would be a VSM-fragment that is already conveniently human-readable. If a certain type of extension occurs often, queries could be updated to search specifically for that information too. So instead of having a data-format that dictates what curators may capture, VSM also enables a curator-driven process, where captured information may steer query development (and standardization) too. This prospect enables a shift of focus onto semantic preservation first, which may then guide or motivate for processing technology to catch up.

With the community. Templates may also be useful to ensure that captured information is readily translatable into other data formats. For example, we have been using templates in our pilot curation project, and these are helpful to export relevant parts of VSM-sentences and to contribute to established projects like GO Annotations and IntAct.[49] Furthermore, the user-friendliness of the interface could stimulate the design of dedicated templates for entry of knowledge that is readily exported toward other formats like BioPAX, SBGN, SBML, or GO-CAM.[50] Validity checks or constraints could accompany these templates to ensure compliance. Templates may also function as easy-to-conceptualize, unifying glue between different curation platforms' particular representations.

With text-mining. When natural language processing (NLP) algorithms extract specific information types from literature, the resulting extracted facts could be automatically entered into associated VSM-templates. These draft VSM statements may then be curator-validated, corrected and/or extended. Alternatively, manually filled templates may serve as a training set for NLP. As an easy-to-use curation technology, VSM may engage more scientists in creating training data, and so accelerate a virtuous circle of curation/NLP co-improvement.

iv. From template-guided use toward full use of VSM

In order to move from templates to using VSM's full flexibility, there are some challenges to overcome.

Relation terms. The first challenge is how to deal with missing terms: while controlled vocabularies are typically incomplete, VSM has a particular need for terms and IDs for relation concepts, which may often not yet be available. A first solution would be to enable curators to build term lists for local use, which are then flagged as new entries that should be considered by ontology experts to be included in shared controlled vocabularies.

Paraphrasing. A second challenge relates to structural flexibility: there is freedom to express the same piece of information in multiple ways, with different levels of structural detail. For example, the single

[49] • GOA: http://geneontology.org/page/go-annotations | • IntAct: https://www.ebi.ac.uk/intact .

[50] • BioPAX (Biological Pathway Exchange): http://www.biopax.org ,
Ref.: "The BioPAX community standard for pathway data sharing", Nat Biotechnol, 2010.
• SBGN: http://sbgn.org , Ref.: "The Systems Biology Graphical Notation", Nat Biotechnol, 2009.
• SBML: http://sbml.org , Ref.: "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models", Bioinformatics, 2003.
• GO-CAM: http://geneontology.org/docs/gocam-overview , Ref.: "Gene Ontology Causal Activity Modeling (GO-CAM) moves beyond GO annotations to structured descriptions of biological functions and systems", Nat Genet, 2019.

term “liver-cell” is equivalent to the more structurally transparent triple “cell in liver”, and both expressions will likely be entered by curators, especially when not guided by templates. This calls for mechanisms that can map structures with equivalent meaning, in order to retrieve all variants that semantically match a particular query. Still, this process will be less complicated than finding all variants that paraphrasing allows in natural language, as VSM presents the knowledge already in clearly defined graph structures.

Assistance. Swift and nimble curation with full VSM may be achieved with assistance tools. First, intelligent autocompletion may be trained, to suggest e.g. common prepositions after entered relations. Also, NLP techniques could suggest and add VSM-connectors. Ultimately, algorithms could suggest partial or full translations of free text (or figures) to VSM. This may be accelerated by compiling a set of VSM translations for common free-text phrases. In the long term, a VSM-sentence may serve as an interface to show how a machine understood a free-text sentence.

v. Potential scale of use

We envision that VSM will first be applied in new curation projects by scientists who would otherwise use spreadsheets or need to develop their own user interface. Such practical use-cases are instructive to inspire new ways forward. Another immediate application of VSM is to display various kinds of computationally generated, detailed knowledge in a human-manageable way. For instance, VSM may present: 1) the results of a text-mining scan over a text corpus; or 2) the detailed interpretation of what an algorithm understood from text (e.g. a scientific publication, or a social media post); or 3) a readable statement about a pattern detected in data by a machine learning or other algorithm, in a semantic form that could subsequently be queried or perhaps reasoned upon. This demonstrates that we are only at the beginning of exploring how VSM can be applied in various use-cases, so it is important to recognize that this publication is just the first, promising step.

We like to point out that VSM’s expressiveness enables a general-purpose curation platform, rather than the current single-purpose ones. VSM’s ease-of use may also facilitate a general crowdsourcing approach, to cover entire biological domains or indeed the entire life sciences. Although this would require a substantial investment in software development, and also a wide community engagement, it could pave the way to a platform that allows for full digital summaries of any research paper. VSM-based curation thus would provide the scientific community with the tools for converting new and existing knowledge into a structured knowledge resource. The content of a paper could be presented in what resembles a wiki page, but with information that is both human- and computer-understandable. While *structured digital abstracts* have long been a dream, VSM provides a major step in that direction.