

Tumbug: A pictorial, universal knowledge representation method

Mark A. Atkins, Ph.D.
aginik@outlook.com

December 26, 2023

Abstract

Since the key to artificial general intelligence (AGI) is commonly believed to be commonsense reasoning (CSR) or, roughly equivalently, discovery of a knowledge representation method (KRM) that is particularly suitable for CSR, the author developed a custom KRM for CSR. This novel KRM called Tumbug was designed to be pictorial in nature because there exists increasing evidence that the human brain uses some pictorial type of KRM, and no well-known prior research in AGI has researched this KRM possibility. Tumbug is somewhat similar to Roger Schank's Conceptual Dependency (CD) theory, but Tumbug is pictorial and uses about 30 components based on fundamental concepts from the sciences and human life, in contrast to CD theory, which is textual and uses about 17 components (= 6 Primitive Conceptual Categories + 11 Primitive Acts) based mainly on human-oriented activities. All the Building Blocks of Tumbug were found to generalize to only five Basic Building Blocks that exactly correspond to the three components {O, A, V} of traditional Object-Attribute-Value representation plus two new components {C, S}, which are Change and System. Collectively this set of five components, called "SCOVA," seems to be a universal foundation for all knowledge representation.

Contents

1 Introduction

6

2	Motivation	9
3	Overview and background of Tumbug	11
4	Derivation of Tumbug	13
4.1	Knowledge representation methods	13
4.1.1	The importance of KRMs in AI	13
4.1.2	Typical tradeoffs of KRMs	15
4.1.3	Clues to the type of KRM the brain uses	35
4.1.4	Ranking of KRM methods	38
4.1.5	Why dependency on learning is misguided	40
4.1.6	Criteria and heuristics that make Tumbug seem particularly promising	40
4.2	Applicable systems from the real world	47
4.2.1	Physics	47
4.2.2	Mathematics	49
4.2.3	Other systems and the Big Picture	50
5	Object-like Building Blocks of Tumbug (O)	55
5.1	Singleton objects	55
5.1.1	Object Circles	55
5.1.2	State Circles	64
5.1.3	Cells	65
5.1.4	Pathway Tubes	65
5.1.5	Sensor Bars	66
5.1.6	Markers	68
5.2	Collection objects	74
5.2.1	Location Boxes	74
5.2.2	Swirly Arrays	91
5.2.3	XOR Boxes	93
6	Attribute-like Building Blocks of Tumbug (A)	100
6.1	Attribute Lines	100
7	Value-like Building Blocks of Tumbug (V)	104
7.1	Value Bars	104
7.1.1	General	104
7.1.2	Ranges	105

7.1.3	Example: Goals	109
7.1.4	Geometrical variations	109
7.2	Wildcards (further development anticipated)	111
7.3	Range Caps	121
8	Change-like Building Blocks of Tumbug (C)	121
8.1	Single Time Arrows	121
8.2	Motion Arrows	125
8.2.1	Motion of objects	125
8.2.2	Streams of objects	128
8.2.3	Shorthand notation	128
8.3	Force Arrows	133
8.4	Causation Arrows	134
8.4.1	Single causes	134
8.4.2	Multiple causes	139
8.4.3	Labeled causes	140
8.5	Correlation Boxes	140
9	System-like Building Blocks of Tumbug (S)	145
9.1	State Diagrams	145
9.1.1	Overview	145
9.1.2	Common pairs of states	149
9.1.3	Applications	149
9.1.4	The importance of avoidance of assumptions	153
9.2	Split Time Arrows (further development anticipated)	154
9.3	Data Set Boxes (further development anticipated)	157
10	Rules for combining the single Building Blocks	159
10.1	Nonquans with Change Arrows	159
10.2	Attribute and Values	161
11	Convenience Building Blocks of Tumbug	162
11.1	Label Strings	162
11.2	Attend Rings	165
11.3	Motivation Triangle	167
11.4	The Modal Verb Icon	172
11.5	The Robinson Icon	172
11.6	Zoom Boxes	181

11.6.1	General	181
11.6.2	WS150 example: #52 (worm)	182
12	Generalized Building Blocks of Tumbug	183
12.1	Nonquantified Objects (= Nonquans)	184
12.2	Interchangeably Actualizable Maps (= IAMs)	185
12.3	Change Arrows	185
13	Heuristics for converting sentences to Tumbug	186
14	Relationship to prior work	193
14.1	Conceptual Dependency theory	193
14.1.1	Building Blocks	193
14.1.2	Slots	194
14.1.3	Primitive Acts	195
14.1.4	Overview of comparison	205
14.2	English grammar representation	213
14.2.1	Kolln sentence patterns	213
14.2.2	Allerton sentence patterns	236
14.3	Object-attribute-value (= OAV) triples	239
14.4	Vector graphics editors	240
14.5	Semantic networks	246
14.5.1	Kinship relations	246
14.5.2	Specific objects versus generic objects	247
14.5.3	Intensions versus extensions	250
15	Applications of Tumbug	250
15.1	Natural language translation	250
15.1.1	Conversion from a natural language to Tumbug	252
15.1.2	Discussion of language translation	254
15.2	Mathematics: Arithmetic	259
15.3	Logic: First Order Predicate Calculus	263
15.3.1	Syllogisms with Venn diagrams	263
15.3.2	Barbara form (AAA-1)	264
15.3.3	Celarent form (EAE-1)	267
15.3.4	Darii form (AII-1)	271
15.4	Computer language representation	271
15.4.1	Sequential code	273

15.4.2	Looped code	277
15.4.3	Branching code	277
15.5	Natural language representation	277
15.5.1	Categories of verbs	277
15.5.2	Active tense versus passive tense	288
15.5.3	Grammatical time aspects	291
15.5.4	Modal verbs	299
15.5.5	Propositional attitudes	312
15.6	Tumbug as software	317
16	A few deep implications of Tumbug	321
16.1	SCOVA: an extension to OAV triples	321
16.2	Mathematical implications	323
17	Insights from Tumbug	324
17.1	CD theory and Parts of Speech	324
18	Pros and cons of Tumbug	329
18.1	The main strengths of Tumbug	329
18.2	Possible weaknesses of Tumbug	329
18.3	Programs writing programs	331
19	Discussion of Tumbug	333
20	Summary	335
21	Future Research Plan	337
21.1	The Visualizer Project	337
21.2	Roadmap of the next phases	338
21.2.1	Phase 2 (non-spatial reasoning)	338
21.2.2	Phase 3 (spatial reasoning)	339
21.2.3	Phase 4 (learning)	339
21.2.4	Phase 5 (visualizer)	339
22	References	339
23	arXiv response	347

1 Introduction

A major underlying problem with all natural language translation is that there does not exist any universally accepted knowledge representation method (KRM) for all natural languages. This problem in turn appears to be based on a more fundamental problem: No known method exists, whatsoever, of representing semantics (meaning) of sentences in any natural language.

Figure 1 shows that a universal representation would require exactly two steps for any language conversion: one step from the source language representation to the universal representation, then one more step to the target language representation. Only 1-2 types programs would be needed, namely 1-2 program(s) that converted between the universal representation and any specific language representation. This is a diagrammatic way to express an "interlingual conceptual base," as Roger Schank described it (Schank 1972, pp. 553-554).

Figure 2 shows an extreme alternative: only one step would be needed to convert from the source language representation to the target language representation, but six total programs would be needed for arbitrary conversion between three languages. In the first alternative, n languages would always require only 1-2 types of programs, whereas in the second alternative, n languages would always require $n^2 - n$ separate programs. The first type of program would be concerned with only one type of mapping, the second type would be concerned with n types of mappings, which suggests the first type scales better and generalizes better, and is therefore ultimately a better solution that saves on computer coding effort. Another motivation for the universal approach is that it would likely lead to insights into the nature of language in general, which in turn might also lead to insights into the relationships between language and other fields such as physics, math, computer programming, and neuroscience.

This article presents a new KRM named "Tumbug" that appears to be universal for representing semantics, and therefore can represent any given sentence in any natural language as a single data structure with appropriately associated values. This structure-with-values combination appears to be invariant for every language used to write a sentence that has the same meaning, provided that the corresponding words have exactly the same meaning across the two languages, and provided that all attributes of the corresponding words across the two languages are the same. Thus Tumbug may implement the "universal representation"—the center circle of the three lan-

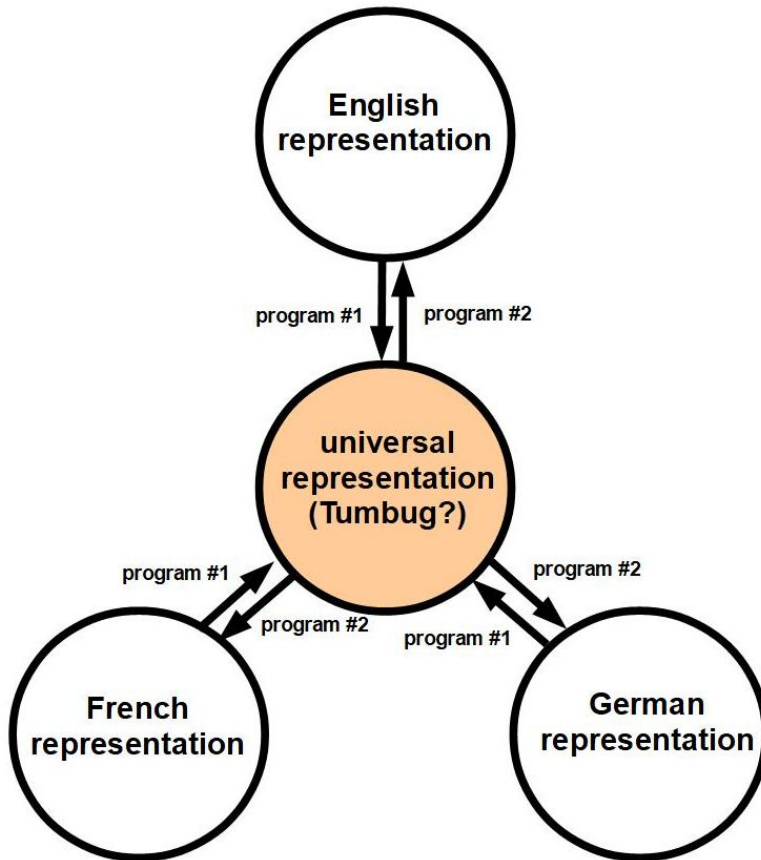


Figure 1: First alternative for language translation: This strategy uses one universal representation, and would need only two programs for three languages.

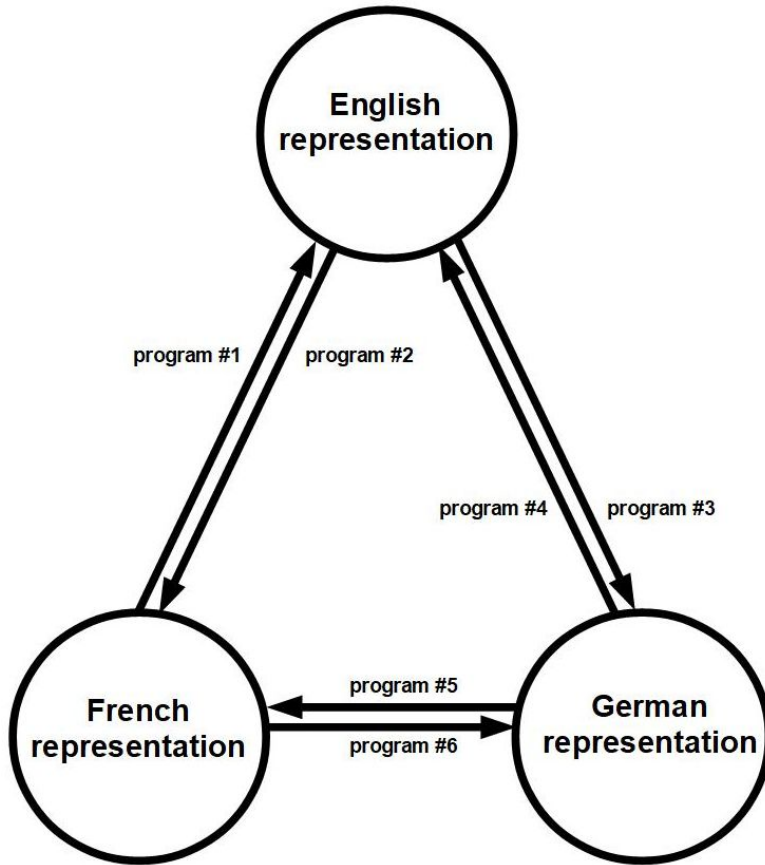


Figure 2: Second alternative for language translation: This strategy uses no universal representation, and would need six programs for three languages.

guage circles shown in Figure 1.

Emphasized clarification: This is a theoretical study, not an empirical study. Misinterpretation of the nature of this study probably arises from the common misinterpretation of the term "artificial intelligence" to mean "machine learning," but machine learning is only one subfield within AI, and the study in this document is not about the subfield of machine learning.

2 Motivation

The overall goal of this research is to advance progress in artificial general intelligence (AGI). AGI is the ultimate goal of the larger field of AI in general, although currently only the practical, applied aspects of AI—the subfield of AI called artificial narrow intelligence (ANI)—is well understood, already implemented, and making obvious progress. In contrast, the subfield of AI called AGI is currently languishing and awaiting new ideas that will vitalize it so that it too can be applied to real-life problems. Since AGI progress is usually claimed to be based on Common Sense Reasoning (CSR) progress (e.g., Davis 1990, p. 1; Devlin 1997, p. 167), the most promising subfield of AGI to tackle appears to be CSR. Within CSR, one well-known test of machine understanding of written language is the Winograd Schema (WS). Therefore example sentences from the WS will be used in this study to generate generalities, insights, and examples into the data structures needed for such a universal semantic KRM, and these insights will be implemented as a KRM called "Tumbug." This KRM therefore has the potential to advance the field of AGI. Figure 3 shows the relationship of the main subfields of AI.

Emphasized clarification: The author has never had any involvement with the Winograd Schema Challenge (WSC), a competition that no longer even exists. The WS is used here only as a convenient, well-known list of benchmark problems in CSR, and its presence in this document should not be interpreted to mean that this research refers to some past, present, or future submission to the WSC by the author.

A comment about notation: There exist various versions of the Winograd Schema, each of which has a designation "WS" or "WSC" followed by the number of sample problems in that schema (list). The WS version used in this document is called "WS150" because it has 150 problems (Davis 2018), though that designation has not been used elsewhere. Some other named versions of the WS are WSC273 (Kocijan et al. 2022, p. 2), which has 273

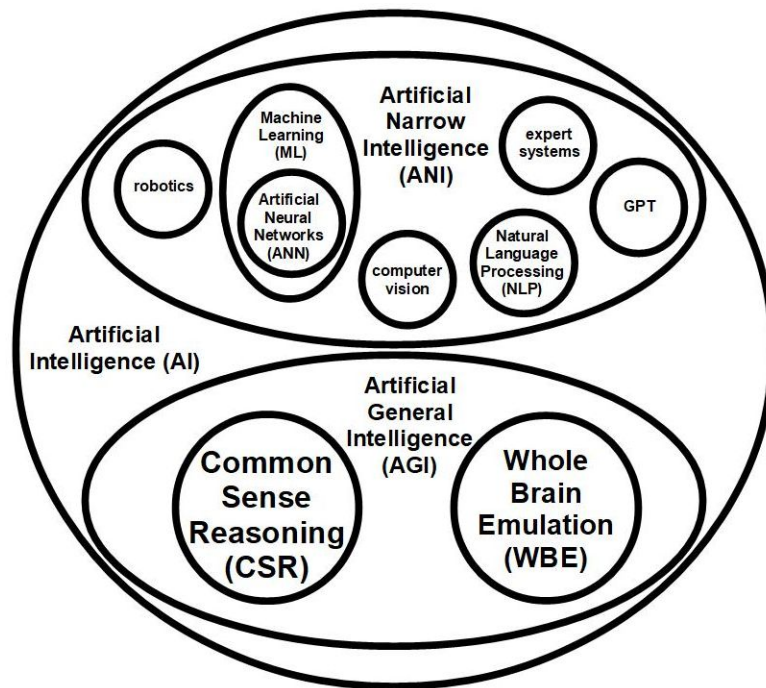


Figure 3: The field of AI is divided into ANI and AGI, and in turn the field of AGI contains CSR. There is currently little or no overlap between ANI and AGI except in trivial ways such as hierarchies or the massive training of large language models (LLMs).

problems, and WSC285 (Kocijan et al. 2022, p. 6), which has 285 problems.

This article's convention: WS150 problems quoted in this document have the second word of the word pair omitted, for simplicity, clarity, and readability. For example, the WS150 sentence "The city councilmen refused the demonstrators a permit because they [feared/advocated] violence" would appear in this document as "The city councilmen refused the demonstrators a permit because they feared violence."

3 Overview and background of Tumbug

The author developed Tumbug from 2021-2023 in an attempt to universalize all the grammars of all the natural languages for the purpose of making foreign language learning easier. The word "Tumbug" is a simpler, more readable version of the acronym "TUMBVG" (Temporal Universal Model-Based Visual Grammar). The term "visual" in the acronym means that diagrams are used in this KRM instead of text, numbers, logic symbols, or statistics, which makes Tumbug an extremely unique KRM of a type that no one appears to have tried yet. The term "model-based" means that this KRM is intended to represent arbitrary models of systems, whether a mechanical system of gears and pulleys, a biological system of organs and bodily fluids, a physical system such as planets orbiting a star, or any other type of system. The term "temporal" means that the models represented also include time, as opposed to static diagrams drawn on paper. Although Tumbug diagrams are typically drawn on paper, Motion Arrows and Time Arrows in Tumbug diagrams show where and how the components are moving relative to each other, which in a good software implementation would literally appear as moving images, such as via simulations or animated GIF files. The word "grammar" originates from Tumbug's originally intended use, namely language translation based on sentence grammatical structure, but Tumbug can be applied to any system whatsoever.

Figure 4 is a taxonomy of the main knowledge-based approaches to CSR, based mostly on a list by Davis and Marcus (2015, p. 11) and a very similar list with more details by Huang et al. (2020, p. 93), but organized here by KRM: mathematics versus algorithm versus images. The highlighted category "Image-based" that includes "Tumbug" is new to this diagram and is the author's approach promoted in this article. Note that Tumbug is not only a completely new approach, but is also an entirely new class of approach

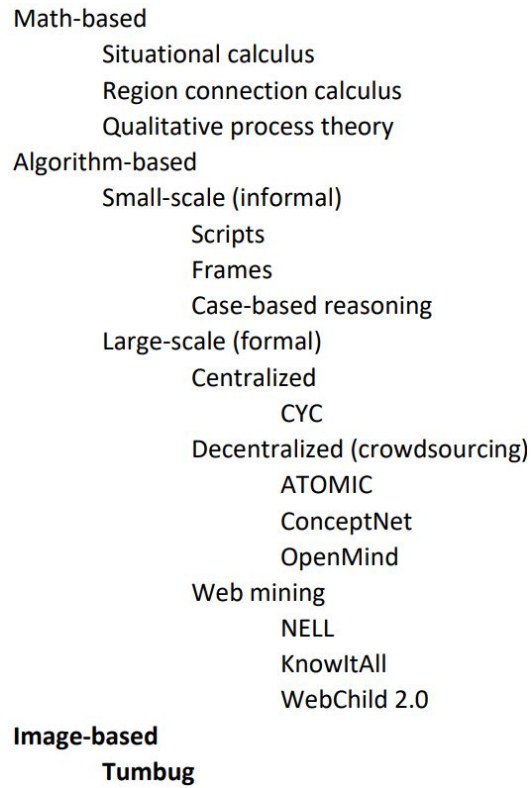


Figure 4: A summary of knowledge-based approaches to CSR. Tumbug is so unique that it has its own category.

for CSR, one that has never been explored yet. This situation suggests that Image-based CSR may contain much new ground for exploration.

So far all code-based approaches have been described as "informal." Tumbug might also be considered "informal" because there is some flexibility in how the visual structures are pieced together, although not much.

4 Derivation of Tumbug

4.1 Knowledge representation methods

4.1.1 The importance of KRMs in AI

The fact that KRMs are extremely important for AI is known to surprisingly few researchers in the field. Related to this, very few AI researchers are working on KRMs.

- Marvin Minsky said, 'Well, I'd like to have lots of people thinking about how to *combine* different approaches. It's not the different approaches themselves: it's 'Why aren't there more people making a machine that uses three different representations of knowledge and crosses over?' That's a very specific kind of research project, and I see no one doing it. So that's, to me, that's the missing link.'" (GBH Archives 1990)
- Herbert L. Dreyfus wrote, "A second difficulty shows up in connection with representing the problem in a problem-solving system. It reflects the need or essential/inessential discrimination. Feigenbaum, in discussing problems facing artificial intelligence research in the second decade, calls this problem 'the most important through not the most immediately tractable.'" (Dreyfus 1979, p. 298)
- Han Reichgelt wrote, "Knowledge representation is a major concern in Artificial Intelligence and its importance cannot be overestimated." (Reichgelt 1991, p. 1).
- John Sowa wrote, "A knowledge representation is a fragmentary theory of intelligent reasoning". (Sowa 2000, p. 134) This suggests that a KRM itself, even without being used in a computer program, is already a partial model of intelligent reasoning.
- Laurence Harris and Michael Jenkin wrote, "The representation of knowledge is fundamental to human cognition and machine understanding" (Harris and Jenkin 1997, p. 2).
- Heather Jenkin wrote, "Fundamental to all human cognition is the representation of knowledge" (Jenkin 1997, p. 268). It is therefore not

surprising that the field of knowledge representation arose from the field of AI (Sowa 2000, p. XI; Cercone 1987, p. 3).

- Haugeland wrote: "In other words, Artificial Intelligence must start by trying to understand knowledge (and skills and whatever else is acquired) and then, on that basis, tackle learning." (Haugeland 1985, p. 11).

There have been cases in the history of science when merely switching to a different KRM was the key to major insights and rapid progress, if not immediate solution to a single difficult problem. Some examples are: (1) The switch from the awkward rules of Roman numbers to Arabic numerals, especially when augmented by use of the placeholder digit "0." (2) Use of Feymann diagrams to ease visualization of particle interactions in physics (Fischler and Firschein 1987, p. 69). (3) Standardization of business information in data base format for consistent, non-redundant storage and ease of access through query languages. (4) The switch from a 1D list of elements to the shaped 2D periodic table of the elements. (5) Use of phase portraits to visualize chaotic systems (Gleick 1987, p. 317). (6) Discovery that the KRM of genetics was a long chain of paired nucleobases from the set A, C, G, T, where A pairs only with T, and C pairs only with G, within each link of the chain.

As for some history on the terminology, the term "knowledge-representation method" dates back to at least 1974 with Marvin Minsky (Minsky 1974, p. 28). The abbreviation "KRM" for this term is still occasionally found on the Internet (in 2023). More recent terminology includes the term "knowledge representation language" (KRL) (e.g., Reichgelt 1991, p. 3; Luger and Stubblefield 1998, p. 293), but that term is avoided here because "languages" are usually assumed to be textual whereas Tumbug is not, which makes the term somewhat misleading when applied to visual KRMs such as Tumbug.

The concept of knowledge representation is very close to the concept of data structures in computer science. Technically a data structure is a structured collection of data and a set of routines to create and manipulate that data, whereas discussion of knowledge representation methods tends to ignore any associated routines. Associated routines are generally ignored in this document, as well.

Emphasized clarification: Tumbug is a knowledge representation method, not an algorithm or program, therefore the system is inherently incapable of

learning in its current, non-software form. Misinterpretation of the nature of this study probably arises from the common misinterpretation of the term "artificial intelligence" to mean "machine learning," but machine learning is only one subfield within AI, and the study in this document is not about the subfield of machine learning.

4.1.2 Typical tradeoffs of KRMs

1. General properties

Numerous KRMs abound in the sciences, especially in computer science, and even in games such as chess. In general, KRMs tend to share the following general properties:

1. Each KRM has at least one competing KRM that can represent the same knowledge.
2. Any given KRM will be inefficient in some way that one of its competing KRMs will not.
3. When a system is rendered in a KRM that is particularly suitable for a given type of problem, algorithms become highly simplified in that KRM. In other words, a good KRM can solve the hardest part of any given problem due to high efficiency arising from the closeness of the system's structure with the system's description.
4. A KRM that attempts to be suitable for solving too many problems at once becomes impractically large in its notational system. This is an example of a time-space tradeoff: increased speed comes at the cost of increased representation size.
5. KRMs that use "exotic isomorphisms" (*verbum* Hofstadter 1979, p. 159) are typically so unwieldy that they are impractical for computational systems. Some examples of exotic isomorphisms are: (1) DNA structure to represent phenotype structure (Hofstadter 1979, p. 160), (2) cellular automata to represent numerical computations (Wolfram 2002, p. 640), and (3) an energy function to "program" a continuous Hopfield network to perform the appropriate optimization. Related to exotic isomorphisms is the desirability of a representation to remain stable when the represented situation changes slightly (Fischler

and Firschein 1987, p. 69), since violation of this desideratum results in discontinuous mappings. This strongly suggests that complexity is another key constraint on system efficiency, in addition to the better known constraints of time, space, and energy. The opposite of "exotic isomorphism" is "prosaic isomorphism" (*verbum* Hofstadter 1979, p. 159). In other words, human designers should design only systems that use prosaic isomorphisms.

More specifically, applied to the question of which type of KRM the brain uses, these general properties can be interpreted as:

1. The brain's KRM could be any one of multiple possibilities, including a non-numerical KRM. This is exactly what Tumbug proposes: a non-numerical KRM that the brain might realistically use.
2. A non-numerical KRM will likely be efficient at solving non-numerical problems but inefficient at solving numerical problems. This is exactly the difference that has been noticed since the early days of computer science: humans are fast at spatial reasoning but slow and error-prone at numerical reasoning, whereas digital computers are slow and error-prone at spatial reasoning but fast and accurate at numerical reasoning.
3. A KRM that is a good candidate for the brain's KRM might solve the hardest part of CSR, which might result in the associated algorithms being quite simple. This is exactly what the author claims: that Tumbug renders WS problems easily solvable via very simple algorithms. In other words, the hardest part of the CSR problem might be easily solvable by a suitable KRM.
4. A universal computing machine is the wrong approach for AGI. This is exactly what the author claims: that digital computers will always have their useful niche of solving large or intricate discretized problems (such as graph problems, data bases, and number crunching) whereas a non-numerical machine would likely have its niche in non-numerical problems with noisy data. In other words, a non-computational/non-computer-like architecture is likely the solution to AGI problems.
5. Exotic isomorphisms should be eschewed as a design heuristic when developing intelligent machines, otherwise such machines will be too

difficult to program and to understand. This is one reason why the author believes that that first practical, intelligent machines cannot be based on cellular automata, chemistry, or quantum computers. In other words, the first intelligent machines should use prosaic isomorphisms.

2. Examples of pairs of competing KRMs

In each of the three domain examples of this section, note that there exist two main, competing KRMs, the "best" of which depends on the type of information elicited, and that both KRMs fail to be useful on the last question, which means there likely exists a third KRM that is applicable and efficient. There may exist yet more applicable KRMs beyond these three that are not discussed.

2.1. Chess moves: algebraic notation versus descriptive notation

The earliest successful symbolic KRM for chess moves is called "descriptive notation." For example, the most common first move on the chess board would be called "P-K4," or "pawn to king 4," which means that the pawn in front of the king (this pawn starts on the 2nd row of the chess board) moves to the 4th row of the chess board. In contrast, the modern, favored KRM for chess moves is "algebraic notation." For example, the move that is called "P-K4" in descriptive notation would be called "e4" in algebraic notation, which means that the pawn in question (a pawn is implied because no piece letter precedes the pair of characters) moves to the square represented by "e" (= the 5th column from the left) and "4" (= the 4th row from the bottom). See Figure 5 and Figure 6.

1. **Question:** In the move by White, KP \times QP (or exd5), what is the destination square of the capturing pawn?

The best KRM to generate the answer: Algebraic notation, since the answer d5 is already provided in that notation, and this answer cannot be determined by descriptive notation, which can only list all five possibilities, namely exd3, exd4, exd5, exd6, exd7.

The simple algorithm used: For move Xij, where X = the unit letter (N, B, R, Q, K, or nil for pawn moves) with a possible capture sign (= "x"), and where ij are the last two characters in the string, the destination square is (i, j).

Answer, and how it was generated: For the move Xij = exd5, ij = d5, which is the answer.

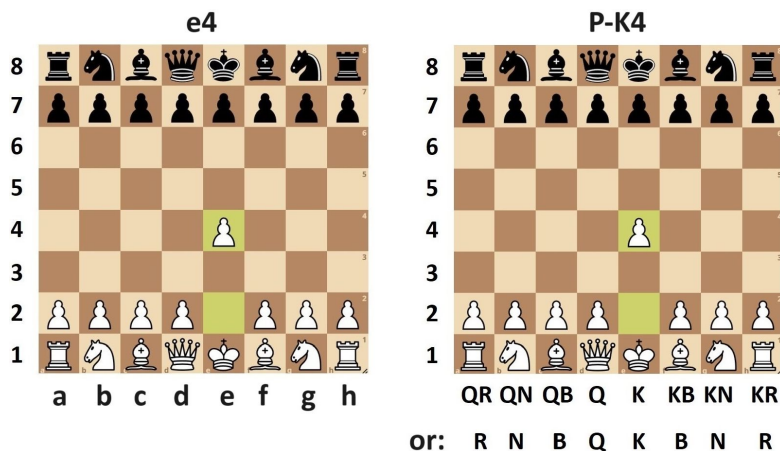


Figure 5: Examples of two KRMs for chess: The shown chess move can be represented as "e4" in algebraic notation, based on the indices on the left, or as "P-K4" in descriptive notation, based on the indices on the right. Which is "best" depends on the type of information elicited.

2. **Question:** How would one describe the moves P-QR3 and P-KR3 in general, regardless of the player and the side of the board?

The best KRM to generate the answer: Descriptive notation, since the answer is the intersection of the two move representations, namely P-R3, whereas algebraic notation can only list all four possibilities, namely a3, h3, ...a6, ...h6.

The simple algorithm used: For moves X and Y, the general description of that move is $X \cap Y$, where the result treats any nils as empty strings that will not show when concatenated.

Answer, and how it was generated: For the two moves P-QR3 and P-KR3, we calculate $P\text{-QR3} \cap P\text{-KR3} = P\text{-R3}$, which is the answer. (See Figure 6.)

3. **Question:** In the move P-K4 (or e4), from which square did the pawn originate?

The best KRM to generate the answer: Neither, since neither notation carries this information. Descriptive notation can only list all two possibilities, namely K2-K4, K3-K4, and algebraic notation can

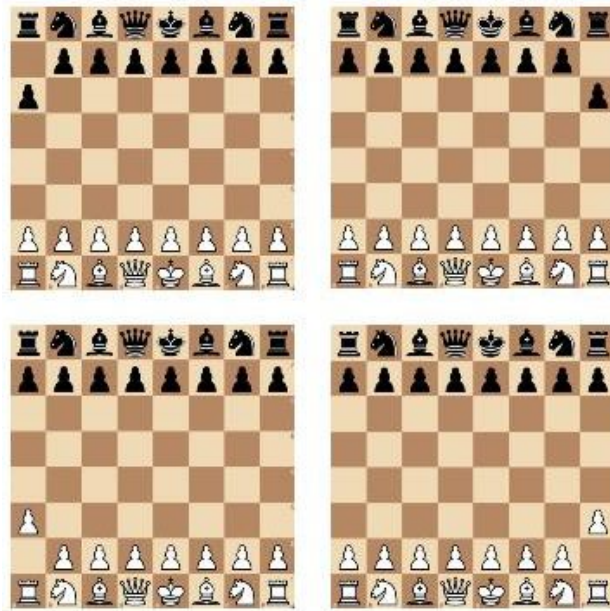


Figure 6: The more general chess move called "P-R3" can refer to either side of the board (kingside or queenside) and to either player (White or Black). This diagram shows all four possibilities.

<u>move #:</u>	<u>algebraic notation:</u>	<u>descriptive notation:</u>	<u>hypothetical combined notation:</u>
1.	e4	P-K4	KP@e2-KE@e4
1...	c5	P-QB4	QBP@c7-KE@c5
2.	c3	P-QB3	QBP@c2-QBE@c3
2...	d5	P-Q4	QP@d7-QE@d5
3.	d4	P-Q4	QP@d2-QE@d4
3...	exd5	PxP	KP@e4xQP@d5

Figure 7: The mentioned hypothetical combined notation clearly shows that increasing the power of the representation (in this case by including more information) tends to increase the size of the representation, in this case in the form of a longer textual description.

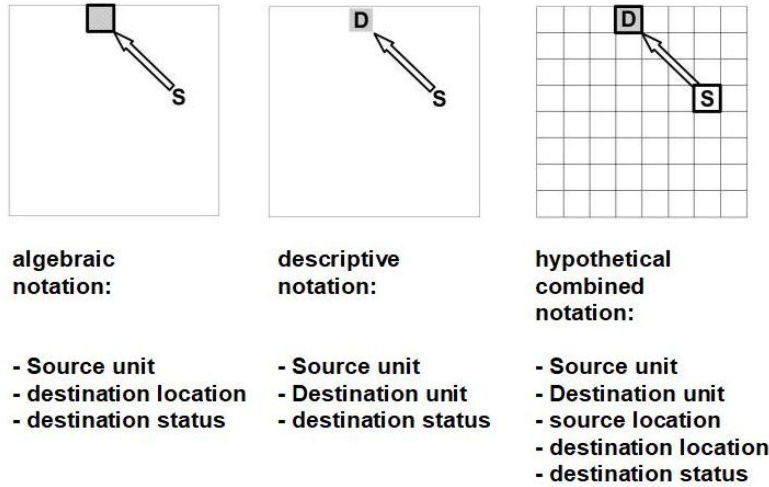


Figure 8: The mentioned hypothetical combined notation approaches a detailed diagram of an arbitrary object’s trajectory.

only list all four possibilities, namely e2-e4, e3-e4, ...e7-e5, ...e6-e5. Another, more verbose type of notation would be needed, such as long algebraic notation (e.g., Eade 1996, p. 331).

An example of both representations combined into a third, hypothetical representation: For a given player (White or Black), a combined notation that would carry all this information could be $S_1 U_1 @ X_1 A S_2 U_2 @ X_2$, where S_1 = side of board (K or Q) of the moving unit, U_1 = originating unit (P, N, B, R, Q, or K), X_1 = originating square, A = action (- or x, for "to" or "takes", respectively), S_2 = side of board of the destination square, U_2 = the unit on the destination square ("E" if the destination square is empty), X_2 = destination square, and the square coordinates can have any representation desired (such as from algebraic or descriptive notation). (See Figure 7.) For example, for the move e4 (or P-K4) of the Ruy Lopez opening, this generalized notation would be KP@e2-KE@e4, and for the move exd5 (or KPxQP) of the Center Counter Defense, this generalized notation would be KP@e4xQP@d5.

Answer, and how it was generated: The move e4 would be represented as $S_1 U_1 @ X_1 A S_2 U_2 @ X_2 = KP@e2-KE@e4$, and the originating square is the variable X_1 in this string, therefore via matching, $X_1 =$

e2, which is the answer.

2.2. Elements: list versus periodic table

- Question:** What is the name of the element that has 18 protons?

The best KRM to generate the answer: A list, since the list equivalent of the periodic table would be ordered by atomic number, which is the number of protons in an element, and a list can be easily stored on paper (as a written list) or on computer (as a data base or a narrow array), and simply traversed with that structure, either by eye or by computer.

The simple algorithm used: For a given, specified number of protons i , look up the name of the element at item $\#i$ in a list ordered by atomic number.

Answer, and how it was generated: The element at position $\#18$ in the list has the name argon, which is the answer. See Figure 9.
- Question:** Is element $\#49$ (indium) a metal?

The best KRM to generate the answer: The periodic table, since a clearly defined region occupying about the left $3/4$ of the periodic table is metals, and all other regions are nonmetals.

The simple algorithm used: If the column of the element in question is ≥ 3 and ≤ 12 , and if its row is ≥ 4 and ≤ 7 , then the element is a metal, else it is not a metal.

Answer, and how it was generated: Indium is located at column 13, row 5, and since column 13 is ≥ 12 , this location is outside of the metal region, indium is not a metal. See Figure 10.
- Question:** Which superheavy elements, not yet seen, are most likely to be stable?

The best KRM to generate the answer: Neither, since neither notation carries enough information to make such a prediction. Not only are several more pieces of information (attributes of the element) needed that are not carried in smaller periodic tables, but nuclides (alternative physical forms of each element) must also be listed, in contrast to the regular periodic table that shows only the most common

nuclide. The additional information needed includes: (1) the number of neutrons, not just the number of protons, (2) half-life information for measuring stability. Also, a spatial arrangement of the number of protons and neutrons is needed for efficiency so that estimates can be made about regions of high stability by using spatial direction and clustering information that results, and the range of proton-neutron values of such nuclides. Ultimately the best way to show the result is by a 3D plot called a "stability graph" of proton number, neutron number, and half-life, as in Figure 11.

An example of both representations combined into a third representation: See Figure 12. The same information can be carried by both the periodic table and stability graph, but there are deep visual alterations. For example, the periodic table's 2D regions, such as the region of metals that is shown in blue, disappears because a stability graph "straightens out" the 2D periodic table that was folded along the Z (= number of protons) axis into a straight Z axis, so any 2D region becomes scattered across the 1D Z axis. Also, the half-life information (t_h) that was textual in the periodic table becomes plotted as a point in a stability graph since that value is of the highest concern in the stability graph. Also, typical periodic tables do not include more than one nuclide per element, though the figure shown illustrates how a periodic table would logically include multiple nuclides per element, a scheme that causes the periodic table to become 3D with nuclides stored along the N (= number of neutrons) axis. In this case a generic representation of both KRMs at once would require a data base (or spreadsheet) + sort and subsort operation + list folding (see Figure 14) + plot ability. With such a foundation, an extended periodic table and stability graph could be converted to one another in either direction. Whether numerical data is represented as a data in a data base or a plot is only a preference in how it is viewed, not as an important distinction in KRM. (See Figure 13.)

Answer, and how it was generated: Visually it can be seen that the center of the cluster called the "island of stability" is at the coordinates $Z = 114$ protons and $N = 184$ neutrons, which corresponds to element 114, or nuclides of copernicium (= element 112) and flerovium (= element 114).

Element	Atomic Number	Atomic Mass
Hydrogen	1	1.0079
Helium	2	4.0026
Lithium	3	6.941
Beryllium	4	9.0122
Boron	5	10.811
Carbon	6	12.0107
Nitrogen	7	14.0067
Oxygen	8	15.9994
Fluorine	9	18.9984
Neon	10	20.1797
Sodium	11	22.9897
Magnesium	12	24.305
Aluminum	13	26.9815
Silicon	14	28.0855
Phosphorus	15	30.9738
Sulfur	16	32.065
Chlorine	17	35.453
Argon	18	39.0983
Potassium	19	39.948
Calcium	20	40.078

Figure 9: A list of elements. The elements are listed here in order of atomic number. This is basically a 1D data structure where each cell contains an embedded 1D list of attributes.

PERIODIC TABLE OF ELEMENTS

PubChem

1 H Hydrogen Nonmetal																	2 He Helium Noble Gas	
3 Li Lithium Alkali Metal	4 Be Beryllium Alkaline Earth Metal																	10 Ne Neon Noble Gas
11 Na Sodium Alkali Metal	12 Mg Magnesium Alkaline Earth Metal																	18 Ar Argon Noble Gas
19 K Potassium Alkali Metal	20 Ca Calcium Alkaline Earth Metal	21 Sc Scandium Transition Metal	22 Ti Titanium Transition Metal	23 V Vanadium Transition Metal	24 Cr Chromium Transition Metal	25 Mn Manganese Transition Metal	26 Fe Iron Transition Metal	27 Co Cobalt Transition Metal	28 Ni Nickel Transition Metal	29 Cu Copper Transition Metal	30 Zn Zinc Transition Metal	31 Ga Gallium Post-Transition Metal	32 Ge Germanium Metalloid	33 As Arsenic Metalloid	34 Se Selenium Nonmetal	35 Br Bromine Halogen	36 Kr Krypton Noble Gas	
37 Rb Rubidium Alkali Metal	38 Sr Strontium Alkaline Earth Metal	39 Y Yttrium Transition Metal	40 Zr Zirconium Transition Metal	41 Nb Niobium Transition Metal	42 Mo Molybdenum Transition Metal	43 Tc Technetium Transition Metal	44 Ru Ruthenium Transition Metal	45 Rh Rhodium Transition Metal	46 Pd Palladium Transition Metal	47 Ag Silver Transition Metal	48 Cd Cadmium Transition Metal	49 In Indium Post-Transition Metal	50 Sn Tin Post-Transition Metal	51 Sb Antimony Metalloid	52 Te Tellurium Metalloid	53 I Iodine Halogen	54 Xe Xenon Noble Gas	
55 Cs Cesium Alkali Metal	56 Ba Barium Alkaline Earth Metal		72 Hf Hafnium Transition Metal	73 Ta Tantalum Transition Metal	74 W Tungsten Transition Metal	75 Re Rhenium Transition Metal	76 Os Osmium Transition Metal	77 Ir Iridium Transition Metal	78 Pt Platinum Transition Metal	79 Au Gold Transition Metal	80 Hg Mercury Transition Metal	81 Tl Thallium Post-Transition Metal	82 Pb Lead Post-Transition Metal	83 Bi Bismuth Metalloid	84 Po Polonium Metalloid	85 At Astatine Halogen	86 Rn Radon Noble Gas	
87 Fr Francium Alkali Metal	88 Ra Radium Alkaline Earth Metal		104 Rf Rutherfordium Transition Metal	105 Db Dubnium Transition Metal	106 Sg Seaborgium Transition Metal	107 Bh Bohrium Transition Metal	108 Hs Hassium Transition Metal	109 Mt Meitnerium Transition Metal	110 Ds Darmstadtium Transition Metal	111 Rg Roentgenium Transition Metal	112 Cn Copernicium Transition Metal	113 Nh Nihonium Post-Transition Metal	114 Fl Flerovium Post-Transition Metal	115 Mc Moscovium Post-Transition Metal	116 Lv Livermorium Post-Transition Metal	117 Ts Tennessine Halogen	118 Og Oganesson Noble Gas	
		57 La Lanthanum Lanthanide	58 Ce Cerium Lanthanide	59 Pr Praseodymium Lanthanide	60 Nd Neodymium Lanthanide	61 Pm Promethium Lanthanide	62 Sm Samarium Lanthanide	63 Eu Europium Lanthanide	64 Gd Gadolinium Lanthanide	65 Tb Terbium Lanthanide	66 Dy Dysprosium Lanthanide	67 Ho Holmium Lanthanide	68 Er Erbium Lanthanide	69 Tm Thulium Lanthanide	70 Yb Ytterbium Lanthanide	71 Lu Lutetium Lanthanide		
		89 Ac Actinium Actinide	90 Th Thorium Actinide	91 Pa Protactinium Actinide	92 U Uranium Actinide	93 Np Neptunium Actinide	94 Pu Plutonium Actinide	95 Am Americium Actinide	96 Cm Curium Actinide	97 Bk Berkelium Actinide	98 Cf Californium Actinide	99 Es Einsteinium Actinide	100 Fm Fermium Actinide	101 Md Mendelevium Actinide	102 No Nobelium Actinide	103 Lr Lawrencium Actinide		

Figure 10: A periodic table. Note that the metals form a clear-cut, roughly rectangular region (colored in blue here, which are: Sc through Zn, Y through Cd, Hf through Hg, and Rf through Cn) when elements are organized into this periodic table. This table is roughly a 2D data structure where each cell contains an embedded 1D list of attributes. (Source: Public Domain, PubChem via NIH.)

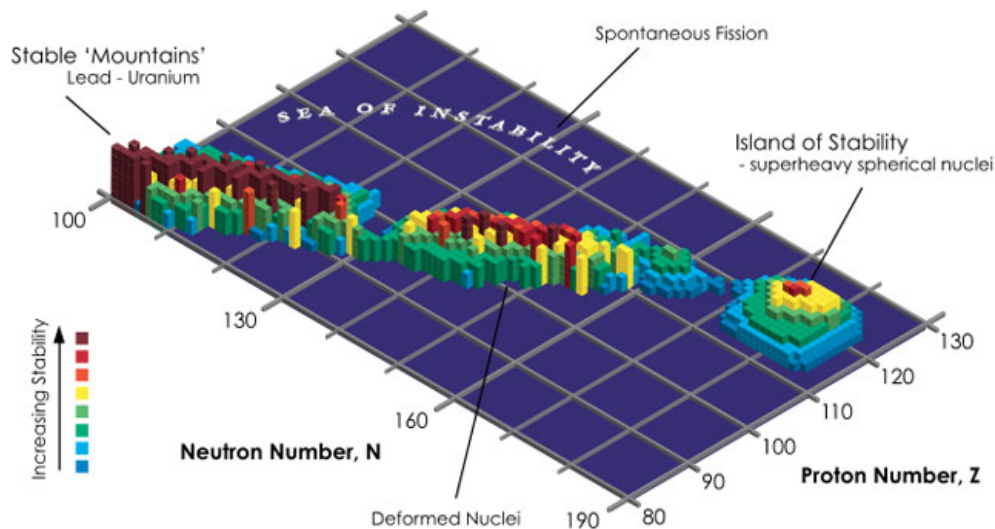


Figure 11: A stability graph. Knowledgeable prediction of stable heavy elements requires spatial assessment of nuclides of the basic elements. This is a 3D data structure. (Source: unknown.)

2.3. Series notation: infinite series versus infinite product

1. **Question:** Is it easier to take the derivative of an infinite series, as in Figure 15, or an infinite product, as in Figure 17?

The best KRM to generate the answer: Infinite series.

The simple algorithm used: Take the derivative of each term. At the end, add all the terms. If this were an infinite product expression, it would instead be necessary to use the product rule $(fg)' = f'g + fg'$, which requires taking two derivatives, two multiplications, and one addition per factor, as well as adding all the terms at the end. See Figure 16.

Answer, and how it was generated: The infinite series in Figure 15. See the steps of differentiation in the lower levels that figure.

2. **Question:** Is it easier to find the roots of an infinite series, as in Figure 18, or an infinite product, as in Figure 17?

The best KRM to generate the answer: Infinite products.

The simple algorithm used: Set each factor in the product to 0,

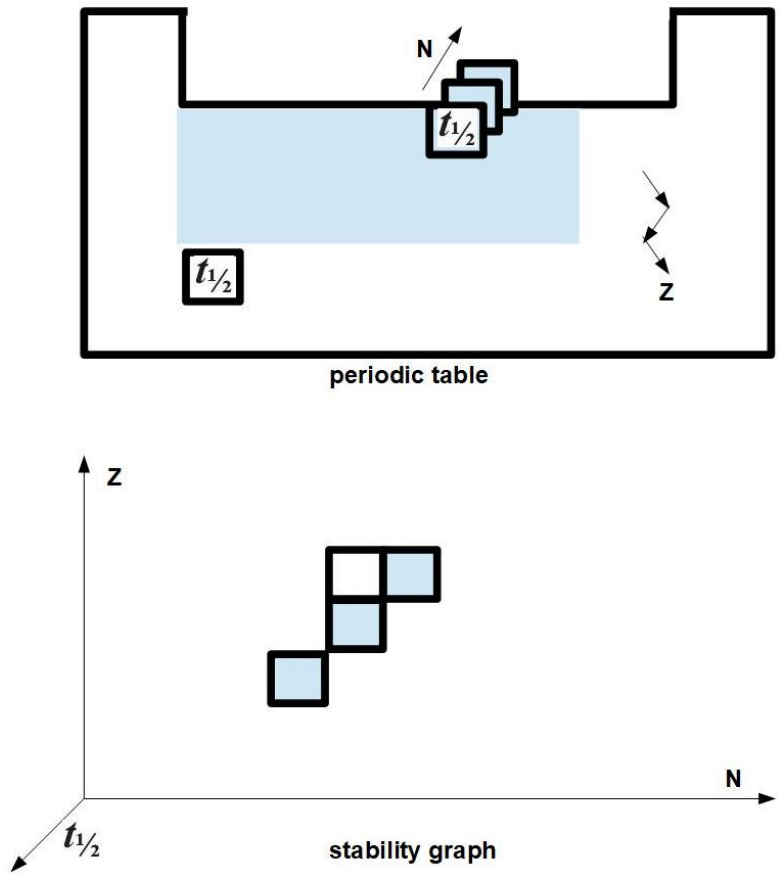


Figure 12: The stability graph has the same information as a periodic table that includes nuclides and half-life ($t_{\frac{1}{2}}$) information, but the information has been spatially rearranged.

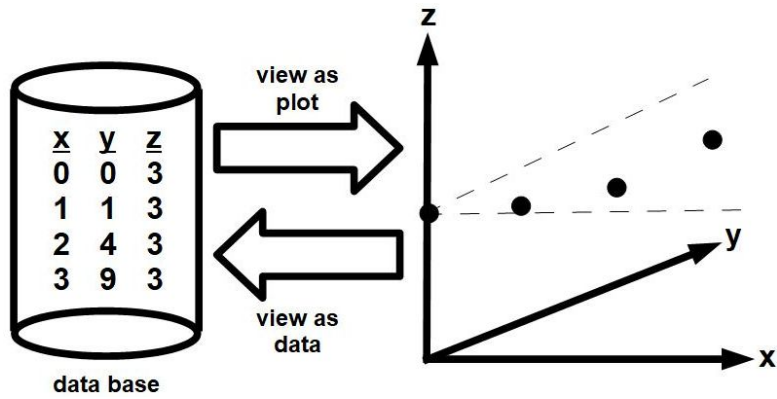


Figure 13: For numerical data, plots carry the same information as a data base, and vary only in the way the data is viewed.

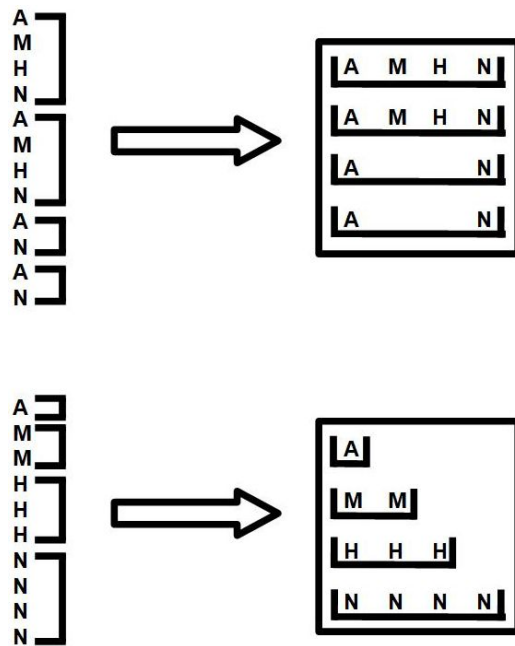


Figure 14: A list in which attributes form a pattern can be easily folded into a 2D table, but the resulting table may not be purely rectangular. The reverse operation of unfolding such a table into a list is even easier.

and solve each such equation for x .

Answer, and how it was generated: The infinite series in Figure 17. Set $1-(x^2) = 0$; $1-(x^2/4) = 0$; $1-(x^2/9) = 0$, etc., or equivalently: set $x^2 = 1$, which has solution $x = 1$; set $x^2/4 = 1$, which has solution $x = 2$; set $x^2/9 = 1$, which has solution $x = 3$, etc.

3. **Question:** Which representation is likely to be the easiest for proving that a number is irrational: infinite series or infinite products?

The best KRM to generate the answer: Neither, considering the history of the first proofs of irrationality for the numbers π and e . Both of those first proofs were based on continued fractions, which are neither infinite series nor infinite products.

An example of both representations combined into a third representation: Infinite sums and infinite products can be artificially generalized by introducing an auxiliary function "a" that can be set to a binary operator such as addition, multiplication, or exponentiation: $a(x, y) = (x + y)$ or $(x * y)$ or (x^y) . An infinite sequence of simple expressions to be combined can then be described by using recursion in conjunction with this function "a" as follows: $P_n = a(P_{n-1}, t_n)$, where P_n is the current partially accumulated value to be computed, P_{n-1} is the previous partially accumulated value computed, and t_n is the current term to be included. For example, the infinite series $1 + (1/2) + (1/4) + (1/8) + \dots = 1/2^0 + (1/2^1) + (1/2^2) + (1/2^3) + \dots$ could be represented with this new notation as $P_n = a(P_{n-1}, (1/2)t_{n-1})$, where $a(x, y) = (x + y)$ to designate summation, $t_0 = 1$, $P_0 = 1$, $t_i = (1/2)t_{i-1} = (1/2)1 = 1/2$. Therefore $P_1 = a(P_0, (1/2)t_0) = 1 + (1/2)1 = 1 + 1/2$, and $t_1 = (1/2)t_0 = (1/2)1 = 1/2$; $P_2 = a(P_1, (1/2)t_1) = (1 + 1/2) + (1/2)(1/2) = 1 + 1/2 + 1/4$, and $t_2 = (1/2)t_1 = (1/2)(1/2) = 1/4$; $P_3 = a(P_2, (1/2)t_2) = (1 + 1/2 + 1/4) + (1/2)(1/4) = 1 + 1/2 + 1/4 + 1/8$, and $t_3 = (1/2)t_2 = (1/2)(1/4) = 1/8$; etc. Some simpler continued fractions can also be represented with this recursive general notation if $a(x, y)$ is set to $k + 1/a(x, y)$, but more complicated continued fractions may need an additional auxiliary function. In general, a pair of coupled recursive formulas are needed, one for the numerator and one for the denominator (Khinchin 1964, p. 4). For a simple example, $\sqrt{2} = 1 + (1/(2+1/(2+1/(2+1/...))))$ in continued fraction form, which recursively would use $k = 2$, or $a(x, y) = 2 + 1/a(x, y)$.

$$\begin{aligned}
e^x &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \text{etc.} \\
(e^x)' &= 1' + (x)' + \left(\frac{x^2}{2!}\right)' + \left(\frac{x^3}{3!}\right)' + (\text{etc.})' \\
(e^x)' &= 0 + 1 + 2\frac{x}{1.2} + 3\frac{x^2}{1.2.3} + 4\frac{x^3}{1.2.3.4} + (\text{etc.})' \\
(e^x)' &= 0 + 1 + \cancel{2}\frac{x}{\cancel{1.2}} + \cancel{3}\frac{x^2}{\cancel{1.2.3}} + \cancel{4}\frac{x^3}{\cancel{1.2.3.4}} + (\text{etc.})' \\
(e^x)' &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \text{etc.} \\
\boxed{(e^x)' = e^x}
\end{aligned}$$

Figure 15: An infinite series (= sum) can be differentiated easily, per term.

Answer, and how it was generated: Partial fraction representation was historically the basis of the first proof that π is irrational, proven by Lambert in 1767, and by Legendre in 1794 (Beckmann 1971, pp. 170-171) (Figure 19), as well as the first proof that e is irrational, proven by Euler in 1737 (Maor 1994, p. 192) (Figure 20). Since these two numbers are the most famous and most commonly occurring natural mathematical constants, history suggests that partial fractions are the best representation for proving that a given natural mathematical constant is irrational.

3. Generalizations from the specific KRM examples

Further examples beyond the three specific KRM examples would be prohibitively long for this document, but in general the following observations seem to hold for all KRMs:

1. The best KRM to use for a given domain depends on which information needs to be made more explicit, though in making that information more explicit, some other information is usually made less explicit to

$$\frac{d}{dx} \left[\prod_{i=1}^k f_i(x) \right] = \left(\prod_{i=1}^k f_i(x) \right) \left(\sum_{i=1}^k \frac{f_i'(x)}{f_i(x)} \right)$$

Figure 16: An infinite product cannot be differentiated easily, since the Product Rule requires an indefinite number (k) of products be multiplied by an indefinite number (k) of sums as k approaches infinity.

$$0 = \pi x \left(1 - x^2\right) \left(1 - \frac{x^2}{4}\right) \left(1 - \frac{x^2}{9}\right) \dots \left(1 - \frac{x^2}{N^2}\right) \dots$$

Figure 17: An infinite product with an infinite number of roots, all easy to find.

$$0 = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

Figure 18: An infinite series with an infinite number of roots, which are difficult to find, in general.

$$\tan(x) = \frac{x}{1 - \frac{x^2}{3 - \frac{x^2}{5 - \frac{x^2}{7 - \dots}}}}$$

$$\tan\left(\frac{\pi}{4}\right) = 1$$

Figure 19: This continued fraction representation for tan(x) and the accompanying identity were used in the first proof that π is an irrational number.

$$e = 2 + \frac{1}{1 + \frac{1}{2 + \frac{1}{1 + \frac{1}{1 + \frac{1}{4 + \frac{1}{1 + \frac{1}{1 + \frac{1}{6 + \dots}}}}}}}}}$$

Figure 20: This continued fraction representation for e was used in the first proof that e is an irrational number.

compensate for the increase in representation size. This is already a very well-known heuristic in computer science.

2. It is possible to retain both types of information in the KRM as one type needs to be made more explicit, but the cost will be the increase in representation size.
3. No matter which new KRM is used for each increase in types of information retained, there will always be some application that needs yet another, different KRM, until a KRM with maximum complexity is reached for that domain.
4. As the representation size becomes large, the KRM approaches a maximum complexity in a way that suggests some very general form of KRM, such as a generic data base without specifying any specific tables, records, or fields, or a generic neural network without specifying any specific number of layers, number of neurons, or connection pattern.
5. Systems with shaped objects in motion reach maximum complexity in the representation form of simulation type systems (i.e., SCOVA representation). Examples of such systems include board games, puzzles, folding molecules, and the piano mover's problem.
6. Systems without shaped objects, motion, aggregation, or rules for directly combining objects reach maximum complexity in the representation form of data base type systems (i.e., OAV representation). Ex-

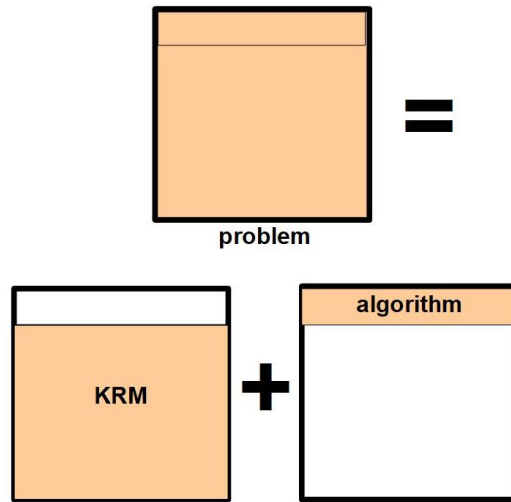


Figure 21: A difficult problem can often be solved by decomposing it first into an appropriate KRM, which then leads to a simple algorithm. The bulk of the effort in solving the problem therefore goes into development of the KRM.

amples of such systems include data bases, data sets, and the periodic table of the elements.

7. Systems with aggregation but without shaped objects or motion, but with rules or heuristics for directly combining objects reach maximum complexity in the representation form of formulas or grammars (i.e., OAV representation), where a large number of relatively simple components of limited types combine according to simple rules or heuristics to form very impressive structures or descriptions. Examples of such systems include mathematics, music, language, chemistry, and genetics.

4. Problem decomposition into KRM + algorithms

The aforementioned generalizations about KRMs can be viewed as a method of decomposing a difficult problem into two parts: the KRM and the algorithm that works on pieces that use that KRM. See Figure 21.

Recent evidence from modern large language models (LLMs) suggests that statistical learning works impressively well for giving the illusion of intelligence operating behind chatbots based on LLMs. Although such systems have no understanding of what any of the words mean, and cannot do

binding or explicit spatial reasoning, an important lesson learned from their performance and popularity is that massive statistical training can work extremely well in providing smooth, real-time, plausible, default assumptions, and can occasionally even give rise to spontaneously emergent behavior such as arithmetic abilities (Rorvig 2021, p. 39).

The author believes that the aforementioned behavior from LLMs suggests that CSR will decompose into six instead of two main components, where the first pair of components deals with knowledge matching, and the last pair of components deals with learning. The training time for learned CSR knowledge is necessarily very lengthy, essentially of the same order of magnitude as for human CSR learning, and although this is a complication, time is not shown in the algorithm decomposition diagram of Figure 22. If this proposed decomposition strategy holds true for CSR then the implication is that Phase 1 alone of this project will be highly incomplete because it address only two components—KRM and algorithm—not lengthy learning time. This is where the fusion of explicit and implicit memory merge: this current study uses explicit memory in the form of connected groups of icons, but true general intelligence will need to have both forms of memory, including statistical fusion of huge numbers of Tumbug icon structures in order to immediately default to proper associations and temporal continuations. This current direction of study must start somewhere, however, so the issues of training on Tumbug icon structures are deferred for future work. Hopefully the follow-up work will be greatly simplified due to the ability of the Tumbug KRM to represent knowledge in a useful and profound way, which echoes Haugeland’s statement: ”It may even happen that, once the fundamental structures are worked out, acquisition and adaptation will be comparatively easy to include” (Haugeland 1985, p. 11). On the positive side, conventional neural network training *per se* is mostly straightforward, although generating quality training data is a challenge. Also, a certain amount of Tumbug programming can substitute for naive training, especially with Correlation Boxes, whose function mappings can resemble the mappings of neural networks.

As most highly experienced people will confirm, especially teachers of enormous topics such as a given foreign language or chess, there is no substitute for experience. General intelligence, just like large academic topics, requires an very large number of training experiences over large spans of time in order to generalize and predict properly. Some of this burden can probably be eased by explicit programming of some heuristics, but the amount, qual-

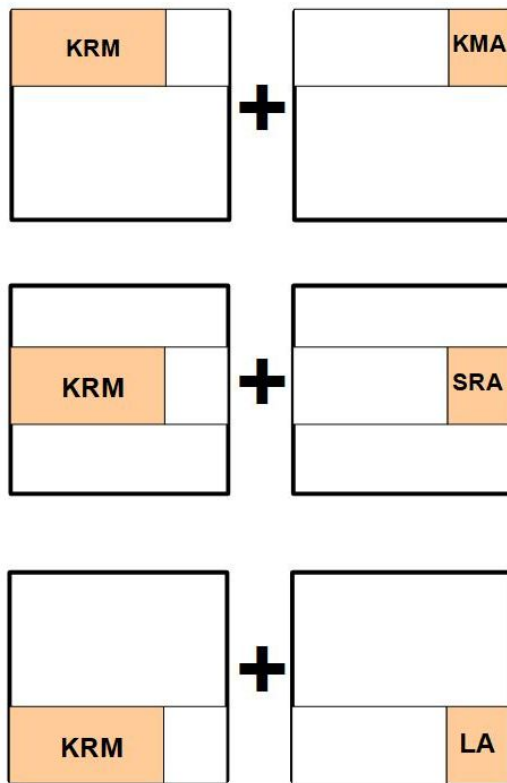
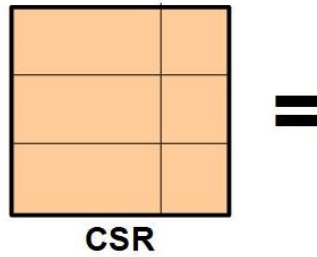


Figure 22: The author's prediction: CSR will require three algorithms in its decomposition, and all will be based on a KRM like Tumbug. The bulk of the effort in solving the problem therefore goes into development of the KRM.

KRM = knowledge representation method (viz. Tumbug)

KMA = knowledge matching algorithm (probably a special case of the SRA)

SRA = spatial reasoning algorithm

LA = learning algorithm

ity, and detail of knowledge learned by the human brain is incredibly vast, too vast to encode, even when summarized with explicitly coded heuristics. This implies that there is no shortcut to overcome the need for vast amounts of training, which usually involves vast amounts of time.

4.1.3 Clues to the type of KRM the brain uses

Not many AI researchers have directly conjectured what type of KRM the human brain might use, but a few researchers have provided clues:

1. Per Roger Schank, predicate calculus is insufficient (Schank 1976, p. 172). Prior to Schank's CD theory, predicate calculus was the only suggested canonical form of propositional information (Schank 1976, p. 172), which is one indication of the paucity of research into KRMs for AGI, a situation that still persists today (as of 2023). Implication: The brain's KRM is more powerful in representational power than predicate logic.
2. Roger Schank reasons that a canonical language representational scheme could not have natural-language words as its elements since natural language is too ambiguous (Schank 1976, pp. 171-172). In fact, this limitation was Schank's motivation for creating CD theory. Implication: The brain's KRM is more powerful in representational power than the tree data structure, which is used in sentence diagramming.
3. William A. Woods wrote, "Viewing the knowledge base of an intelligent agent as a model of the external world focuses attention on a number of problems that are not normally addressed in database systems or knowledge-based expert systems." (Woods 1986, p. 1323) Implication: The brain's KRM is more powerful in representational power than data bases or rule-based expert systems, which means more powerful than tables or trees, even when outfitted with search and matching routines, even when outfitted with an uncertainty management system such as confidence factors, fuzzy logic, or Dempster-Shafer theory (e.g., Coppin 2004, p. 483).
4. Ben Coppin wrote, "The human mind uses some form of representation for all concepts, which enables us to understand such abstract ideas as "happiness," "lateness," and "common sense." . . . Clearly

this internal representation has a lot to do with our ability to think, to understand, and to reason, and it is no surprise, therefore, that much of Artificial Intelligence research is concerned with finding suitable representations for problems” (Coppin 2004, p. 466). Implication: The brain’s KRM needs to represent abstract concepts as well as concrete concepts.

In summary, it is clear the human brain’s KRM is much more complex and expressive than any of the commonly used KRMs of AI. At the least, the human brain’s KRM covers any form of symbols. Jerry A. Fodor wrote, ”I suspect that the representational system with which we think, if that’s the right way to describe it, is so rich that if you think up any form of symbolism at all, it probably plays some role in thinking” (Fischler and Firschein 1987, p. 308). Similarly, John Haugeland wrote, ”AI has discovered that knowledge itself is extraordinarily complex and difficult to implement—so much so that even the general structure of a system with common sense is not yet clear” (Haugeland 1985, p. 11).

Side conjecture: The KRM used by the human brain may be at or above the pictorial level of representation in a list of KRMs ordered by their representational power that has the most powerful KRMs at the top.

One possibility is that the human brain’s KRM involves images, or some something like images (= ”imageoids”), which a possibility that has been suggested by several authors over the years, and even has an entire book devoted to the topic, viz. Kosslyn et al. 2006:

1. Martin Fischler and Oscar Firschein believe the mind uses two, major representations, namely propositions and images (Fischler and Firschein 1987, p. 308), though they admit that, when considering images, some representations are isomorphic to images but would not be considered images *per se*. Fischler and Firschein mention some important advantages and disadvantages of using images as a KRM.
2. John Haugeland notes, ”The beauty of images is that (spatial) side effects take care of themselves. If I imagine myself astride a giraffe, my feet ’automatically’ end up near its belly and my head near the middle of its neck. If I have a scale model of my living room, and I put the model couch in the front alcove, it ’automatically’ ends up by the bay window and opposite the door. I don’t have to arrange for these results

deliberately or figure them out; they just happen, due to the shapes of the images themselves” (Haugeland 1985, p. 229).

3. James Hogan wrote, ”Vision and language involve what are perhaps our most complex and abstract representations of the external and internal realities we experience. Comprehending how they operate would get close to the core of what true intelligence is all about” (Hogan 1997, p. 179).
4. Kenneth Haase came up with an idea of visual memory that influenced Marvin Minsky’s idea of K-lines (Minsky 1986, p. 82), which is a theory of memory. ”You want to repair a bicycle. Before you start, smear your hand with red paint. Then every tool you need to use will end up with red marks on it. When you’re done, just remember that red means ’good for fixing bicycles.’ Next time you fix a bicycle, you can save time by taking out all the red-marked tools in advance.” He then generalized this idea to different colors for different jobs.
5. Rodney Brooks wrote: ”There is certainly no AI vision program which can find arbitrary chairs in arbitrary images; they can at best find one particular type of chair in carefully selected images. This characterization, however, is perhaps the correct AI representation of solving certain problems; e.g., a person sitting on a chair in a room is hungry and can see a banana hanging from the ceiling just out of reach. Such problems are never posed to AI systems by showing them a photo of the scene. A person (even a young child) can make the right interpretation of the photo and suggest a plan of action. For AI planning systems, however, the experimenter is required to abstract away most of the details to form a simple description in terms of atomic concepts such as PERSON, CHAIR and BANANAS.” (Brooks 1992, p. 143)
6. Donald R. Tvetter wrote: ”The fact that people do store many memories as pictures and do at least some of their reasoning about the world using pictures ought to be one of the most obvious principles of all, and yet it has been neglected, in part due to the predominance of symbol processing and in part due to the fact that processing pictures is still underdeveloped and has not been used in conjunction with representing the real world in programs where the goal of the program is to reason about the real world.” (Tvetter 1998, p. 18)

The fact that imageoids are such an obvious candidate for the brain's KRM and yet are still not being researched to any great extent is puzzling. If a breakthrough in AGI is going to be made, this is likely the subject area in which it will occur. This is why the Tumbug KRM has been based on imageoids.

4.1.4 Ranking of KRM methods

Can the "best" KRM be determined by its properties alone? Probably, but this has not been done yet. Only a few authors have mentioned ways to measure the relative power of KRMs. One is Arthur B. Markman, who suggested three attributes (Markman 1999, p. 14):

1. the duration of the representational states
2. the presence of discrete symbols
3. the abstractness of representations

The following previously unpublished, additional criteria would likely be useful, as well:

1. analog representation capability (e.g., for representing time/space with indefinite precision)
2. data structure intricacy or nonlinearity (e.g., tree structures are more intricate than lists)
3. ability to represent time (e.g., for representing moving images, or other changes over time)

Another such author is Han Reichgelt (Reichgelt 1991, p. 5), who divided KRMs into two practical considerations:

1. syntactic aspects - "the naturalness and the expressiveness" of the KRM
2. inferential aspects - "the power of the underlying inference machinery"

Despite the lack of consensus on this topic, the following hierarchy attempts to show a few well-known KRMs, and the author's estimate of their approximate representation power relative to each other, where #1 is the most powerful KRM listed here.

1. pictures or illustrations
2. directed graphs (e.g., semantic nets, knowledge graphs, state diagrams, DFAs, NFAs)
3. directed acyclic graphs
4. trees
5. binary trees
6. jagged arrays
7. arrays or adjacency graphs
8. Venn diagrams
9. lists
10. vectors
11. scalars

A few other KRMs such as predicate calculus, pushdown automata, CD theory, and Petri nets are not on this list since their advantages and disadvantages involve attributes that are distributed across multiple dimensions and are distributed excessively across even the levels of a 1D list.

The "pictures" type of KRM mentioned at #1 occurs in Tumbug only inside of a Location Box of some type, which is a Tumbug Building Block icon for representing space. Such pictures may be represented verbatim, in which case a Verbatim Box is used, or they can be abstracted into general regions and general objects without being rigidly specified, in which case a Descriptive Box is used. Together these types of Location Boxes allow concrete images and abstracted images to be represented, or both types mixed together. The fact that Tumbug uses the most powerful KRM, viz. images, suggests that Tumbug could be the most powerful KRM developed so far, provided that its Building Blocks prove to be strong and consistent.

Side conjecture: In theory, it is likely that ultimately no canonical form of meaning exists, any more than a canonical description of 2D shapes exists, since any given description can be based on different conceptual foundations that can be combined in uniquely different ways that produces unmatched

forms. However, in practice there usually does exist a description that is more intuitive or natural than others, and usually also simpler than others, which in practice serves as the canonical form.

4.1.5 Why dependency on learning is misguided

A common fallacy is that all that is needed to produce AGI is to let a system learn. Haugeland (1985, p. 11) described this misconception well:

”Learning is the acquisition of knowledge, skills, etc. The issue is typically conceived as: given a system capable of knowing, how can we make it capable of acquiring? Or: starting from a static knower, how can we make an adaptable or educable knower? This tacitly assumes that knowing as such is straightforward and that acquiring or adapting it is the hard part; but that turns out to be false. AI has discovered that knowledge itself is extraordinarily complex and difficult to implement—so much so that even the general structure of a system with common sense is not yet clear. Accordingly, it’s far from apparent what a learning system needs to acquire; hence the project of acquiring some can’t get off the ground.”

Figure 23 shows basic conceptual components of processors (both for IT and AGI) and their dependencies. Note that learning is dependent upon data structures or knowledge structures, or in general learning is dependent of *structured* data/information/knowledge/wisdom (DIKW), which in turn are based on KRMs. Therefore there exists no easy way to avoid KRMs because they are basic to every processor of which humans can perceive.

4.1.6 Criteria and heuristics that make Tumbug seem particularly promising

1. The-only-move that-doesn’t-lose Principle

Every current AI design appears to have at least one fatal flaw that prevents it from every achieving true general intelligence status. An overview is shown in Figure 24. A question mark in the table means ”unknown.” Merely by virtue of being new, Tumbug has a particularly promising design because most of its table cells are still blank because not enough research has been done on Tumbug to identify any fatal flaws. Although many blank cells seems to be weak justification of a promising design, Tumbug was specifically designed to overcome the flaws of every major AI paradigm. Also, by analogy,

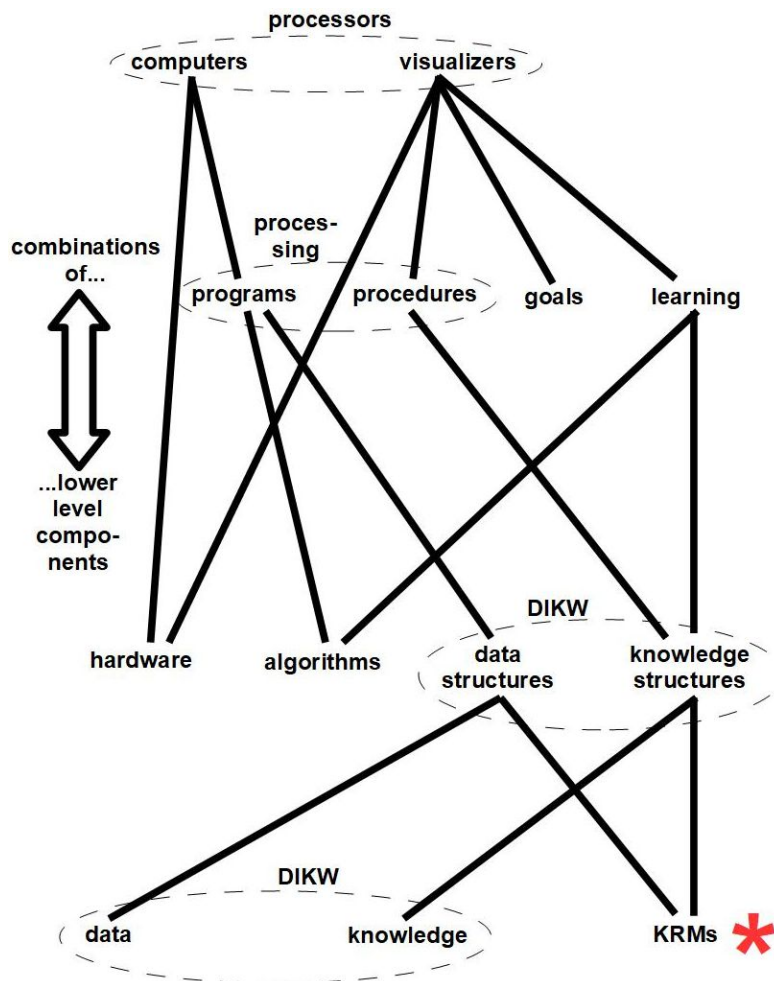


Figure 23: Conceptual dependencies in processors. Note that KRMs underlie both types of processors.

ART = Adaptive Resonance Theory
 NN = neural network

	prosaic homo- morphism	1-pass learning	spatial reasoning	general- ization learning ability
rule-based expert systems	+	+	-	-
Shastri reflexive reasoning	+	+	-	-
backpropagation NNs	+	-	-	+
ART NNs	+	+	-	+
continuous Hopfield NNs	-	-	-	-
cellular automata	-	-	-	-
Large Language Models	+	-	-	+
Tumbug	+	+	?	?

Figure 24: Some of the most promising current approaches to AGI.

a row of the table is like an evaluation of physical paths through a rugged landscape, where the traveler knows in advance that at least one navigable path exists: many paths can be tentatively ruled out because they appear to have impasses, whereas a path where nothing is known has a greater likelihood of being one of the paths that are navigable. This type of ignorance evaluation logic also appears in chess, where the proper move can sometimes be chosen only because all the well-known alternatives are known to lose, therefore the best candidate for the best move is a move of which little is known.

2. Organization is more important than performance

Some of the deepest insights into the nature of intelligence and its relation to AI come from well-known authors who emphasize that what is not important for producing AGI is not speed (therefore faster computers, parallel computers, and quantum computers will not be the key to AGI) or superficially impressive results, but rather the organization of the system. AGI will clearly need a clever type of underlying organization that has not yet been publicly considered. Below are some quotes to this effect.

Marvin Minsky: "Hardware is not the limiting factor for building an intelligent computer. We don't need supercomputers to do this; the problem

is that we don't know what's the software to use with them. A 1 MHz computer probably is faster than the brain and would do the job provided that it has the right software." (Sabbatini 1998)

Jeff Hawkins: "According to functionalism, being intelligent or having a mind is purely a property of organization and has nothing inherently to do with what you're organized out of. A mind exists in any system whose constituent parts have the right causal relationship with each other, but those parts can just as validly be neurons, silicon chips, or anything else." (Hawkins 2004, p. 36)

3. Pei Wang's tables

Professor Pei Wang, in his course on AGI at Temple University, Philadelphia, collected and summarized the approaches to AGI that he considers the most promising, which are listed in Figure 28. His list of criteria is shown in Figure 27. The architectures are allowed to come from any of three types of integration, which are listed in Figure 25. The "integrated" type is the same concept that the author called "fusion" type in 2000 (Atkins 2000), which is the type that the author favors. The author's Visualizer Project that includes Tumbug fits two out of three of Wang's criteria for a promising AGI architecture:

1. The Visualizer Project is not application-specific, therefore Tumbug fits criteria #1.
2. The Visualizer Project is still currently being worked on, therefore Tumbug fits criteria #2.
3. The Visualizer Project is too new to be documented well, therefore Tumbug does not fit criteria #3.

Wang's last two criteria are based on the current worldly success of a given architecture, so if an architecture fails to meet either of these criteria it does not necessarily mean that the architecture is deficient *per se*, only that it is not popular for some reason.

Regarding Wang's three categories of combined architectures (viz., hybrid, integrated, and unified), see Figure 26. Some examples of two different architectures being combined in these three ways in a single system are:

- rule-based expert system + neural network (NN):

Strategies of building AGI

- Hybrid: linking multiple existing AI techniques into one system
- Integrated: creating an architecture with multiple functional modules
- Unified: extending a single technique to cover various functions

Figure 25: Wang's three types of combined systems are hybrid, integrated, and unified.

1. hybrid: each type of architecture can be called from a program running in the system, as needed
 2. integrated: a heuristic search can use the expert system for the search, and the NN for the heuristics
 3. unified: the rule-based expert system is implemented with artificial neurons
- digital computer + analog computer:
 1. hybrid: each type of computer can be called from a program running in the system, as needed
 2. integrated: the digital computer performs only algebraic operations, and the analog computer performs only calculus operations
 3. unified: each calculation is done in analog, but is iterated digitally to confirm accuracy
 - object-oriented program + data base program:
 1. hybrid: each type of program can be called from a program running in the system, as needed
 2. integrated: the data base holds the initial values of each object, and the object-oriented program performs simulations with these objects
 3. unified: an object-oriented data base

4. Kocijan's criteria

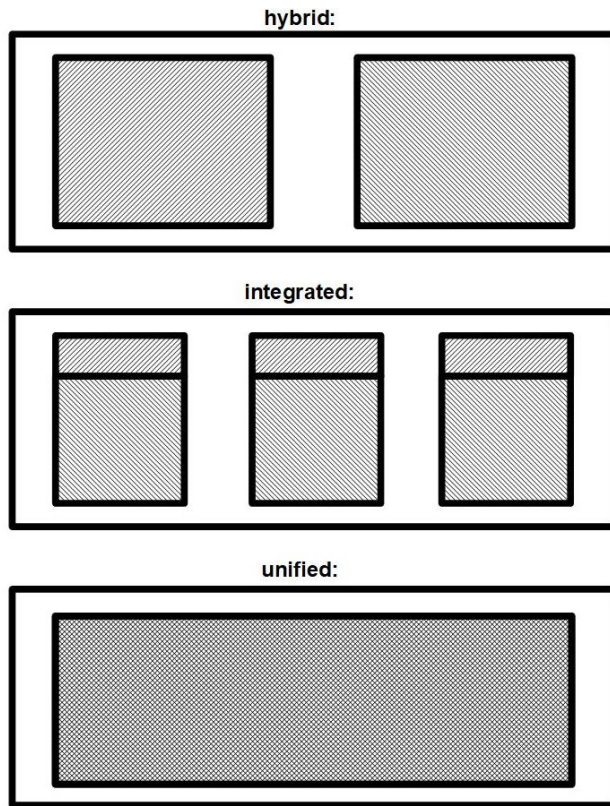


Figure 26: A diagrammatic view of Wang’s list of three strategies for combining existing disparate architectures. Per Wang, the ideal strategy for AGI is unified, which happens to be the type that Tumbug is.

Representative AGI Projects

1. It is clearly oriented to AGI.
Not fitting this criterion are: IBM's Watson, Deep Mind's AlphaGo
2. It is still very active.
Not fitting this criterion are: Pollock's OSCAR, Brooks' Cog
3. It has ample publications on technical details.
Not fitting this criterion are: many recent AGI projects except Gato and a LeCun paper

Figure 27: Wang’s table of criteria of promising AGI approaches.

Representative AGI Projects

<u>goal/path</u>	<u>hybrid</u>	<u>integrated</u>	<u>unified</u>
<u>principle</u>	-	-	AERA, AIXI, NARS
<u>function</u>	-	Open Cog, Sigma, Soar	SNePS
<u>capability</u>	-	-	Cyc, Gato
<u>behavior</u>	-	ACT-R, LIDA, MicroPsi	-
<u>structure</u>	-	-	HTM

Figure 28: Wang’s table of some of the most promising AGI approaches.

Kocijan et al. (2022) assessed different approaches used so far to solve WSC problems, and mentioned the following five desiderata. Each desideratum below is listed with a comment about how Tumbug (or more generally, a visualizer) is believed to obviate the mentioned pitfall. Page numbers in this list refer to the aforementioned reference.

1. The inference should be carried out automatically, before the question points out the anomaly (p. 4).
A visualizer conforms to this criterion, but demonstration of this will require a knowledge matching algorithm, which will not be fully investigated and documented until the end of Phase 2. Tumbug is quite simple, so an architecture such as a visualizer using Tumbug does not require "commonsense reasoning of some depth and complexity" (p. 4). A visualizer is a novel implementation of a "reflexive reasoning" architecture (cf. Shastri and Ajjanagadde 1990).
2. The approach should not be easily solvable using word correlation (p. 16).
As used so far, the Tumbug approach completely ignores the statistics of word frequency or word correlation, so obviously Tumbug bypasses this pitfall.
3. The approach should demonstrate some semblance of CSR, not merely technical tricks (pp. 7-8).
Because Tumbug uses a visual grammar that the author believes is universal across all natural languages, and because Tumbug deals with the

actions of real-world objects, at the very least each scenario that Tumbug visualizes could be run as a video simulation, which would confirm that the correct corporeal actions on the correct corporeal objects are being used.

4. The system should have reasoning capability, not just general pattern mapping as in neural networks (p. 23).
Tumbug is mostly symbolic/iconic, not a neural network, so it avoids naive pattern mapping of the neural network type.
5. The system should be able to generalize, so that good performance is attained across any subset of WC, not just one selected subset (p. 26).
If this statement refers to learning, then a visualizer should conform to this criterion, but research on this will not begin until Phase 4. If this statement refers to programmable representation of a concept, that a Tumbug diagram can already do this.

4.2 Applicable systems from the real world

4.2.1 Physics

Obviously physics is intended to provide explanations for events that happen in our physical universe, so a KRM that is trying to represent those same events should do well by basing itself on physics. Less reliable alternatives would be to cover only topics in a narrow domain, cover only encountered topics in some convenient corpus of text, or to guess at which foundations that might span all known possibilities. Physics is the study of space, time, matter, motion, energy, and any other related phenomena (Baez 1967, p. 3), and certainly a large percentage of English sentences describe objects (which consist of matter) and motion, so physics is one of the most obvious foundations of descriptions of real-world events. Also, many authors (e.g., Pezzulo 2007; Berthoz 2000, p.22) have noted that brains of higher animals seem to simulate real-world events, especially for purposes of prediction, planning, and CSR, and to simulate is to represent events partly by relying on physics formulas, therefore physics is probably the best field on which to found a set of Building Blocks.

The most basic concepts in physics are often listed at the beginnings of physics textbooks (e.g., Baez 1967, p. 3), and include especially the following:

- spacetime: a single structure that includes space and time as separate dimensions
 - SUBCOMPONENTS: space, time
- mass-energy: physical objects
 - SUBCOMPONENTS: mass (m), energy (E)
 - CONSERVATION: approximately yes, for total energy of a system in classical mechanics
 - INTERCHANGABILITY: with energy via the equation $E = mc^2$
- momentum: a property of a physical object
 - SUBCOMPONENTS: mass (m), velocity (v)
 - CONSERVATION: yes, for an isolated system
 - INTERCHANGABILITY: with mass and velocity via the equation $p=mv$; with force via the equation $F = \Delta p/\Delta t$
- force: deals with the interaction of objects, not so much with single objects
 - SUBTYPES of force: contact force, non-contact force
 - SUBTYPES of force: strong nuclear, weak nuclear, electromagnetic, gravity (a pseudo-force)
 - CONSERVATION: not in general, but depends on the type of force
- charge: deals more with single objects, not so much multiple objects
 - CONSERVATION: yes, for an isolated system
- probability: used in quantum mechanics where waves consist of probabilities, not energy
 - CONSERVATION: no

Since Tumbug is intended to model primarily the human-centric view of the world, especially that of human intuition, Tumbug currently does not make a distinction between true forces (viz., electromagnetic force, strong nuclear force, weak nuclear force) and pseudo-forces (viz., gravitational force). Tumbug also uses its own version of spacetime, a much simpler concept than Minkowski space, largely because no quantified interactions (e.g., the formulas of relativity) between space and time are considered in Tumbug unless explicitly modeled. Also, Minkowski space is limited to four dimensions, which Tumbug space is not. Tumbug is able to model those more complex versions of physical phenomena, but such constraints are not built into Tumbug.

4.2.2 Mathematics

Physics is not the sole and ultimate science for modeling the real world, however, since the laws of physics are in turn described with mathematics; mathematics is assumed to be the descriptive foundation of physics. A few mathematical concepts used in physics that are seldom found in basic physics books but that are concepts that involve useful mathematics for modeling the real world are:

- combinatorics (from probability)
- randomness (from probability)
- connectivity (from topology)

A few of the more fundamental mathematical concepts needed to represent natural language concepts have been mentioned and included with Tumbug concepts in this document, but there may exist many more mathematical concepts, possibly infinite in number, that may be needed to describe real-world situations. In particular, more concepts from first-order logic, which is a subset of math, may well be needed in Tumbug since predicate logic is one of the oldest and soundest KRMs for AI systems (Cercione and McCalla 1987, p. 3). Already a form of the NOT concept (viz., XOR) and a form of the existential qualifier (viz., C-A Aggregation Boxes) from predicate logic have needed to be added to Tumbug in order to represent WS150 sentences.

4.2.3 Other systems and the Big Picture

In the big picture, humankind has recently begun to struggle with its ancient systems of representation such as mathematics and language to cope with the extreme advancements of science. Several authors have noted that some new type of mathematics is likely needed to cope with the very complicated concepts that modern science is uncovering (e.g., Bailey 1996, pp. 9, 26, 28; Devlin 1997, pp. 282-283; Wolfram 2002, p. 627; O’Neill 1981, pp. 48-49). Presumably the need for new KRMs is correlated with the need for new mathematics, since mathematics is itself a type of KRM.

Figure 29 shows the most basic fields from which each concept arises, where "most basic" means "most universal": (1) mathematics, (2) logic, (3) physics, (4) human world, (5) computer science. This justification is based on the observations that: (1) mathematics is applicable to any universe because it is based on nonphysical abstractions, (2) physics is universal for this universe, regardless of whether physical objects exist or not, (3) human perception and mental organization can detect natural patterns not covered by physics, (4) computer science is a human-created system of abstraction not covered naturally by the human mind.

See Figure 30 includes the same list with more detail, but with the corresponding Building Blocks of Tumbug, in all their higher and lower generalizations. The specific icons that implement these concepts have names that are capitalized. These icons visually tend to be a type of Box, Arrow, Circle, Bar, Line, or String. A few of these icons have not yet been used for any encountered sentences, but are believed to be essential for certain future extensions and modifications of Tumbug. Since those unused Building Blocks are untested, those are the Building Blocks most likely to have unforeseen drawbacks in their current form. All these Building Blocks of Tumbug are discussed in detail later in this document.

Relative to the importance of the five overall categories (S, C, O, V, A) that are clearly defined by SCOVA (see the section on SCOVA), the particular choices of categories to be considered Building Blocks of Tumbug are relatively unimportant, other than for historical value in that those icons gave rise to the more general SCOVA insight and are more frequently used than other categories with other levels of generality. For example, since a Physical C Object Circle and a Data C Object Circle can be used interchangeably with ease, with possibly nobody even noticing the difference in appearance of their respective icons (viz., a solid line versus a dashed line), it is not very

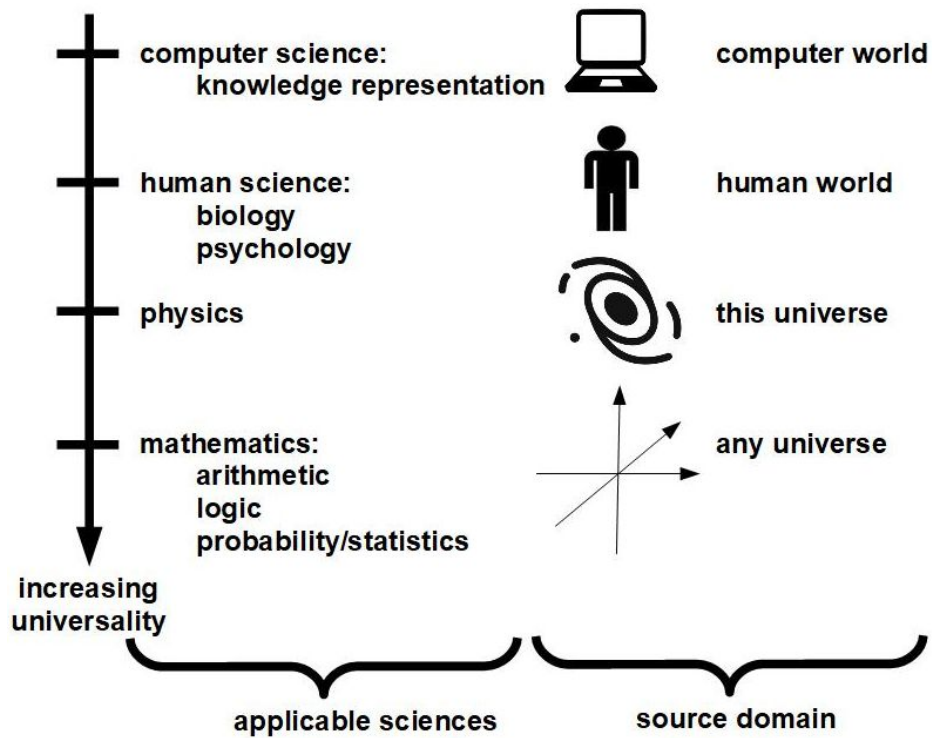


Figure 29: A summary of the originating fields of Tumbug icons, in order of increasing universality.

<u>Tumbug icon name:</u>	<u>description of concept:</u>	<u>main field of origin:</u>
Nonquantified Objects (O)	entities	
singleton objects	single object	human world
Object Circles	object	human world
C Object Circles *	regular object	human world
Physical C Object Circles	physical object	human world
Data C Object Circles	data object	CS
Data Points *	data	mathematics
C-A Object Circles *	abstract object	CS: KR
State Circles *	state	physics
Cells *	arrays	mathematics
Pathway Tubes *	pathway	CS: KR
Sensor Bars *	sensor	CS: hardware
Markers	marker	
Geometrical Markers *	marker	mathematics: geometry
0D Markers	point marker	CS: hardware
1D Markers	line marker	mathematics: geometry
2D Markers	region marker	mathematics: sets
Relationship Markers *	relationship	CS: KR
collection objects	multiple objects	human world
Location Boxes	space/place/location	physics
Verbatim Location Boxes *	pictures	human world
Descriptive Location Boxes *-	descriptions	human world
Aggregation Boxes	aggregation	human world
C Aggregation Boxes *	regular aggregation	CS: KR
C-A Aggregation Boxes *	abstract aggregation	CS: KR
Swirly Arrays *	array	mathematics: linear algebra
XOR Boxes *	NOT/negation	mathematics: logic
attribute-like Building Blocks (A)	attribute	
Attribute Lines *	attribute	human world
value-like Building Blocks (V)	value	
Value Bars *	pictorial magnitude/value	mathematics
Wildcards *	wildcard	CS: KR
Range Caps *	endpoints	mathematics
Change Arrows (C)	change	
Single Time Arrows *	time/timeline	physics
Motion Arrows *	motion	physics
Force Arrows *	force	physics
Causation Arrows *	causation	human world
Correlation Boxes *	correlation/function	mathematics
system-like Building Blocks (S)	system	
State Diagrams *	state diagram	CS: KR
Split Time Arrows *-	timeline split	mathematics: probability
Data Set Boxes *-	data set / value distributions	mathematics
convenience Building Blocks		
Label Strings *	label/text	CS: KR
Attend Rings *	ATTEND / focus of attention	human world
Motivation Triangle	wants/desire/motivation	human world
The Modal Verb Icon	modal verbs	human world
The Robinson Icon	emotions	human world
Zoom Boxes *	zoom	CS: KR

C = concrete
 CS = computer science
 * = considered a Building Block

C-A = concrete abstract
 KR = knowledge representation
 - = further development anticipated

Figure 30: The current Building Blocks of Tumbug and their originating fields.

useful to consider those different Building Blocks. At the other extreme, a Change Arrow is too general to be of practical value in translating sentences into Tumbug because, for example, time and cause-and-effect are very different concepts, so the Change Arrow icon is not considered a Building Block, only a Basic Building Block of theoretical value for categorization purposes. In turn, the exact count of Building Blocks in Tumbug is not particularly important, although an approximate count is useful for comparing Tumbug to CD theory, or in estimating the amount of time it would require to code Tumbug, for example.

Regardless of topic, the concepts and relationships between concepts must be represented somehow, and some system or systems must bear the burden of performing that task. Ideally there would exist a universal standard representation system, a system that Roger Schank sought for language. Since Tumbug largely follows this ideal, and since Tumbug shows signs of exceeding the representative power of CD theory, Tumbug may be the representation system needed to replace math in certain areas such as AI.

The goal of Tumbug is to collect all concepts of the real world that humans would reasonably want to express in language, and then to represent those concepts with images. Mathematics and its branches are obvious candidates for sources of concepts, and those fields have been used in AI for decades. Physics and model-based reasoning (e.g., Luger and Stubblefield 1998, p. 231) have also been used in AI, though to a lesser extent. The world of human beings, which includes psychology and sociology, contains concepts (especially the concept of "object" and "motivation") not covered in the hard sciences of inanimate objects, so Tumbug needs to include those concepts, also. Several knowledge representation concepts (especially general aggregations and labels) do not occur in their general form in the more general fields so computer science must be included. (For example, math has sets and parentheses, but does not generalize these notions into a general "aggregation" concept.) (See Figure 31.) Because of Tumbug's eclectic nature and the infinite nature of the real world, Tumbug is not guaranteed to cover every basic concept in the real world, but in practice its span is so extreme that the author has found no lacunas despite extensive perusal of several fields. If a lacuna does exist, it would probably be found in some of the less frequent concepts of mathematics, possibly from group theory or topology.

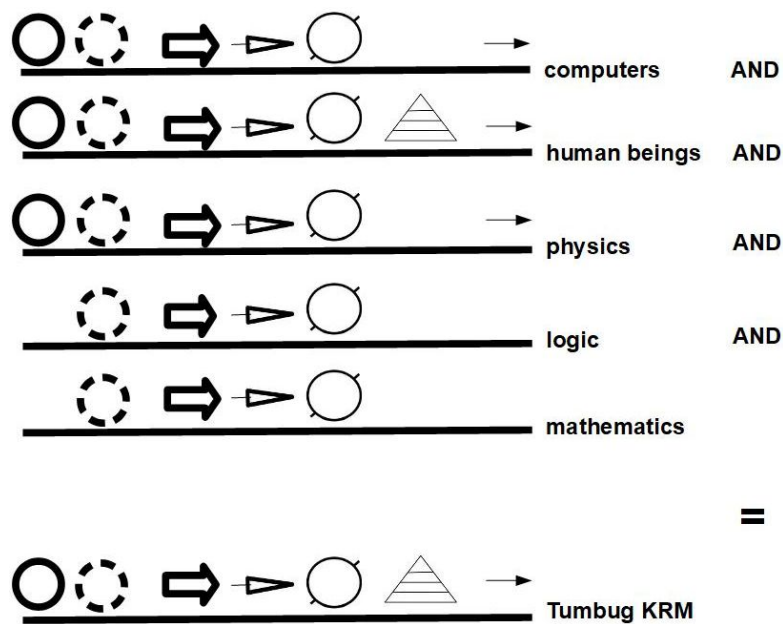


Figure 31: The Tumbug KRM considers this collection of the fundamental entities from several systems from the real world, all AND-ed together for to cover the basics of every one of those fields.

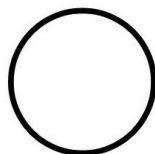


Figure 32: Tumbug's icon for a Physical C Object Circle.

5 Object-like Building Blocks of Tumbug (O)

The term "Building Blocks" comes from other authors' descriptions of Roger Schank's CD theory (Schank 1975), evidently not from Schank himself. More specifically, the term "building blocks" is used to describe all the components of CD theory, including the Primitive Acts and the Primitive Conceptual Categories. The term "Building Blocks" is used in this document in the same way as in CD theory: to describe Tumbug's basic components, especially since CD theory and Tumbug have roughly the same intent: to form a foundation on which any action can be described. Tumbug's Building Blocks were derived from generalizing many examples of sentence representation, primarily from the following sources: (1) WS150 problems, (2) CD theory, (3) Kolln grammatical examples, and (4) the author's own insights.

Terminology used in this document for CD theory:

Primitive Conceptual Categories = {PP, ACT, PA, AA, LOC, T}
Primitive Acts = {ATRANS, PTRANS, PROPEL, MTRANS, MBUILD, SPEAK, ATTEND, MOVE, GRASP, INGEST, EXPEL}
building blocks = Primitive Conceptual Categories \cup Primitive Acts

5.1 Singleton objects

5.1.1 Object Circles

The preliminary letter "C" is a formality that stands for "concrete." Usually C Object Circles are called simply "Object Circles."

1. C Object Circles

1.1. Physical C Object Circles

1.1.1. General

A Physical C Object Circle is shown in Figure 32.

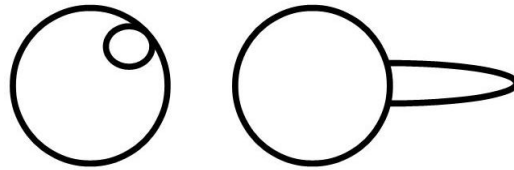


Figure 33: Left: Example of a physical object with interior structure. Right: Example of an object with exterior structure.

Computers ultimately represent everything with numbers. This is because computers evolved from calculating machines that work with only numbers, and because the foundation of all science is mathematics, which primarily uses numbers. (Some exceptions are group theory and topology.) Even images are represented in a computer by matrix-like file formats such as BMP, JPG, and GIF, which are ultimately only numbers arranged in an orderly manner in space. Mathematics, however, is only an abstraction of the real world. In the real world, the fundamental entities of importance that are perceived and mentally manipulated by animals are objects, which are different than both numbers and pictures.

It is probably this last observation that was primarily responsible for the development of object-oriented programming (OOP), which groups data into an abstract data structure called an "object" whose simulated behavior is similar to that of its corresponding real-world object. Tumbug likewise treats objects as the fundamental entity of importance. The Tumbug icon used to represent an object is usually a plain one-lined circle called a "C Object Circle."

Any objects where more detailed anatomy (e.g., taste buds on the object's interior, or an arm on the object's exterior) is important to represent can be drawn as bumps or protrusions on the perimeter of the C Object Circle, protruding in the appropriate direction, and can have any desired size, length, or shape, though typically a single truncated or elongated circle or ellipse is sufficient for the typical level of detail in WS problems. See Figure 33.

Very commonly an object, especially a person, needs to be represented as generating output (e.g., speaking, shouting, gesturing) or actively processing input (e.g., watching, looking, listening). Logically this would involve an anatomical structure that is partly inside and partly outside the body, so

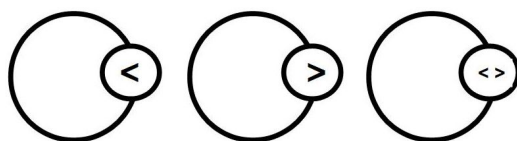


Figure 34: Left: Object with input sensor. Middle: Object with output generator. Right: Object with input-output.

Tumbug uses a smaller C Object Circle on the border of the entity's C Object Circle to represent such an input-output device. The symbol "<" can be placed on this structure to designate it as an input device (sensor), ">" for output, and "<>" for either input or output, as shown in Figure 34.

Why are Tumbug representations so simplistic and abstracted, instead of being more detailed and realistic, for example by showing arms and mouths? The answer is that Tumbug in a future software implementation would eventually need to view and interpret its own icons, therefore the simpler the icons that Tumbug must interpret, the better. This is especially true in language translation, where the Tumbug representation would need to be read after it had been stored, in order to make the final translation to the target language.

C Object Circles are allowed to overlap, although such a need tends to be uncommon. For example, two circular spots of light would logically be represented as two C Object Circles, and since spots of light can overlap, C Object Circles should also be allowed to overlap. Similar logic would apply to clusters of objects, where each cluster could be represented by one C Object Circle or by one Aggregation Box, and would apply to set boundaries as seen in Venn diagrams, where each set boundary could be represented by one C Object Circle or by one Aggregation Box.

This article's convention: Tumbug object icons are kept as simple as possible, as often as possible. For example, details such as hands on arms or taste buds in the mouth are typically never shown unless relevant to the problem, or unless explicitly stated in the sentence to be represented.

1.1.2. Icon variations

For the purposes of WS problem representation, Tumbug can be easily and optionally supplied with icons for any encountered concepts whatsoever, especially for physical objects. A predefined class of intelligent, human-like entities seems particularly useful for high-level, cross-ideology-set discussions

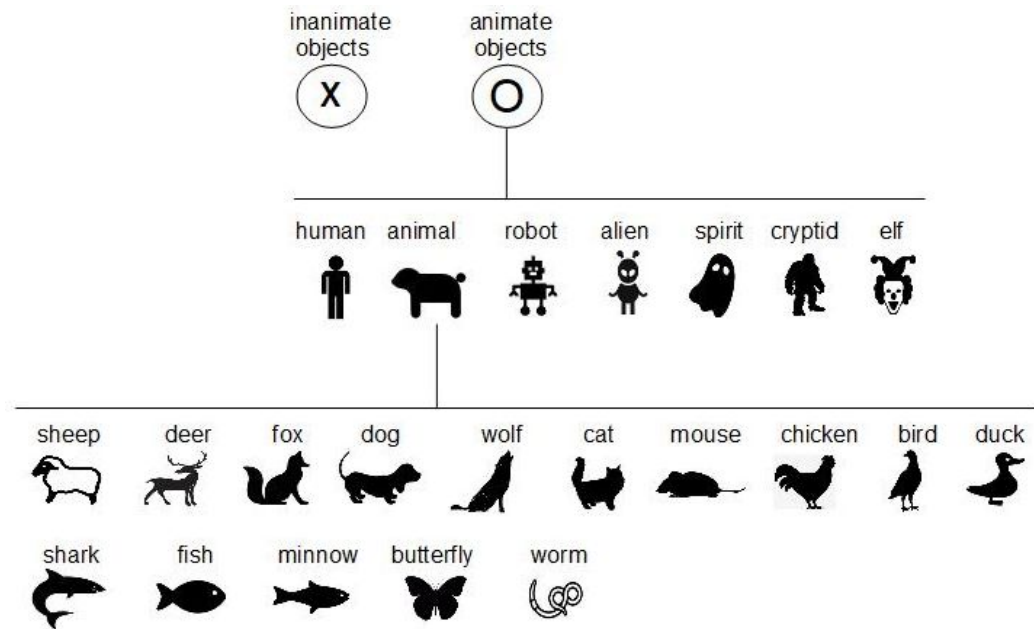


Figure 35: Elaborate hierarchies of types of animate objects can be created with icons for a Tumbug/WS library.

in philosophy, science, politics, and religion: human, generic animal, robot, alien, spirit, cryptid, elf. At the lower-level end of the hierarchy, the set of animals sheep, deer, fox, dog, wolf, cat, mouse, chicken, bird, duck, shark, fish, minnow, butterfly, worm would be particularly useful because those are all the animals specifically mentioned in WS150 problems, the solution of which one Tumbug-based algorithm was originally designed to provide. The hierarchy of Figure 35 shows animate object types and their icons is only one example of the endless elaborate hierarchies that can be created.

The exact icons used are not critical, though the existence of such a hierarchy of objects is useful. Such a library is useful because the more specific the Tumbug icons are, the more readily a Tumbug diagram can be understood. One downside of any hierarchy, however, is that a hierarchy is necessarily rigid, therefore any unexpected item (such as a spork) is automatically excluded from fitting into any existing classes (such as spoon or fork). In order to keep this presentation simple, the circle with "X" and circle with "O" are not used elsewhere in this document, but in other documents such

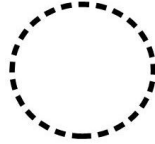


Figure 36: Tumbug's icon for a Data C Object Circle.

a distinction could be quite useful.

Roger Schank referred to objects in CD theory as "PP," which stands for "picture producers," meaning objects that have a physical appearance that typically comes to mind upon hearing or seeing the word for that concept. This term suggests that use of icons would be particularly beneficial to use in place of Physical C Object Circles in Tumbug whenever possible. Throughout this document, icons may appear in place of Physical C Object Circles.

As an aside, the icons of Tumbug can be grouped into different types:

1. geometrical icons - very simple circles, rectangles, arrows, lines, etc.
2. composite geometrical icons - organized groups of geometrical icons
 - (a) composite tangible geometrical icons - for articulators, sensory organs, etc.
 - (b) composite intangible geometrical icons - for emotions, verb modality, etc.
3. realistic icons - for cats, dogs, computers, bottles, etc.

1.2. Data C Object Circles

1.2.1. General

A complication in the real world, especially in the modern world, is that many of the "objects" being used are collections of data, not physical objects. This distinction can be fairly important in the meaning and implications of sentences, therefore Tumbug modifies the Physical C Object Circle icon so that the circle has a dotted border (or sometimes a dashed border) when it represents a label-worthy chunk of data, as shown in Figure 36. As with physical objects, data objects can have attributes, such as topic, format, size in number of bytes, truthfulness, source, and so on.

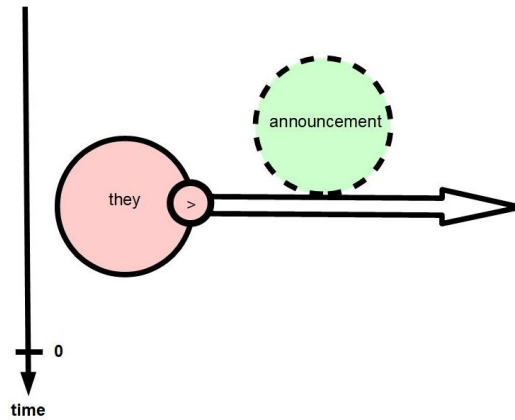


Figure 37: [35] Tumbug for "They broadcast an announcement."

Data C Object Circles can be thought of as envelopes or flash drives that hold their contents inside the circle. For example, a digital photo is a data package whose contents could easily be represented verbatim inside a Data C Object Circle by placing the photo inside a Verbatim Box inside the C Object Circle.

1.2.2. WS150 example: #35 (subway)

The example in Figure 37 is from a portion of WS150 question #35. A Data C Object Circle is necessary because data is implied via the words "broadcast" and "announcement."

"[35] They broadcast an announcement, but a subway came into the station and I couldn't hear it. What couldn't I hear? POSSIBLE ANSWERS: {the announcement, the subway}"

As with physical objects, data objects have direction of travel, although their transmission pattern can be more flexible since data without a carrier path they can either be transmitted in a straight line (as with a laser beam modulated to carry a message) or in a spherical plane (as with a loudspeaker broadcasting a message). Dashed planes are more difficult to draw than dashed lines, so a Tumbug diagram uses three representative lines with a single Data C Object Circle and a single Motion Arrow to help convey the concept of an emanating plane wave using simpler icons that are already used in simpler scenarios.

Conceivably a system of icons could be developed for data objects as

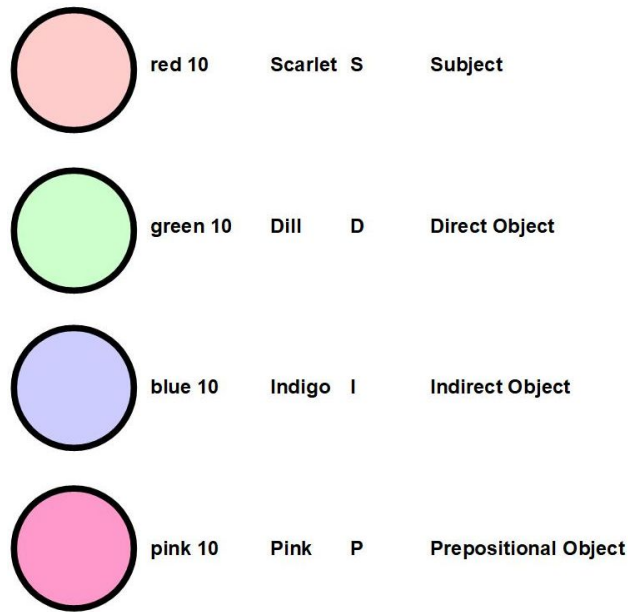


Figure 38: This is the Tumbug color coding scheme used in this document for the different types of grammatical objects, and the underlying logic for this scheme. Colors are optional in Tumbug, however.

well as physical objects. For example, speech (which is particularly common in WS examples and grammatical examples) could have its own icon, as could vision.

This article's convention: Although Tumbug diagrams never need to be in color, a color-coding convention has often been used on C Object Circles in this document to aid understanding of grammatical examples, as shown in Figure 38 and Figure 39.

1.2.3. Data Points

More general Building Block: Data C Object Circle

Simplifications from the more general Building Block: (1) Only 1-2 attributes used, namely the attribute-value pairs at which it can be plotted, and the event(s) from which the measurement was taken.

Figure 40 shows a specific Data Point with two attribute-value pairs. Since a single data point may have arisen from more than one event or observation or measurement, in general there may be a list of event IDs in

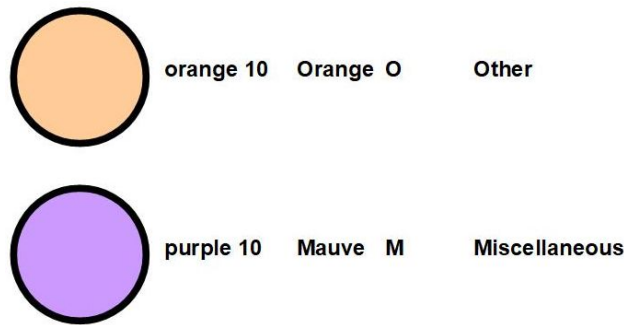


Figure 39: This is the Tumbug color coding scheme used in this document for other types of objects, and the underlying logic for this scheme. Colors are optional in Tumbug, however.

the event ID field.

2. C-A Object Circles

The Tumbug icon for a C-A Object Circle is shown in Figure 41.

”C-A” stands for ”concrete-abstract,” which is a pair of concepts at the two extreme ends of the abstraction spectrum. Both C Object Circles and C Aggregation Boxes can be abstracted to produce C-A versions of those icons. In both cases some attribute-values (those protruding above the associated icon) are forced upon the object by the user, and the other attributes (those protruding below the associated icon) are the resulting detected attribute-values. The value of the outputs may or may not have been influenced by a Correlation Box. For example, an output could be a default value based on the object’s class, and untouched by a Correlation Box.

An example of usage of a C-A Object is shown in Figure 42. Other uses of C-A objects can sometimes be found in the WS150 problems. For example:

”[9] The large ball crashed right through the table because it was made of steel. What was made of steel? POSSIBLE ANSWERS: {the ball, the table.}”

In this case, the composition of the unknown object is fixed at steel, which forces the object’s weight and strength to be very high. Since tables tend to be made of wood, which forces a table’s weight and strength to be only moderate (or at most, high), the ball described will tend to have more weight and strength than the table, which allows default reasoning to occur

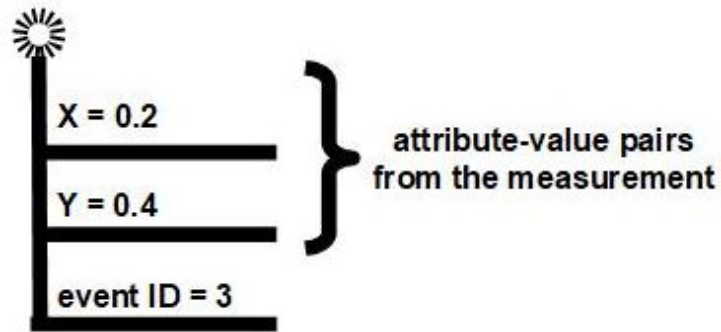


Figure 40: A Data Point is a small Data C Object Circle that contains only attribute-value pairs and optionally a reference to the event that produced it.

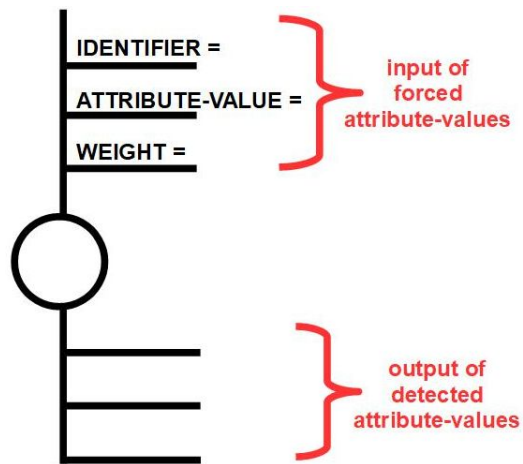


Figure 41: Tumbug's icon for a C-A Object Circle.

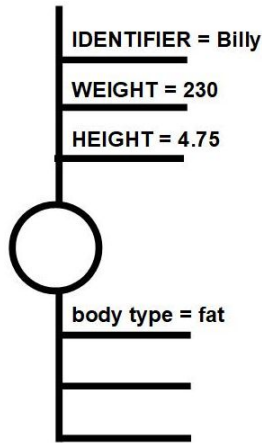


Figure 42: If Billy’s weight is fixed at a high value and his height is fixed at a low value, then he is forced to be fat.

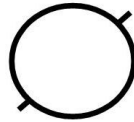


Figure 43: Tumbug’s icon for a State Circle.

and for the correct default answer to be chosen.

5.1.2 State Circles

Figure 43 shows a State Circle

In Tumbug the need to represent states is almost as common as the need to represent objects. This is because verbs apply to states almost as frequently as they apply to objects, and the best icon to use is the icon whose concept most closely matches the sentence meaning, even if the specific words of the sentence and their grammatical order superficially suggest otherwise. The State Circle icon used for a state in Tumbug is intended to resemble the Greek phi symbol (ϕ), which is used in physics for phase, which is a concept that is similar to state. The similarity between the C Object Circle icon and State Circle icon is intentional since it is sometimes desirable to use a State



Figure 44: Tumbug's icon for a Pathway Tube.

Circle instead of a C Object Circle, so swapping one for the other allows the viewer to overlook the minor conceptual difference when interpreting a Tumbug diagram quickly.

Ordinarily a State Circle is only a part of a state diagram, as described in the section on State Diagrams.

5.1.3 Cells

The meaning of "Cell" in Tumbug is identical to the meaning of "cell" in array terminology: A cell is a discrete location that can hold a single object. The object may be a number (as in linear algebra), a text string (as in data bases or spreadsheets), software objects (as in OOP), or any other type of object, virtual or real. Cells are usually used only in composite structures such as vectors, arrays, or jagged arrays.

The Cell concept differs slightly from the Tumbug concepts of Verbatim Box and C Aggregation Box since a Cell may hold only one object whereas those other structures may hold any number of objects. A Cell is similar to a State Circle, except the contents of a State Circle must be a state whereas a Cell can contain an object that is not a state. A Cell could also be accurately called a "pigeonhole," as in the "pigeonhole principle," since a cell conforms to the same constraint of one object per location.

5.1.4 Pathway Tubes

Figure 44 shows a Pathway Tube.

A Pathway Tube shows a pathway for anything. It can represent a hiking path used by people, a pipe for carrying water, a track for carrying trains or monorails, a fiberoptic path used by photons, a data path in the form of a USB cable between digital computers, a token path used by Petri nets, the arrows of logic flow direction in a flowchart, a wave guide for electromagnetic waves, or other. The pointed end of a Pathway Tube shows the direction of travel of objects or data within the tube/pathway, like a one-way street sign.



Figure 45: Pathway Tubes normally contain a 0D Marker, like the tokens of Petri nets.

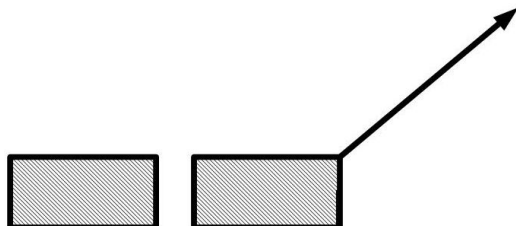


Figure 46: Tumbug's icon for a Sensor Bar (left), and Sensor Bar with a signal output line (right).

The objects moving in a Pathway Tube can be represented by 0D markers, as shown in Figure 45.

Pathway Tubes in this document are always shown as straight, but in general they need not be. Straight Pathway Tubes can be connected end-to-end to implement changes in direction, if for example a vector graphics editor does not a simpler means to create such a shape.

5.1.5 Sensor Bars

A Sensor Bar is shown in Figure 46.

Tumbug contains Value Bars to represent values, but these values cannot be used unless the values can be read somehow. Sensor Bars solve this problem. Each Value Bar (by current convention) is typically outfitted with a Sensor Bar that has a signal output line to represent a signal being sent from the sensor to some other component if and only if that sensor is activated.

Typically a Sensor Bar is overlaid upon a Value Bar so that if the value in the Value Bar falls within the span of the Sensor Bar then the Sensor Bar sends an output signal (marked here as "*"") on its output line, as shown in Figure 47 and Figure 48.

In turn, a single activated state within a state diagram may trigger further activity within that state diagram, or possibly within a program

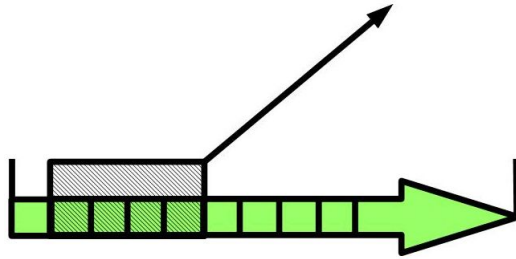


Figure 47: A Sensor Bar overlaid upon a Value Bar.

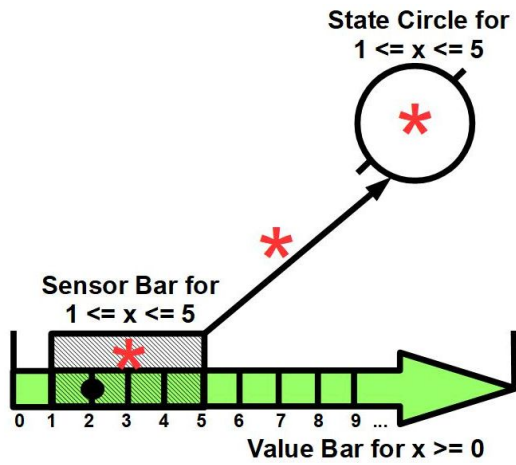


Figure 48: A Sensor Bar senses a value in the specified range within a Value Bar and sends a signal to a State Circle.

containing an "if" statement that relies on that state being active in order to choose with branch of code to execute next.

Value Bars and Sensor Bars are currently proposed for Tumbug for completeness purposes, but for diagram simplicity it could be assumed that the system automatically activates whichever applicable state needs to be read by the system, and does so in the background, without need to illustrate the details of how that state became activated.

Sensor Bars in this document are shown with a -45-degree hatching pattern, whereas 2D Markers are shown with a +45-degree hatching pattern.

5.1.6 Markers

Technically, a Marker in Tumbug can be defined as a labeled manifold that marks something of significance in a Tumbug diagram. For example, a non-intersecting border drawn around an irregular region in a plane would be a 1D manifold in 2D space that would bear a label mentioning that it is a border, and typically also what the border surrounds. Markers can be of any dimension, and can occupy any dimension, and can have any non-intersecting shape. For example, a wire that traced the edges of a collection of stacked cubes would be a 1D manifold in 3D space. Markers can also have unbounded ends, such as an arrowhead at one end of a 1D line segment, or a 3D arrowhead pointing outward at the border of a 3D planet's atmosphere.

1. Geometrical Markers

1.1. 0D Markers

More general Building Block: C Object Circle

Simplifications from the more general Building Block: (1) Only 1-2 attributes used, namely location, and optionally priority.

The zeroth dimension (0D) is a point, so in Tumbug a 0D Marker is a single point that is used to highlight some item or location. This 0D Marker icon is currently represented in Tumbug as a sizable red spot, as shown in Figure 49.

Some practical uses of 0D Markers are:

- A checkmark of any kind. (Note that this implements Kenneth Haase's red paint idea described by Marvin Minsky (Minsky 1986, p. 82).)
- A red dot on a map labeled "You are here."



Figure 49: Tumbug's icon for a 0D Marker.

- A radar blip that indicates the position of an aerial object detected by radar.
- A bouncing dot over lyrics to help singers keep track of the current part of the song.
- An "X" on a treasure map that marks the spot where the treasure is buried.
- A spot like the spot from a laser pointer that shows the part of an overhead slide being discussed by the lecturer.
- Color-coded labels on unmarked cardboard boxes.
- The desired attribute of a query.
- The current state within a state diagram. If the current state is in transition between two states then the 0D Marker can also represent this easily by being placed on the Pathway Tube itself.
- The current statement being executed within a flowchart.
- A token within a Petri net.
- A mental marker to remember a possible king position on a chessboard when visualizing possibilities in a chess endgame.
- A marker like a painted stone on a trail that indicates a place that was formerly visited, to prevent going in circles if lost on a hike.
- A position affixed to the rim of a rolling circle.

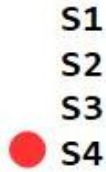


Figure 50: One common usage of a 0D Marker is to step sequentially through program statements.

- The conjectured K-Lines of Marvin Minsky (Minsky 1986, p. 82) to mark tools that were used for a given job.

One use of 0D Markers in Tumbug in this document is to represent instruction pointers in an executing computer program. See Figure 50. Whereas a digital computer uses an instruction pointer that points to the instruction currently being executed in a program, Tumbug uses a small red dot as a visual means to mark this instruction in the program itself. Then, instead of incrementing an address in computer memory to step through a program, Tumbug moves a spot sequentially through a program. Since the program structure is already represented as a graph-like flowchart, use of a 0D marker in such a flowchart maintains the higher-level view of the program as completely visual.

An important use of 0D Markers is in diagramming queries. For example, a data base type query might equate to the question "What color is Bob's car?", which would place a 0D Marker with an "information goal" attribute atop the value of the "color" attribute in the data base record for Bob's car, then the value of that attribute would be noted. In this way an entire question reduces to a single placed 0D Marker. See Figure 51, where the color red here substitutes for the attribute "information goal." Note that a single spot—the 0D Marker—is difficult to see on a diagram at a glance, therefore a clearer representation is recommended in practice, which is the representation with the slanted line with one end on the desired value and the other end marked with "DK" (= Don't Know). (Currently, use of "?" to represent queries in Tumbug is discouraged since "?" has a different meaning as a regular expression wildcard.) Since all Tumbug lines are normally drawn either horizontally or vertically, a 1D Marker line that is slanted at about a 45-degree angle stands out at a glance.

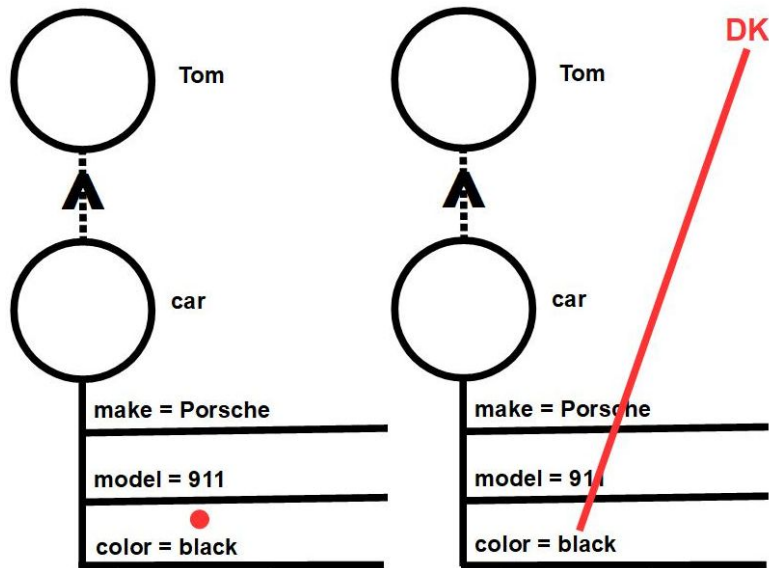


Figure 51: A slanted line in Tumbug is much more visible than a tiny 0D marker when diagramming queries.

In WS150 problems the queries are only slightly more complicated in that they are always about disambiguation, but 0D Markers can be used there, also. For example: "[108] Grace was happy to trade me her sweater for my jacket. She thinks it looks great on her. What looks great on Grace? POSSIBLE ANSWERS: {the jacket, the sweater}" In this query a 0D Marker would be placed atop value of the "clothing type" attribute for the item of warm clothing that Grace is wearing after the trade, with the same attribute "information goal" attached to the 0D Marker, then the value of that attribute would be noted. Oftentimes such queries require following an extra link. In this case, Grace would first need to be located in the Tumbug diagram, then the 0D Marker placed on her warm clothing.

1.2. 1D Markers

1D Markers are lines used as a visual aid to mark something as lying in a straight line. In the WS150 problems such lines would be desirable to show visual interference along a line of sight, for example. The lines could also be useful to show alignment, or to show an angle from a given point.

Some practical uses of 1D Markers are:

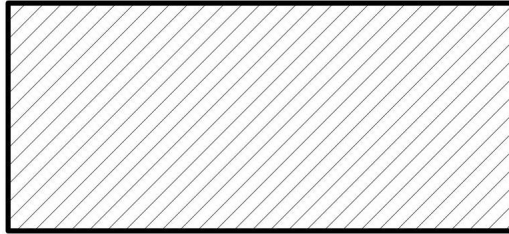


Figure 52: Tumbug’s icon for a 2D Marker.

- A straight line that makes clear the alignments of objects or text, whether through their tops, bottoms, centers, or other.
- A straight line that separates plotted data points into classes.
- A straight line across an opening in a concave geometrical figure, where the line separates interior from exterior.
- A straight line between extrema in a convex geometrical figure, where the line shows regions that can be obscured via occlusion from certain angles.

1.3. 2D Markers

2D Markers are merely shaded regions, as in Figure 52. Such regions are probably most often used for Venn diagrams, such as to show intersection of two sets as in Figure 53, or to show the conceptual allowable range of elements of one set. 2D Markers in this document are shown with a +45-degree hatching pattern, whereas Sensor Bars are shown with a -45-degree hatching pattern.

Some practical uses of 2D Markers are:

- A circle around an important region within a picture or within text.
- A ”set cordon”—an irregular contour that shows which parts of a Venn diagram are being considered.
- A contour that encloses data points that are considered part of a given classification.
- Boundaries between regions on a geographical map.

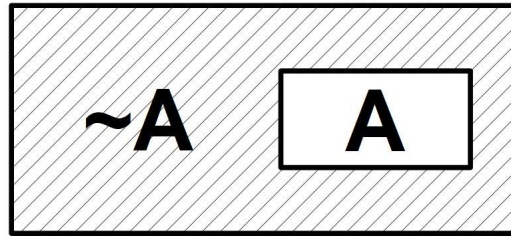


Figure 53: 2D Markers are usually used to show 2D ranges within sets represented by Venn diagrams.

- The four borders on a JPG image that is being cropped.

2. Relationship Markers

See Figure 54. Relationship Markers use dotted lines, which no other Tumbug convention uses, which is done for uniqueness, clarity, and consistency. The consistency arises because a 0D Marker can already be considered a Relationship Marker with a single point, so two points could be considered a 1D Marker, three non-colinear points could be considered a 2D Marker, and so on.

Relationship Markers are very generic markers that point out any important relationship that the user would like to flag. Depending on how Relationship Markers are used, Relationship Markers may be optional or required.

In Venn diagram representations of First Order Predicate Calculus, a Relationship Marker would almost be required in some cases. For example, the final inference of the "Socrates is mortal" example already shows Socrates as an element of a subset of both men and mortals, but the user may have overlooked this 2-step removed inference during the derivation of that final result since each addition of a Venn diagram circle potentially creates several new inferences simultaneously, so the important final inference should be explicitly pointed out by a unidirectional Relationship Marker pointing from Socrates to mortals.

Relationship Markers often have direction. For example, the sentence "Bill is Mark's uncle" must be stated unidirectionally in text in the form "X is Y's uncle" since the converse "Y is X's uncle" is not true, therefore one arrowhead (centralized on the arrow's stem) is used to point in the needed

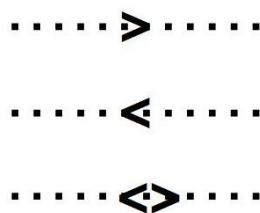


Figure 54: Relationship Markers have centered arrowheads on dotted lines, in any of three possible arrangements.

direction (per the user’s convention), such as from X to Y to show the direction of meaning of ”X is a/an of Y.” Some sentences are bidirectional, however, such as ”Denny and Wanda are siblings,” therefore in that case two arrowheads pointing in opposite directions are used to show both directions of unidirectional meaning simultaneously.

Relationship Markers are also nearly required for genitive case, meaning possessive situations, such as ”Sam’s drawing” [14], ”Ann’s son” [27], and ”Joe’s uncle” [28], all of which are examples from WS150. (The version of Tumbug described in the 2022 article on Tumbug described such possessive use of Relationship Markers as ”genitive arrows” (Atkins 2022), but Relationship Markers are more general than just genitive case, such as in the phrases ”Socrates is mortal” or ”Denny and Wanda are siblings.”)

Relationship Markers are essentially the same node-with-link data structures used in semantic networks. Since many sentences and phrases can be easily represented with semantic networks, this suggests that a good use of Relationship Markers might be as an intermediate stage of conversion from text to Tumbug.

Relationship Markers could be considered a compound type of icon since Relationship Markers contain 1D lines that are bent into 2D.

5.2 Collection objects

5.2.1 Location Boxes

”Location Boxes” is a generic term of Tumbug that currently refers to three types of rectangular icons in Tumbug, each type of which is intended to be a border in which to hold multiple items inside, to show the locations of those items inside the box, and to show the location of the Location Box

itself, at least relative to any other icons that may exist outside the box. Figure 55 shows that these three types of Location Boxes exist in a spectrum of severity of location constraints. These boxes may be any shape that is needed, although rectangular is the default shape.

- Verbatim Box: every item inside must exist only in the place where it is shown. There is no flexibility.
- Descriptive Box: every item's location is subject to constraints that are specified with the box. There is some flexibility.
- Aggregation Box: any item is allowed to be in any location within the box. There is complete flexibility inside the box.
- no box: there are no constraints at all on the locations of the items. There is complete flexibility and there is not even a border.

Although the two extremes of this spectrum, namely Verbatim Boxes and Aggregation Boxes, are conceptually simple and fully defined, the middle of this spectrum, namely Descriptive Boxes, have tremendous flexibility, so would need more research to fully investigate. For example, Figure 55 shows that four Object Circles in a square arrangement can be scaled to produce an unmistakably similar arrangement, but scaling is just one geometrical transformation. Another geometric transformation that would produce an unmistakably similar arrangement is translation. The result of rotation may also be considered an unmistakably similar arrangement, depending on a person's taste and goals. Figure 56 shows several such geometrical transformations, and which common transformation groups include those basic transformations.

Further development anticipated: Since different types of Location Boxes are allowed to overlap or to be embedded in one another, precedence rules are likely to be needed, the same issue as scope of name binding in computer programming. Probably Verbatim Boxes at the strict end of the strictness spectrum would need to take highest precedence, for example.

This document usually assumes that Descriptive Boxes: (1) retain the same spatial arrangement of the original items (i.e., allow differences in scaling), (2) disallow rotation, (3) disallow reflection and any other more complicated transformations such as projective transformations. Rotations are

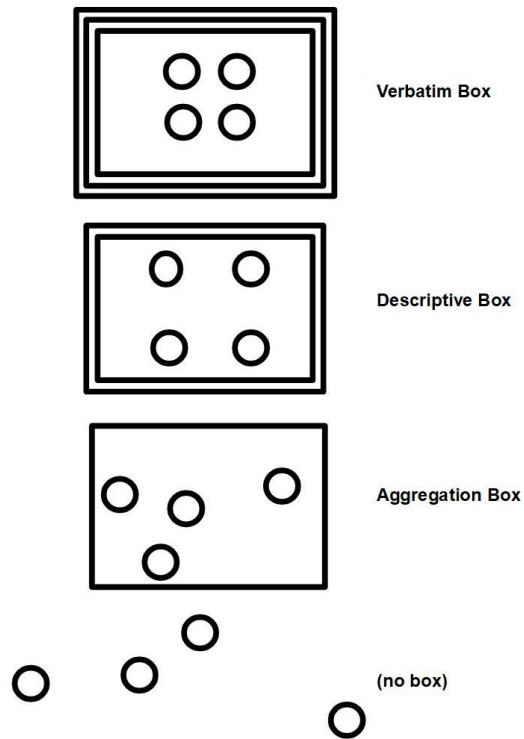


Figure 55: Tumbug has three types of Location Boxes that span a spectrum of severity of constraints on the locations of the items inside. Here the original locations of four Object Circles are shown by the Verbatim Box, this square arrangement has been scaled to be larger but still maintains the same shape in the Descriptive Box, and in the Aggregation Box there are no constraints on location at all except that the Object Circles must be inside the box. If not even an Aggregation Box exists then no Object Circles have any location constraints whatsoever.

TRANSFORMATION GROUPS:	BASIC GEOMETRIC TRANSFORMATIONS:				
	<u>translation</u>	<u>scaling</u>	<u>rotation</u>	<u>reflection</u>	<u>shear</u> <u>projective</u>
proper rigid transf.	+		+		
rigid transf.	+		+	+	
affine transf.	+	+	+		+

Figure 56: There exist several common types of transformation groups, each with a unique name and with a unique set of more basic transformations (translation, scaling, rotation, etc.) that are considered part of that group.



Figure 57: Tumbug's icon for a Verbatim Box.

disallowed partly because each diagram is usually assumed to contain a gravity vector that points straight downward on the page, which in turn suggests that depicted items are usually viewed from the side, and are either resting flat on something or have an inherent bottom (such as buildings, transportation, people, animals, plants, and natural geographical features). Ideally all such assumptions should be stated somewhere and somehow within each Tumbug diagram or each project, which in turn suggests that it would be useful to attach codes to Descriptive Boxes, such as placing an "R" in the corner of a Descriptive Box to indicate that all objects in that box have the freedoms of transformation of the "Rigid transformations" group.

1. Verbatim Boxes

A Verbatim Box icon is shown in Figure 57.

1.1. Concepts that imply space

It should be clear that the concept of space is common in the WS150 problems. Below, two examples are given for each of the concepts of interior/exterior and direction. Together these two concepts imply the need to represent spatial relationships in some parts of some sentences. The Tumbug icon used to represent places inside of it is a double frame or a triple frame to suggest a picture frame. Any objects where space (i.e., interior/exterior or direction) is important to represent must be placed inside of this frame, which can have any desired size or length-to-width ratio. Some concepts that inherently imply space are: (1) interior or exterior, (2) direction, (3) position, (4) alignment.

The Verbatim Box is where *bona fide* images are contained, therefore this is where Tumbug has its best prospects of demonstrating that its KRM is more powerfully expressive than any other existing KRM. "*Bona fide*" images in this context mean images other than predetermined icons or text. For example, a textual description such as "two circles of diameter 1 have their centers aligned along the 64-degree angle line, and their centers are 2.7

diameters distance from each other” is a *bona fide* image that potentially spans an uncountably infinite number of similar images by merely changing two numerical values. In this way Tumbug can output an uncountably infinite number of images. Similarly, in applications where the goal is to depict a default scene to aid the viewer’s comprehension of that scene, any photo whatsoever may be inserted verbatim into the Verbatim Box. For example, in a Jonny Quest animated cartoon, stock real jungle scenery (and even background sounds) could be used as a background wallpaper within the Verbatim Box to make the cartoon more realistic. Even a video could be placed in the Verbatim Box, such as a video of rapidly passing city buildings as background for an animated film where the characters in the foreground are riding aboard a train in the city.

1.1.1. Interior

This example is from a portion of WS150 question #52. A Description Box is necessary because the general interior of one object (viz., the fish) is implied by the word ”ate,” but no exact pictorial depiction of the event was provided.

”[52] The fish ate the worm. It was tasty. What was tasty? POSSIBLE ANSWERS: {the worm, the fish}”

The verb ”eat” would be defined in object-and-action terminology as something like ”Subject eats Object = Subject arranges for Object to go inside Subject’s mouth, whereupon Subject closes Subject’s mouth to entrap Object, then moves Object toward Subject’s stomach to be digested.” Since any such definition involves the concept of ”inside,” which implies the concept ”interior,” direction is inherent in the sentence meaning, therefore some type of Descriptive Box should be used, as shown in Figure 58.

In Figure 58, the fish has moved toward the stationary worm in time and has engulfed the worm into an anatomically interior position such as a mouth or stomach. This interior structure is represented by the smaller C Object Circle inside of the larger C Object Circle that represents the fish as a whole. The eaten worm then is represented as an even smaller C Object Circle inside of the fish’s interior C Object Circle.

In practice, any juxtaposition that involves ”interior” tends to be so clear that the surrounding Description Box can optionally be removed from the diagram without confusion. Figure 59 shows a version of the fish diagram that removes the Location Box and—even more visually effective—uses realistic icons instead of C Object Circles.

The fish diagram with realistic icons is more standard for Tumbug, de-

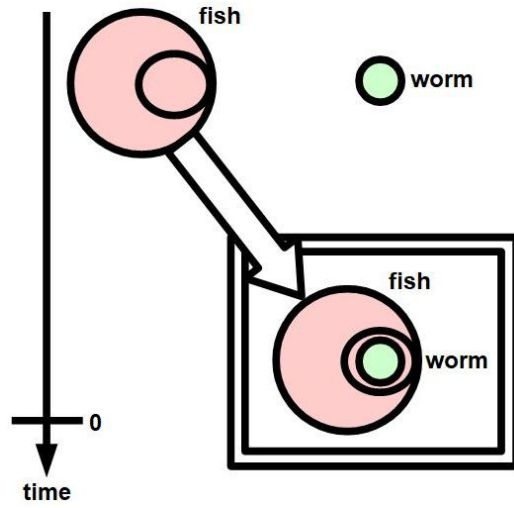


Figure 58: [52] Tumbug for "The fish ate the worm." (with abstract icons)

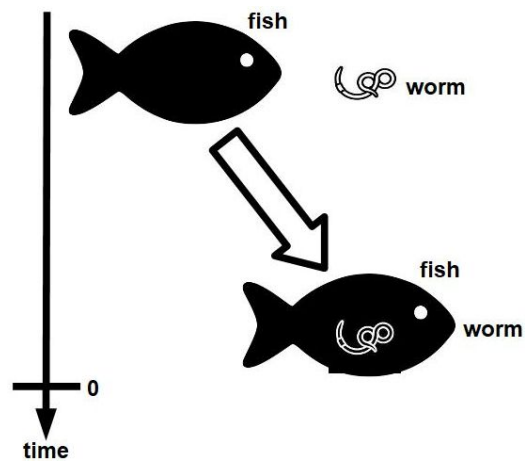


Figure 59: [52] Tumbug for "The fish ate the worm." (with realistic icons, and without emphasizing space)

spite the intermediate positions in time not being shown. Any other 2D temporal representation, viz. an icon in discrete time or an icon in continuous time, will not be clear since the separate time axis in 3D space would not be shown, which would force a moving object to overlay part of itself in the same plane, which in total will result in a smeared, lengthy composite image. The most practical option for simulations is to render real-world time the same as rendering simulation time, so that an object moving in the real world will be seen as an object moving in the simulation. For applications that require careful consideration of all points along a trajectory, such as determining whether several rotating disks in a plane, each of different radius and speed, and each with a visible spot near its rim, will ever align, rendering time as a separate dimension might be preferable.

Ideally Tumbug should be implemented in at least 3D space so that the states of moving 2D objects can be seen easily and collectively across the entire time span.

1.1.2. Direction

This example is from WS150 question #11. Some type of Location Box is necessary because the relative direction of an object is specified via the words "down" and "top."

"[11] Tom threw his schoolbag down to Ray after he reached the top of the stairs. Who reached the top of the stairs? POSSIBLE ANSWERS: {Tom, Ray}"

If the phrase about the stairs is omitted for simplicity, then the partial sentence would be diagrammed in Tumbug as in Figure 60.

The "0" on the Time Arrow represents "now," so the fact that the event completely finished in the time before time 0 means that the event happened in the past. The direction of the backpack is straight down because the sentence says "down" and no mention or implication was made of the throwing having a horizontal component; if the schoolbag had also been thrown in a horizontal direction as well as downward, then the trajectory of the schoolbag would be shown as angled, say toward the right of the diagram, as it approached Ray at the bottom, who would also be positioned more toward the right.

Further development anticipated: Note that throwing the schoolbag requires a force, and gravity would be exhibiting a smaller force even without throwing. So far Tumbug examples have not attempted to show all applicable forces, the resulting force vector after adding component forces, or even different length arrows that are proportional to the different force magni-

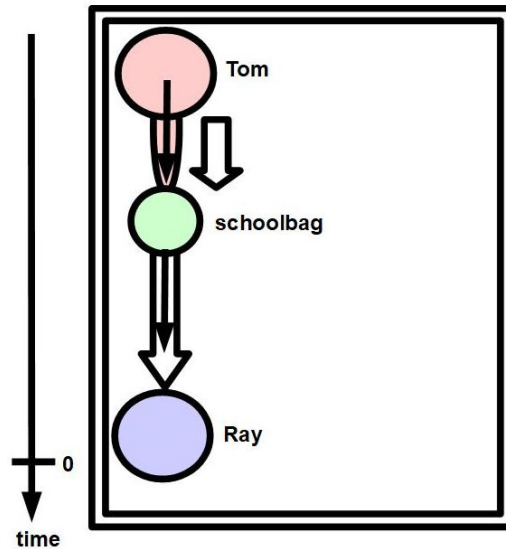


Figure 60: [11] Tumbug for "Tom threw the schoolbag down to Ray."

tudes. Ideally such fine details should be worked out eventually, though in the WS150 examples the need to differentiate between force magnitudes does not ever appear to arise.

The full sentence would be diagrammed in Tumbug as in Figure 61.

Note that a longer Time Arrow was used in order to show two events, one of which happened after the other. Each event—(1) the reaching of the top of the stairs, and (2) the throwing down of the schoolbag to Ray—is enclosed in its own C Aggregation Box to make the situation conceptually simpler. Since the person called "he" was not identified in the sentence, the diagram does not yet assign a color to the "he" C Object Circle, or assign the name on that "he" C Object Circle to "Tom." (That is for CSR to resolve at a later stage of processing.) The word "after" would be implied by the relative times of the two events. As before, relative to the current time "0," both events happened in the past. The word "top" is understood to mean "top of the diagram" since the direction "down" has already been established as the bottom of the page, and since "top" is defined as the opposite direction of "bottom."

1.1.3. Position

Figure 62 shows an example from a portion of WS150 question #19.

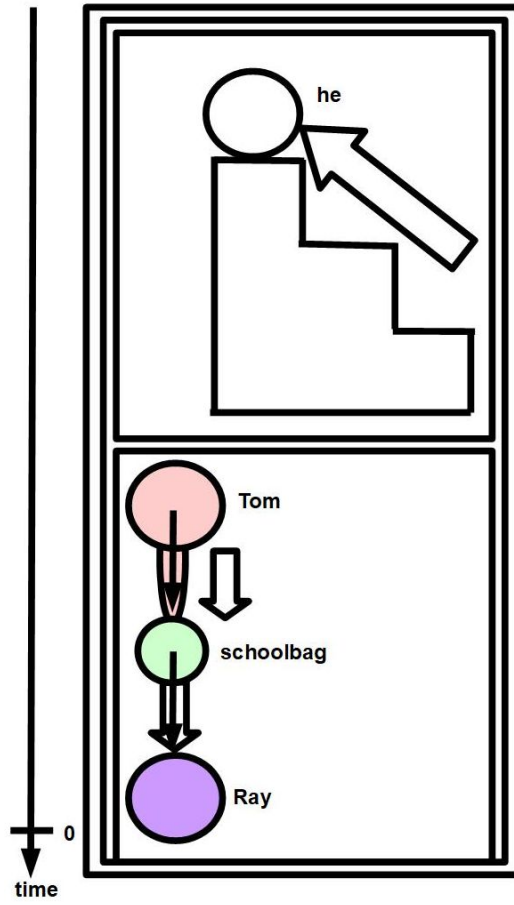


Figure 61: [11] Tumbug for "Tom threw the schoolbag down to Ray after he reached the top of the stairs."

Some type of Location Box is necessary because the relative position of an object is specified via the word "above."

"[19] The sack of potatoes had been placed above the bag of flour, so it had to be moved first. What had to be moved first? POSSIBLE ANSWERS: {the sack of potatoes, the bag of flour}"

1.1.4. Alignment

Figure 63 shows an example from a portion of WS150 question #34. Some type of Location Box is necessary because the relative alignment of objects is specified via the word "between."

"[34] There is a pillar between me and the stage, and I can't see it. What can't I see? POSSIBLE ANSWERS: {the stage, the pillar}"

The dotted line is technically a 1D Marker, used to help the viewer can discern that the three objects are aligned.

2. Descriptive Boxes (further development anticipated)

The Tumbug icon for a Descriptive Box is shown in Figure 64.

Whereas a Verbatim Box shows exact, inflexible locations of parts of images, a Descriptive Box is more flexible and therefore more general in that it uses links that describe spatial relationships via text instead of rigidly placed objects in space. See Figure 65.

Descriptive Boxes of any type may be arranged and nested arbitrarily. For example, to show the upstairs and downstairs of a house, two such Descriptive Boxes would logically be stacked. Similarly, to show two regions of a picture where each region could contain only one type of item, but each item could be in any location within that region, two such boxes would be placed inside of a third, larger such Descriptive Box. The same observation holds true of Aggregation Boxes.

However, this observation about nested boxes does not hold for mixtures of different types of Location Boxes since placing a less constrained Location Box inside of a more constrained Location Box logically suggests that the constraints of the less constrained box must then be made more constrained, which is probably not what a diagrammer would want. For example, in Figure 65, the painting inside the Verbatim Box could not logically contain regions of Descriptive Boxes because then those Descriptive Boxes would presumably fall under the rules of the encompassing Verbatim Box.

3. Aggregation Boxes

3.1. C Aggregation Boxes

The preliminary letter "C" is a formality that stands for "concrete."

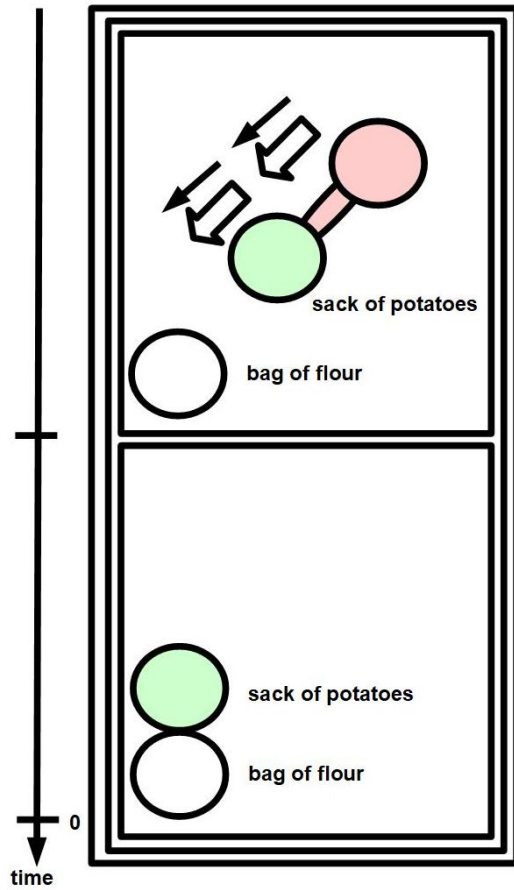


Figure 62: [19] Tumbug for "The sack of potatoes had been placed above the bag of flour."

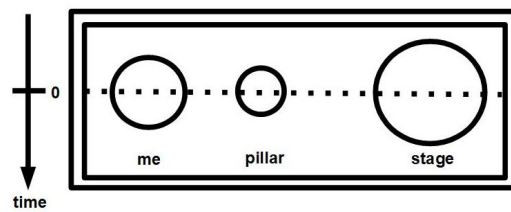


Figure 63: [34] Tumbug for "There is a pillar between me and the stage."



Figure 64: A Descriptive Box has one less border than a Verbatim Box to suggest that a Descriptive Box is less constrained than a Verbatim Box.

(“Corporeal” would be a good alternative synonym, if so desired.) Usually C Aggregation Boxes are called simply “Aggregation Boxes.”

Tumbug has two types of Aggregation Boxes: C Aggregation Boxes and C-A Aggregation Boxes. Tumbug’s icon for an C Aggregation Box is shown in Figure 66. C-A Aggregation Boxes are described in the different section.

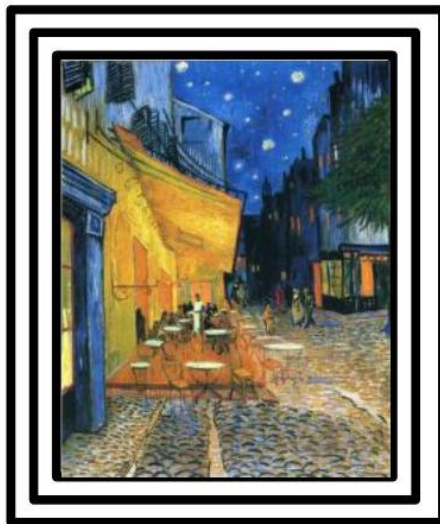
Location Boxes and Aggregation Boxes may be arranged and nested arbitrarily. For example, to show the upstairs and downstairs of a house, two such boxes would logically be stacked. Similarly, to show two regions of a picture where each region could contain only one type of item, but each item could be in any location within that region, two such boxes would be placed inside of a third, larger such box.

Note that the border of any type of Aggregation Box need not be rectangular, though rectangular is the default shape.

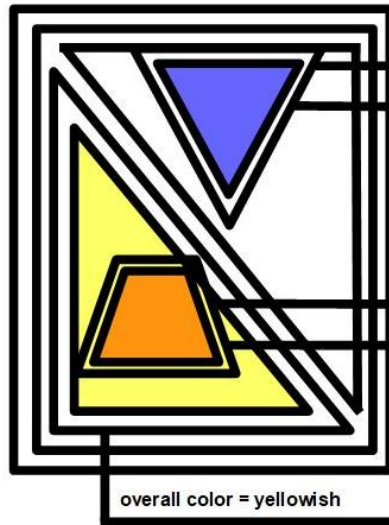
Nearly all representation systems have some means of representing aggregation. Some examples are:

1. In English grammar phrases can be aggregated by parentheses, such as “The flower fields of Encinitas were gradually either replaced (by condominiums or a tourist attraction) or left to fall into neglect.”
2. In algebra, terms can be aggregated by parentheses. For example: $2(x + y) - 3(x + y)$.
3. In human societies, residential regions are organized into communities, which are grouped into cities, which are grouped into states or provinces, which are grouped into countries, and so on.

In Tumbug a (single-bordered) rectangle called a “C Aggregation Box” is used to surround the visual objects that are to be aggregated.



**Verbatim
Box**



overall color = dark bluish

contents = several stars

**Descriptive
Box**

overall color = orangish

contents = several tables with chairs

overall color = yellowish

Figure 65: A Verbatim Box has every feature in a fixed location, but a Descriptive Box has only loosely constrained locations. (Painting: "Café Terrace at Night" (Vincent Van Gogh, 1888))



Figure 66: Tumbug's icon for a C Aggregation Box.

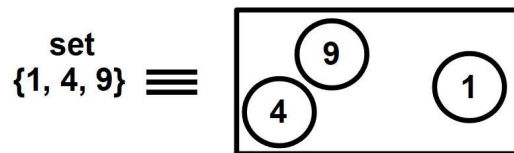


Figure 67: A C Aggregation Box would be used to represent a set since element location is unimportant.

If mathematical sets are being represented in Tumbug, an Aggregation Box would be used since there is a homomorphism between a parenthesized list of numbers in any order and a box of objects in any location, as in Figure 67. For an ordered list, however, a Descriptive Box would probably be used since the relative order (relative places) of the listed objects must be preserved: an isomorphism. See Figure 68.

An Aggregation Box may also have attribute-value pairs, and often do. Some attribute-value information that might be useful to include: the count of the items in the Aggregation Box (e.g., "count = 3"), attribute-values that all the items in the Aggregation Box share (e.g., "shared attribute-value =

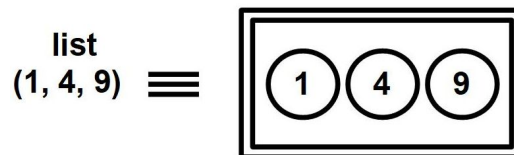


Figure 68: A Descriptive Box would be used to represent a list since element location is important.

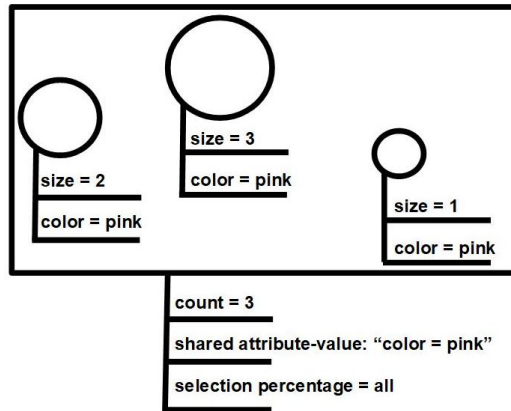


Figure 69: Tumbug for "All the pink souvenirs."

'color = pink)'), the source of the items (e.g., "source = Madonna Inn gift shop"), discounted price if a customer buys everything inside (e.g., "5"), the name of the set (e.g., "things that Fred does routinely"). See Figure 69.

The "shared attribute-value" attribute on any type of Aggregation Box is currently necessary in Tumbug when representing a conjectured set whose members (and therefore count) are not known.

3.2. C-A Aggregation Boxes

Tumbug's icon for a C-A Aggregation Box is shown in Figure 70.

The C-A Aggregation Box is one of the Building Blocks where Tumbug becomes the most profound. "C-A" stands for "concrete-abstract," which is a pair of concepts at the two extreme ends of the abstraction spectrum. A comparison of the C Aggregation Box and the C-A Aggregation Box follows:

- Both have a rectangular boundary that surrounds a set of arbitrarily selected objects.
- Both may have a list of detected attribute-values associated with the represented set.
- Only a C-A Aggregation Box has a second list of forced attribute-values that define the set.
- A C-A Aggregation Box typically contains an unknown number of objects, so an ellipsis is shown inside.

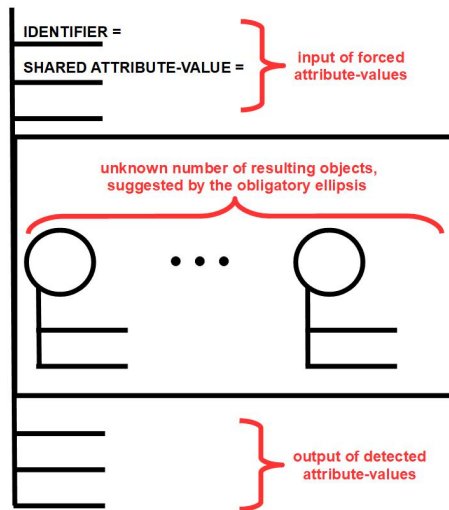


Figure 70: Tumbug's icon for a C-A Aggregation Box.

A C-A Aggregation Box is a deeply abstracted form of a mathematical function, where the input is analogous to an independent variable and the resulting output is analogous to a dependent variable. As in functions, the primary interest is usually in observing the effect on the output after inputting a specific value. For example, if the forced input to a C-A Aggregation Box is the attribute-value pair "species = human," the resulting set would contain all existing humans, and a useful attribute would be the count of all objects (= all humans) in this set. In this way a C-A Aggregation Box is similar to a SQL data base query where query word "WHERE" specifies the attribute-value pair (which in a data base is the column and row, respectively), the aggregate function "COUNT" is a detected (and outputted) property of the set, and the query word "FROM" refers to the set in question. Also, as in math functions and SQL queries, the mapping may not be invertible, such as when the math function is $y = x^2$ for $x = -1$ and $x = +1$, or when the SQL query returns more than one applicable record when queried with "WHERE surname = Smith". Since the resulting matches are typically large and of an unknown count, an ellipsis is used inside a C-A Aggregation Box.

There also exists an analogy of C-A Aggregation Boxes with first order predicate calculus (FOPC). For example, the statement of the classic syllogism, "Every man is mortal," which is represented in FOPC as " $(\forall x) (MAN(x) \rightarrow MORTAL(x))$ ", which could be read as "For all x, if x is an element of the set of humans (MAN), then that implies that x has attribute mortal (MORTAL)," can be represented in Tumbug with the C-A Aggregation Box shown in Figure 71. Note that the "count" attribute of this set (one of the outputs) has unknown value, and its current magnitude would be measured in the billions. Note also that the forced input that defines this set is that "species = human" but that the detected output that describes the resulting set is "mortality = mortal," even though that attribute-value pair was not mentioned in the input.

C-A Aggregation Boxes are needed for mathematical theorems. For example:

"Theorem: In any graph with at least two nodes, there are at least two nodes of the same degree."

Tumbug would illustrate this theorem with a C-A Aggregation Box that contains only one graph of at least two nodes, as shown in Figure 72. The set membership requirement could be represented as the textual description "each graph contains at least two nodes," and the resulting implied output

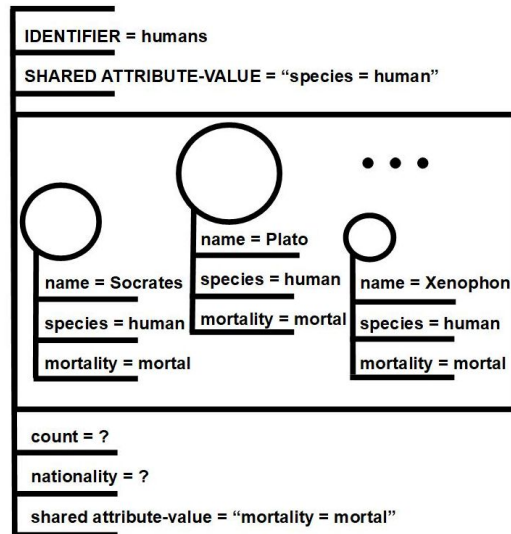


Figure 71: The classic "All men are mortal" syllogism.

could be represented as the textual description "each graph has at least two nodes of the same degree." The upper, forced attributes of the graph could collectively be considered a large "if..." condition, and the lower, resulting attributes of the graph could collectively be considered a large "...then..." result. In constructive theorems that show a concrete example, a visual example would go inside the Aggregation Box, but not all theorems are constructive, so such a diagram is a luxury. This theorem happens to be a non-constructive theorem.

5.2.2 Swirly Arrays

The components of this composite component: Location Box, Cells, optionally any number of 0D Markers up to the number of cells.

An array, also called a matrix, is a well-known mathematical data structure that contains cells in a rectangular structure. If the structure contains only one row or one column then it is called a vector. Information technology has extended the array concept so that the rows of the array need not be the same length (or equivalently, that the embedded "member arrays" need not be the same length) though the rows are usually shown aligned on their left-hand sides. Such arrays are called "jagged arrays" or "ragged arrays."

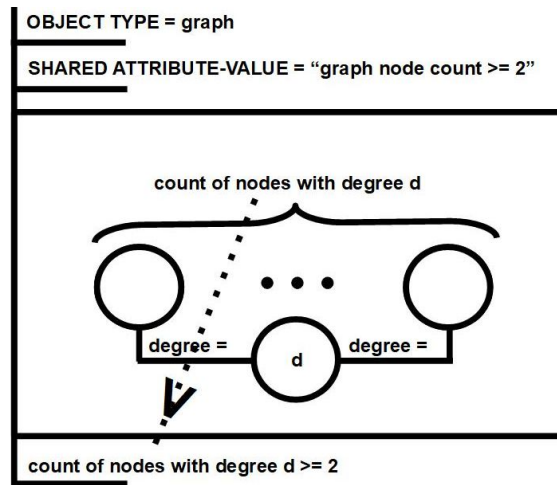


Figure 72: Tumbug representation of a graph theory theorem: "In any graph with at least two nodes, there are at least two nodes of the same degree."

Tumbug extends the concept of a jagged array even further: In Tumbug a "swirly array" is defined as a given spatial arrangement of cells where the cells do not need not to be contiguous or aligned horizontally or vertically. See Figure 73. (Many mathematicians, high school students, and science fair project creators have probably reinvented the concept of swirly arrays many times before, but if so, such documentation is not well-known and not readily available. The author hereby gives advance credit to any such prior inventors and their proposed invention names.) Arrays of any kind in Tumbug context could be called a "composite Building Block."

Swirly arrays as used here are treated more like annotated diagrams than mathematical entities. Using obvious methods, two swirly arrays could easily be added together, and one swirly array could easily be multiplied by a scalar, but special functions such as taking a determinant of a swirly array, or multiplying two swirly arrays together would not be possible in general. However, such structural flexibility in swirly arrays could also be beneficial for customizing arrays for certain special purposes that might need to hold one or more adjunct values not related to the main task at hand, such as storing coefficients of squared terms, which might allow an elegant extension to linear algebra.

The aforementioned array examples use squares for cells because that

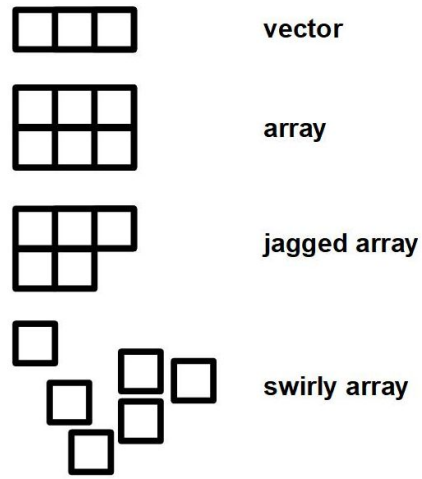


Figure 73: Examples of different types of array arrangements.

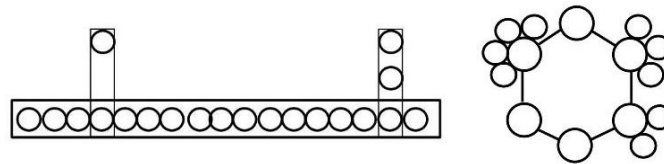


Figure 74: Two examples of Swirly Arrays in Tumbug are the Modal Verbs Icon (left) and Robinson Icon (right).

is standard mathematical practice, but Tumbug uses circles instead, because in Tumbug cells hold concepts or circles, both of which are represented as circles. Swirly arrays are motivated in Tumbug by the need to place detector neurons (or just detector lights) at fixed locations that match irregularly-shaped diagrams of organized concepts, neurological structures, or maps of any kind, for the positional coding of concepts. It is the on-off status of the detectors in combination with their locations that represent more complicated concepts such as emotions or modal verbs. See Figure 74.

5.2.3 XOR Boxes

1. Applicable icon



Figure 75: The Tumbug icon for an XOR Box.

To represent logical OR in Tumbug, a box similar to a C Aggregation Box is needed, a box that holds all the objects that are being OR-ed. However, a C Aggregation Box is not quite specific enough for the concept of OR because a C Aggregation Box is merely a set that does not indicate that only one choice must be selected from the collection of choices in the set. This selection restriction is the concept of mutual inhibition: if all the choices were mutually inhibited neurons, for example, each firing neuron would attempt to suppress all the other neurons for firing, which is what is called a Winner Take All network. To implement a Winner Take All in Tumbug, a variation of a C Aggregation Box is needed, a variation that is called an XOR Box, the Tumbug icon of which is shown in Figure 75. The black dots in the corners are intended to be suggestive of the typical link diagrams used to represent mutual inhibition in artificial neural networks.

2. Inclusive OR

It should not be surprising that the most basic logic operations (AND, OR, NOT, XOR, NAND, NOR, etc.) should be present in Tumbug in some manner. Some traditional methods of representing the (inclusive) OR operation in computer science are shown in Figure 76.

Although the Venn diagram representation is an appealing candidate for Tumbug, it has two practical problems: (1) It normally shows both possibilities in detail, which is spatially prohibitive in Tumbug unless variables are used; (2) The intuitive visual logic of the real world is the opposite of that shown in Venn diagrams.

The latter problem is the most important and the most interesting. In particular when the existence of real-world objects is considered, "AND" is intuitively understood to mean that all the listed objects are present. For example, the statement "nuts and berries are present" implies a visual depiction of nuts existing in the vicinity of berries, whereas the statement "nuts

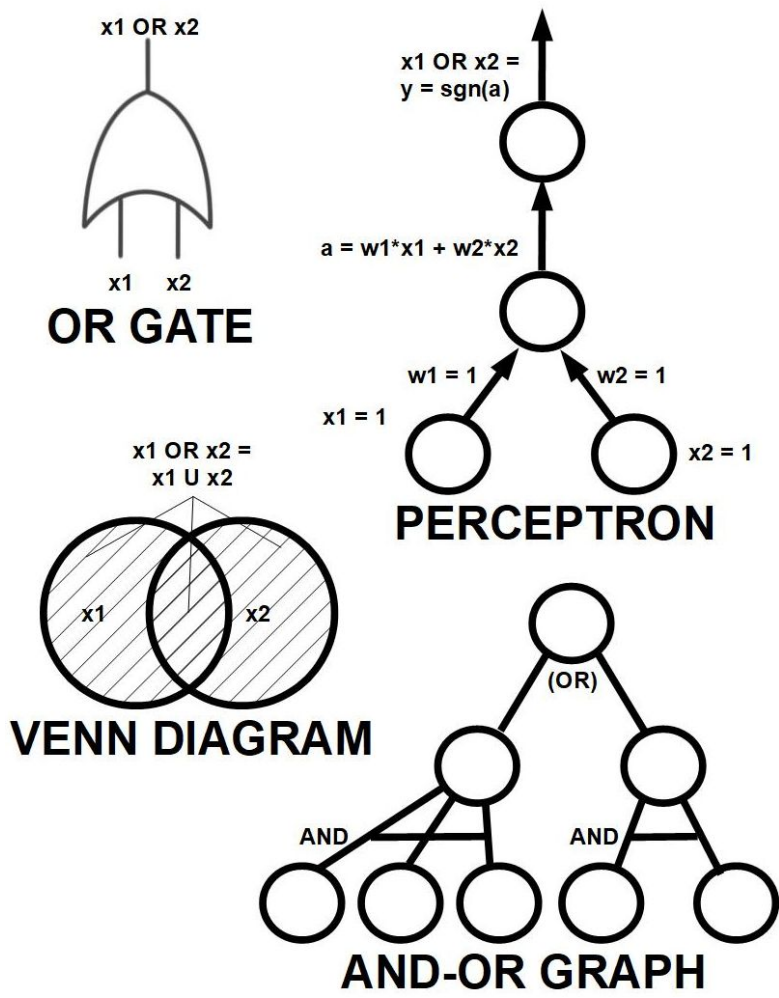


Figure 76: Some traditional visual representations of logical "OR."

or berries are present” (the ”or” of which is an inclusive OR, by default) implies a visual depiction of nuts existing or berries existing, or possibly both. A Venn diagram fails at representing this situation since a Venn diagram shows an OR (= union) of two sets by showing both sets at the same time, which is intuitively wrong, and shows an AND (= intersection) of two sets by a small region of intersection, which in real life would be interpreted as showing the presence of only objects that are categorized as both nuts and berries, which is also intuitively wrong. The conclusion: Intuitive, visual ”OR” cannot be easily and intuitively represented by a Venn diagram, therefore intuitive, visual ”OR” is intuitively an abstract concept that requires some other representation. Therefore Tumbug represents visual ”OR” using the perceptron method. (A perceptron is a type of artificial neural network, therefore Tumbug could be said to use neural networks, to some extent.)

The Tumbug solution for the representation of OR relies on another Tumbug Building Block discussed elsewhere: the Correlation Box. One main difference is that whereas the Correlation Box examples shown earlier represented a function of a single variable, such as $A = f(B)$, OR requires a function of two variables, such as $A = f(B, C)$. Specifically, the function would be named $A = \text{OR}(B, C)$. Correlation Boxes can be generalized to any number of input variables, as shown in Figure 77.

Figure 78 shows the intermediate Tumbug representation of OR, one diagram for inclusive OR (= IOR), and one diagram for exclusive OR (= XOR). Note that the use of a C Aggregation Box functions as a set of possible choices. For example, $\text{IOR}(A, B)$ produces the set $\{A, B, A \text{ AND } B\}$.

The comparison of IOR and XOR is shown again in Figure 79, using XOR Boxes.

3. The existence attribute

An attribute that is important enough to address in its own section is the attribute of existence. In mathematical literature, for example, the ”there exists” symbol (\exists) is extremely common, so to render mathematical theorems in Tumbug, one would need to know how to represent ”there exists.”

The Tumbug solution is simple, but unfortunately the solution not satisfyingly visual: the current solution is to use a typical Attribute Line on a typical C Object Circle (or C Aggregation Box) whose existence is to be made explicit, and to place the attribute name ”existence =” followed by a value from the range 0 through 1. The 0-1 range describes the range of likelihood from ”does not exist” (0) through ”exists” (1). See Figure 80. The main visual drawback of this representation is that existence is such a funda-

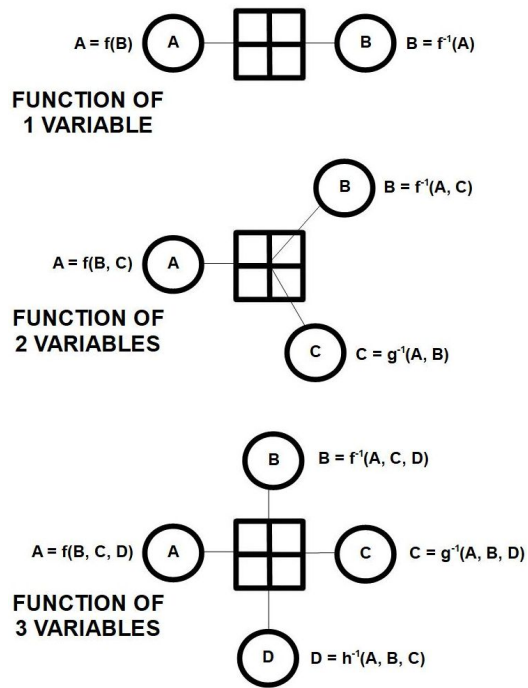


Figure 77: Correlation Boxes can be used with any number of input variables, not just one.

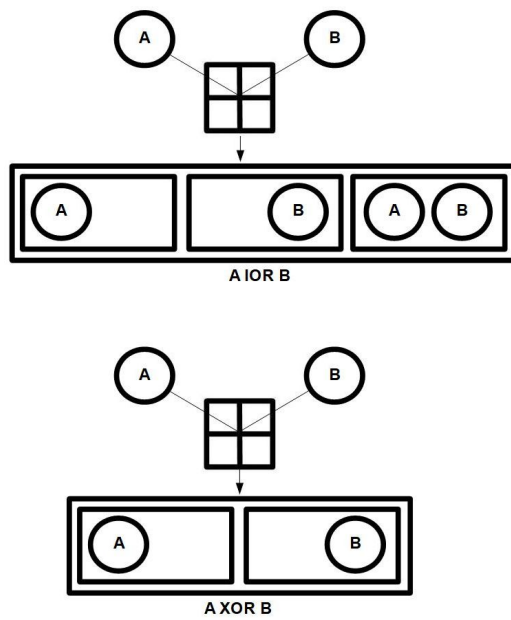


Figure 78: Tumbug representation of inclusive OR (= IOR) versus exclusive OR (= XOR).

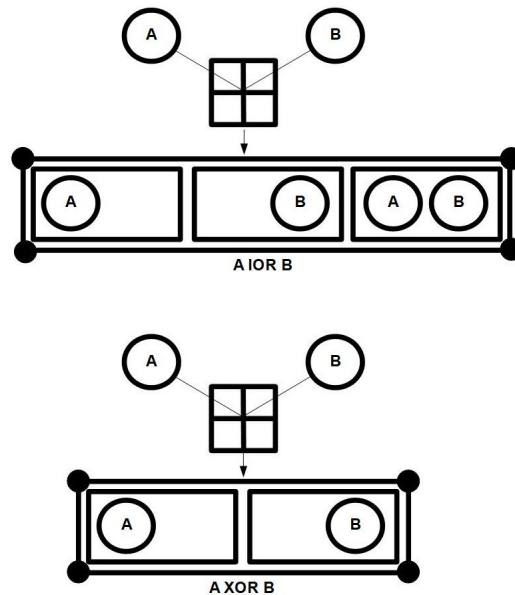


Figure 79: Comparison of IOR and XOR functions, using XOR Boxes.

mental attribute of real-world objects that one would expect that attribute of an object to be overwhelmingly obvious in a diagram, but in Tumbug it is not.

The reason this non-visual, attribute-value type of solution seems to be necessary is that an object depicted in a Tumbug diagram is ordinarily assumed to exist, therefore to show that an object does not exist would suggest that mere omission of the depicted object would represent non-existence. However, that solution in turn implies that any object not shown must not exist, which in turn implies that a Tumbug diagram must depict everything in the universe. Although one could adopt the convention that anything not drawn on a Tumbug diagram is merely a "don't care" rather than a "does not exist," by the Tumbug convention of wildcards (discussed elsewhere in this document), a blank already means "don't care," when then leaves no way to uniquely represent "does not exist." This uncomfortable and counter-intuitive situation is a variation of the "closed-world assumption" of formal logic, where anything not explicitly marked "true" must be "false."

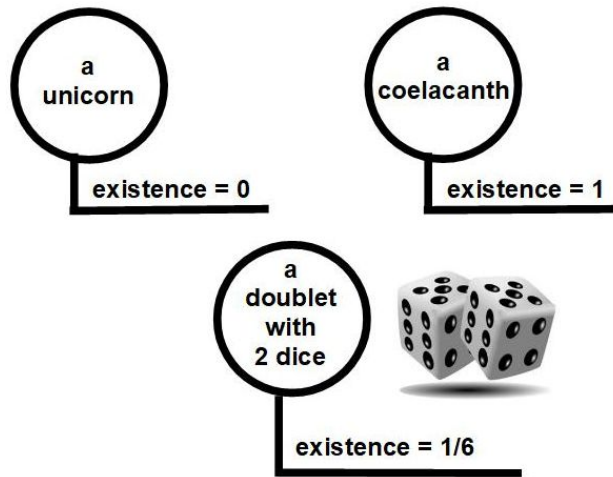


Figure 80: Existence in Tumbug is considered merely another attribute whose value can vary.

6 Attribute-like Building Blocks of Tumbug (A)

6.1 Attribute Lines

In Tumbug, attributes are almost always shown together with values, and vice versa, separated by an equal sign “=”. An attribute followed by its value, such as “speed = quick,” are written together on the same horizontal line. Figure 81 shows the basic line that holds text, Figure 82 shows how such lines are attached to a C Object Circle, Figure 83 and Figure 84 shows generalizations of this type of structure, and Figure 85 shows an example of a specific object with specific attributes. There is no theoretical limit on how many Attribute Lines may be attached to any type of Object Circle. In a physical implementation of Tumbug, the equals sign “=” would not exist since the equals sign merely distinguishes between two types of stored concepts (viz., attributes and values), and in a physical implementation there would presumably be two separate memory locations that hold those concepts, and the separation of these memories would be already clear.

Some Attribute Lines terminate at icons or small diagrams when the attribute values are not simple textual or numerical descriptions. This happens



Figure 81: Tumbug's icon for an Attribute Line.

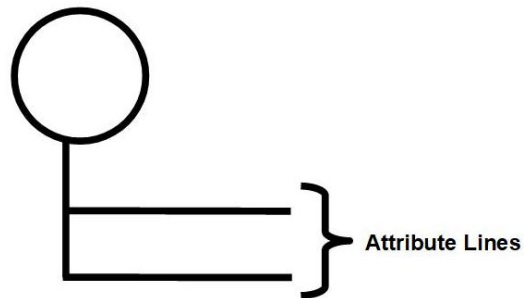


Figure 82: Attribute Lines for a given object branch off from a single stem, and can be of any quantity.

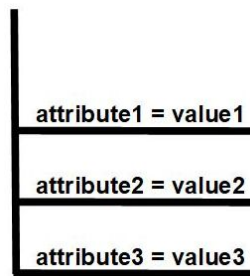


Figure 83: Three AV pairs alone.

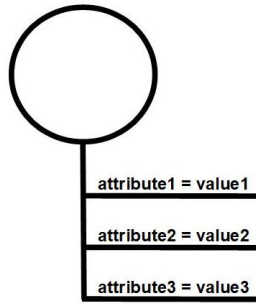


Figure 84: Three AV pairs attached to an object, which is the usual situation.

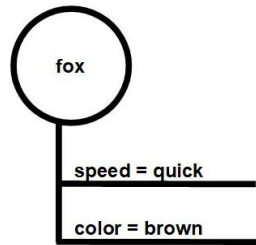


Figure 85: Tumbug for "quick, brown fox."

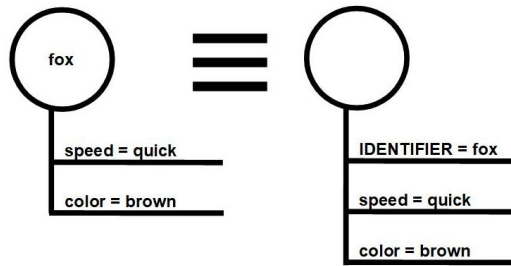


Figure 86: These two Tumbug representations for "quick, brown fox" are equivalent. Currently both are acceptable.

with State Diagrams and Motivation Triangles.

As for the values to which attributes are set, in typical, partial-text Tumbug, numbers are given exactly as in their standard math counterparts, such as "3." Tumbug ultimately was intended to be a fully visual representation system, however, so such numbers can also be given as lengths of a special measurement object whose icon is called a Value Bar, described elsewhere in this document.

This article's convention: Tumbug attribute-value pairs are almost always connected to the object to which they are associated, rarely isolated. If there exists more than one attribute shown per object, a single stem is usually used, unlabeled, with all the attribute-value pairs branching off that stem. The stem can be shown emerging from the bottom, top, or (less frequently) at another arbitrary point on the C Object Circle's perimeter.

Tumbug minor convention: It is not always necessary to have an attribute and value paired, although this is usually the situation. For example, "life status = alive" could be abbreviated to merely "alive" if the meaning is clear enough.

On C Object Circles shown in this document, the main label on the circle, such as "fox" or "Susan" or "they," is technically an attribute from the set of Attribute Lines for that object. That main label could be considered to be the value assigned to the first attribute in the list, an attribute called "IDENTIFIER," such as in the attribute-value "IDENTIFIER = fox." See Figure 86. Such a single, enlarged Label String on the C Object Circle itself is intended only to make the diagram easier to read and understand at a glance.



Figure 87: Typical forms of Tumbug's icon for a Value Bar.

7 Value-like Building Blocks of Tumbug (V)

7.1 Value Bars

7.1.1 General

The possibility of using an additional Building Block only to represent magnitude was alluded to, elsewhere in this document. Tumbug's icon for such a Building Block is called a Value Bar, shown in Figure 87. Since numerical values are so terse to write as digits, there is currently little motivation to use Value Bars unless possibly to demonstrate that Tumbug can be made entirely visual, as the Tumbug acronym suggests. Also, if Tumbug were to be implemented in an artificial neural network in hardware form, digits (and text) would be too difficult to represent unless they were visual, which would be another justification for use of Value Bars.

The length of the shaded segment along the bar (= thin rectangle) is primarily what is most important in this icon. The Tumbug icon for a Value Bar is a bar, typically with an arrow at one end, but the arrow is not required when the value is obviously limited to a finite range. The two extreme values within the given range are emphasized by a vertical hatch mark at each end of the bar, as if to show where a ruler or tape measure would be laid while measuring the bar end-to-end. The hatch mark at the arrow tip means infinite magnitude of the associated attribute's value.

Side conjecture: It may be possible that each specimen of a higher animal, especially if human, uses its own individual height as the primary unit of height measurement in the real world. Most likely the ratio between heights

is subconsciously computed as a convenient estimate for the likelihood of winning a fight against another animal. For example, if the self-to-other height ratio were ≥ 1.25 , meaning the self is 1.25 times taller than the other animal, this could automatically trigger the psychological impression and attitude in the self that "I can take him."

Side conjecture: The above conjecture about height also holds true for other physical, emotional, and psychological measures of the self, in relation to others, such as weight, speed, aggression level, empathy level, intelligence, and knowledge.

Values of motion, such as speed, acceleration, and direction, can be represented by Tumbug in nearly the same way as length. Compare Figure 88 for distance to Figure 89 for speed. In general, this magnitude coding scheme is of great benefit for neural network implementation since a single hardware mechanism can represent any of the Building Blocks of Tumbug, whether numerical or visual—length, circumference, angle, area, volume, curvature, speed, acceleration, duration, age, weight, force, energy, power, friction, temperature, color, texture, volume, range, precision, accuracy, probability—and even human-based, abstract attributes such as price, stock keeping unit (SKU), attractiveness, popularity, sentimental value, and cuteness.

As for neurobiological verisimilitude of Tumbug, various biological visual systems are known to handle motion information using specialized hardware, which implies certain neurons fire in response to certain types of motion or certain values of motion, which would then map fairly directly to Tumbug's Motion Arrows with appropriate attribute values of speed, acceleration, collective looming motion, and so on, filled in. For example, detection of the movement of images is one of the four primary visual operations in a frog's eye (Lettvin et al. 1959, p. 257), a fly's visual system determines if the visual field is looming sufficiently fast for the fly to contemplate landing (Marr 1982, p. 35), speed gradients are computed in middle temporal/V5 area of the macaque extrastriate cortex (Orban 2008), and the human visual system uses at least two motion detection systems, one of which is not even aware of the object whose motion is being automatically detected (Chubb 1995, p. 109).

7.1.2 Ranges

Tumbug represents ranges as shown in Figure 90.

Value Bars can also represent ranges. Each equivalent of a unit square on









TWO OBJECTS, AS A DIAGRAM:	DISTANCE BETWEEN THEM, AS A NUMBER:	DISTANCE BETWEEN THEM, AS A DIAGRAM:
	0	
	1	
	2	
	3	

Figure 88: Tumbug can represent distances as lengths, if desired.









ONE MOVING OBJECT, AS A DIAGRAM:	SPEED, AS A NUMBER:	SPEED, AS A DIAGRAM:
	0	
	1	
	2	
	3	

Figure 89: Motion attributes such as speed are almost as simple to represent as static attributes such as distance.

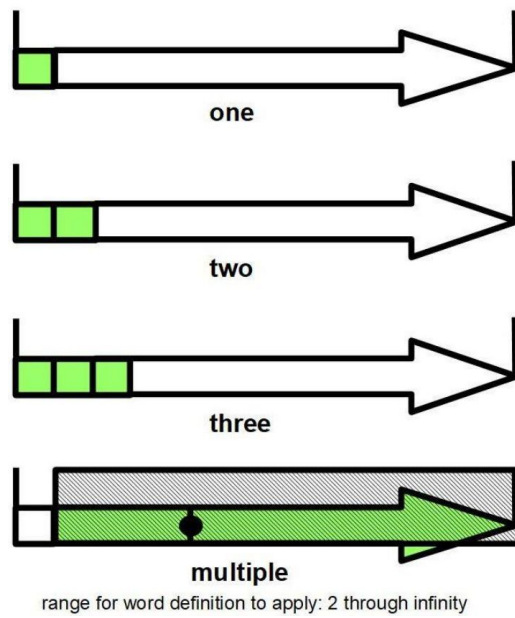


Figure 90: The word "multiple" applies only if the observed value falls within the hatched range of ≥ 2 .

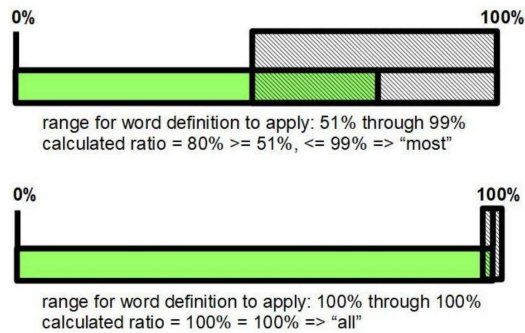


Figure 91: The words "most" and "all" apply only if the calculated ratio falls within the hatched range.

a Value Bar can represent a span of one unit (\Rightarrow a count of 1) in the horizontal direction, therefore such squares placed side-by-side into a horizontal bar can represent a specific integer (e.g., a count of 3). In Figure 90, the shaded region makes the count easier to see at a glance because a 2D area is easier to see than a 1D hatch mark on a number line, and the green color can represent known (i.e., non-fuzzy) information. A 45-degree hatched pattern—the hatched pattern used by 2D Markers—could logically be used instead of the green color, but that scheme was not used in this document.

A semi-ambiguous quantity like the concept conveyed by the word "multiple" (meaning more than one) typically means an arbitrary point within a range, so both the point and range must be represented somehow. If the specified range can be represented by a different shading, such as the purple shading in Figure 90, then a small ball can represent an arbitrary point within that range: the location of the ball becomes a type of Wildcard. The ball icon suggests that the ball is free to roll to anywhere within the range, but at any given time the ball must occupy a specific point within the range.

In the aforementioned examples, fixed values (viz., 0, 1, 2, 3) were used. By using ratios instead, especially 0% through 100%, relative concepts such as "most," "all," "few," and "many" can be defined with Value Bars, as shown in Figure 91 and Figure 92. Some of these examples and values as shown do not exactly match human intuition about the meanings of these words, but all come close, and can be modified as desired.

The delimiting hatch marks are important in this icon because they allow quick estimation of the ratio of shaded bar length to the full bar length. Such

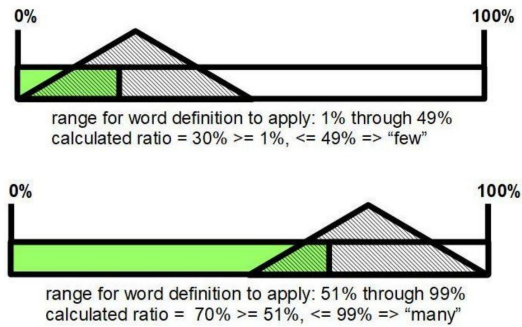


Figure 92: The words "few" and "many" apply only if the calculated ratio falls within the hatched range.

ratios are needed to give visual estimates to language terms such as "few," "several," and "many," since such terms conceptually are fuzzy-bordered regions whose lengths and locations are a fuzzy percentage of the entire region. The hatched triangles in the diagram represent triangular, fuzzy logic membership functions.

7.1.3 Example: Goals

Such value representations can be combined in various, useful ways. For example, a certain range of distances can be given a fuzzy textual description such as "far," and conjunctions of such ranges from different measurement types can be a given fuzzy textual description such as "far" and "fast." In turn, these conjunctions of ranges can become useful for expressing common motivation, and therefore common goals, as shown in Figure 93, Figure 94, Figure 95, and Figure 96. The ability to represent goals in general is extremely important in artificial intelligence since "goal-directed" behavior is often cited as a component of the definition of intelligence itself (e.g., Minsky 1986, p. 22; Kurzweil 1990, p. 18; Kurzweil 1999, p. 73).

These nuances of which goals might be important to a person come into play in language translation, discussed later in this document.

7.1.4 Geometrical variations

Value Bars imply the existence of several interesting side topics that have not been addressed here. For example, in many cases a geometrical structure

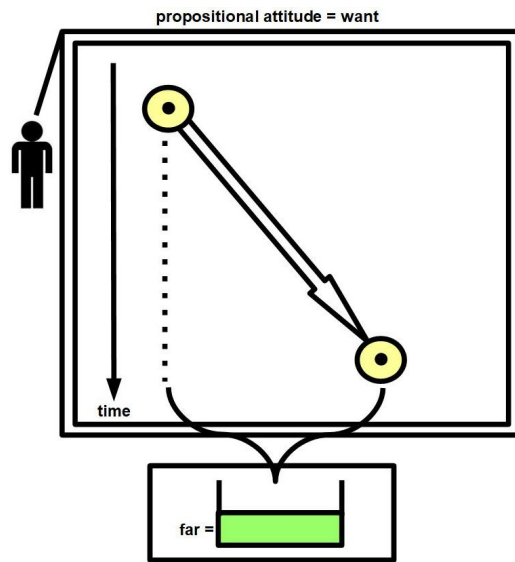


Figure 93: "I want to get that (yellow) thing away from me, as far as possible. (I don't care where or how fast.)"

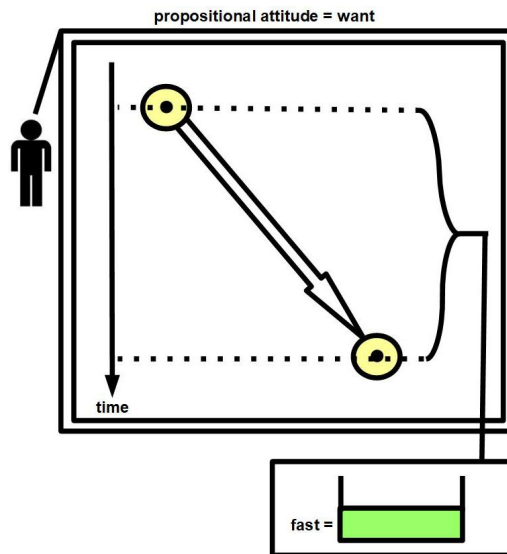


Figure 94: "I want to get that (yellow) thing away from me, as fast as possible. (I don't care where or how far.)"

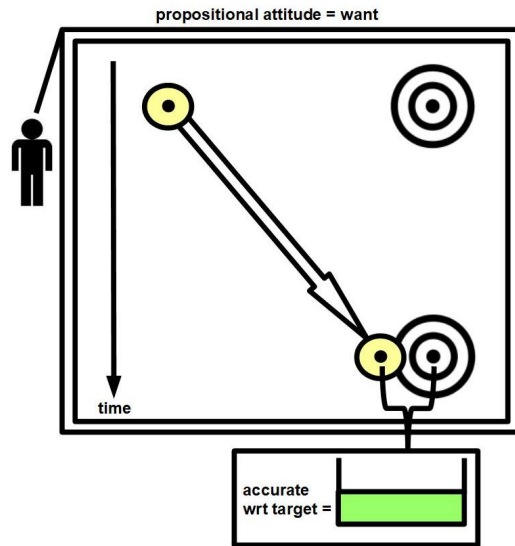


Figure 95: "I want to get that (yellow) thing on target, as accurately as possible. (I don't care how far or how fast.)"

other than a rectangle is preferable, such as when angles are being measured, whereupon a circle would be more suitable. Similarly, many phenomena that have a wide numerical span are probably mentally understood via exponential scales, or via hyperbolic termination, such as representation of infinity by an asymptote's location.

7.2 Wildcards (further development anticipated)

Because Tumbug is a visual representation, the usual type checking of digital programs is made much more difficult with Tumbug. In particular, when Tumbug represents an OAVC system, a person would ordinarily expect the value (V) to be numerical, as in Figure 97. However, Tumbug is so general that it needs wildcards, which include "values" like "Don't Know," "Don't Care," or "Not Applicable," as in Figure 98. Such strings are few enough in quantity that they could be implemented with special strings, as in the "NaN" (= Not a Number) special value implemented in the programming language Java (e.g., Gosling et al. 1996, p. 35), but Tumbug is more complicated still. For example, a noisy numerical value would need multiple pieces

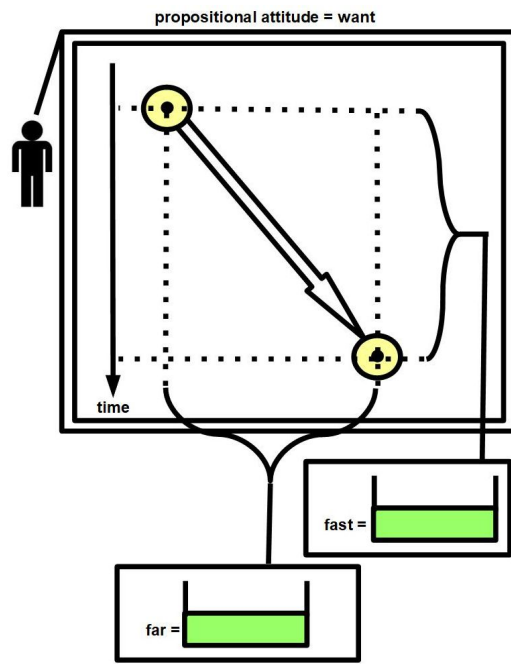


Figure 96: "I want to get that (yellow) thing away from me, as far as possible, as fast as possible, I don't care where."

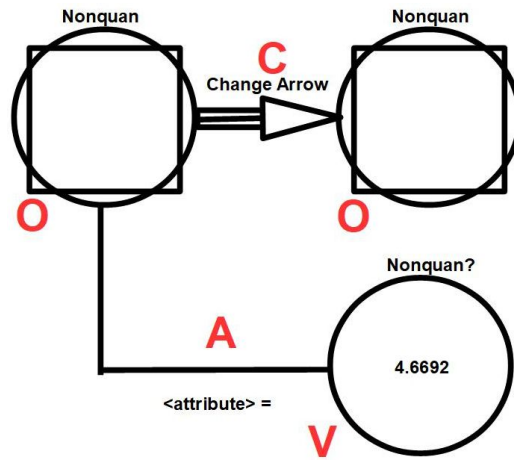


Figure 97: In this Tumbug representation of an OAVC system, the value (V) is numerical.

of information: (1) the value, (2) the probability density function (pdf) (e.g., Gaussian) that is associated with that value, (3) the mean of the pdf, (4) the variance of the pdf. This situation cannot even be represented with a four-element vector since the pdf is a function, not a numerical value.

The wildcards currently of interest in Tumbug are the following:

- * = any value, matches even 0 values
- ? = any value, matches even 0 or 1 values
- + = any value, matches at least 1 value
- DK = don't know
- DC (or blank) = don't care
- DNE = does not exist
- allowed range

A very useful wildcard is the asterisk "*", which is often used to mean "any value." In computer science, regular expressions routinely use this asterisk wildcard "*", as well as the two other wildcards "+" and "?". To generalize this situation, ideally a wildcard should be able to represent any

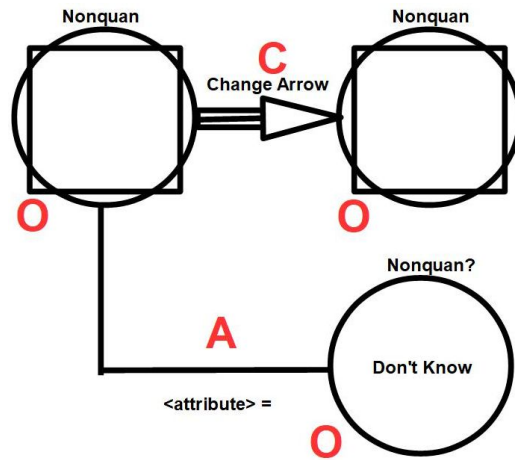


Figure 98: In this Tumbug representation of an OAVC system, the "value" is a string, which is closer to the concept of an object (O).

type of range description, which suggests use of a number line of real numbers, or even a 2D region or higher of real numbers. Figure 99 shows two simple Wildcards. Theoretically it seems that the "V" (= value) slot in a Tumbug diagram should allow any object in order to be completely general, though the implications of such a decision have not been investigated in this study. However, this is the same design decision made in Allen Newell's architecture Soar (Newell 1990, p. 169): "Both the attributes and values may be other objects, so that arbitrary attribute-value structures can occur." This situation is not too different from situations that humans encounter, such as "His final grade was the average of his midterm score and final exam score, where each letter grade is assigned to each 10% section of the top of the spectrum." Effectively such a description of a value is equivalent to a deferred answer that requires data to be gathered and a function to be applied to that data before the resulting value can be known.

The icons in the figure are intended to suggest that the small ball inside the smooth tube can roll to any position within the tube, especially unpredictably. This pictorially represents that the ball can exist at any point within the range of values within the tube. More commonly this range has values marked on it, as in Figure 100.

The region in which the ball can roll can be easily extended to 2D, 3D,

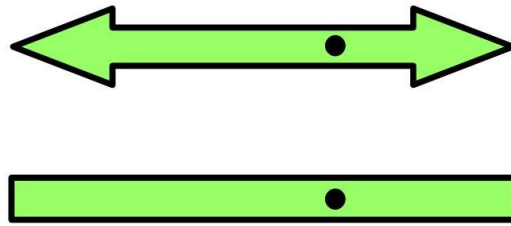


Figure 99: Tumbug's icon for a Wildcard. Top: Where any real value is possible. Bottom: Where only a limited range is possible.

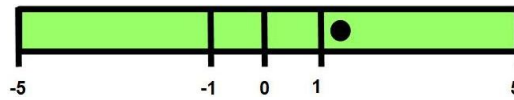


Figure 100: A Wildcard that is limited to a point within the continuous range of values from -5 through 5.

or higher. Figure 101 shows a 2D wildcard region.

As an example, consider a specific attribute value x . There exist a few common situations where the visual value of x may be difficult to plot:

1. **Problem:** x has no magnitude.
One solution: Plot a point at $x = 0$.
Justification: Simple and obvious.
2. **Problem:** x has no value, not even zero, because x does not exist.
One solution: Do not plot any point. Conceptually $x = \text{nil}$, the same

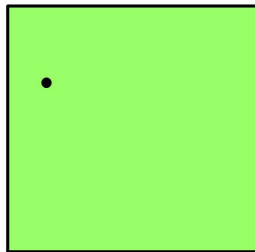


Figure 101: A Tumbug 2D Wildcard that is a square region.

value given to new pointers.

Justification: Any plot of x at a given point means x has that value, so there should be no plot.

3. **Problem:** x is unknown, although the user does not care about its value.

One solution: Pick a random value for x , and plot that value.

Justification: Since the user does not care which value x has, any value will work fine.

4. **Problem:** x is unknown, and the user does care about its value.

One solution: Plot all possible values, and apply probability to the ranges.

Justification: The answer does get plotted, and the probability becomes a caveat for acceptance.

Each of these cases and its suggested visual equivalent is shown in Figure 102.

The described solution in the figure works acceptably for attributes where x is simply a 1D magnitude, but the problem is more severe when x is a multivalued attribute like color (which is 3D) or shape (which is ∞ D), not because of the multiple dimensions but because in practice such an object cannot even be depicted since any depiction must be rendered with some color and some shape. There exist workarounds, but the workarounds are less intuitive for problematic attributes like color.

The aforementioned situations have an interesting implication: Numerical values from a human perspective have more attributes than mere value. These attributes are largely independent and consist at least of:

- magnitude
- existence
- range
- importance

The zero case is ordinarily clear, though for clarity a KRM should not default to value = zero only because it is convenient, since zero is often a

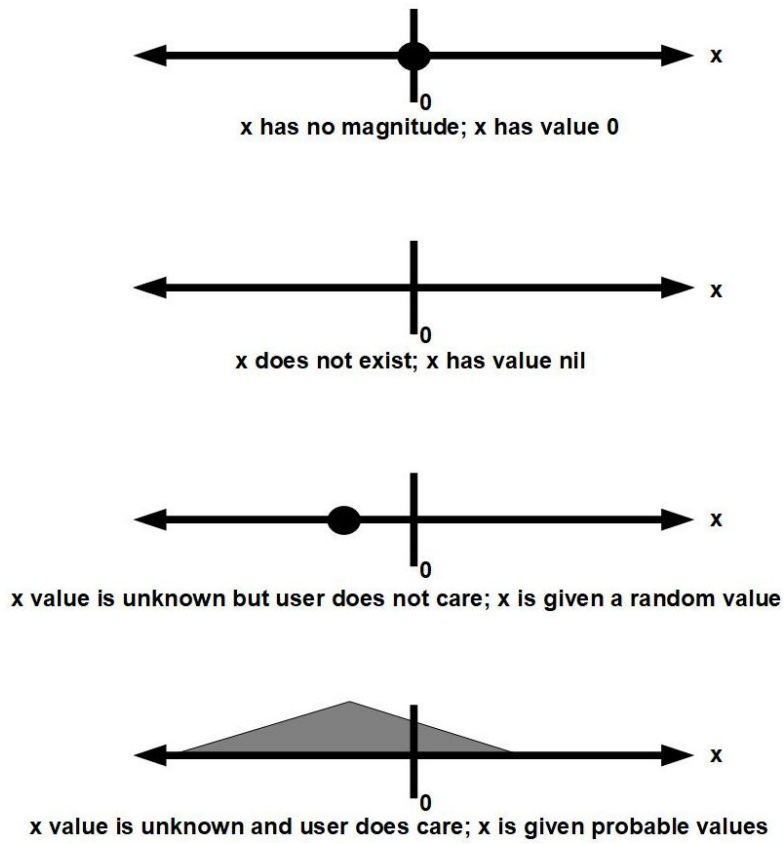


Figure 102: For the attribute of magnitude, good visual representations of Wildcards exist.

mathematically problematic point and the KRM display should be as unexpected as tolerable for visualizing the system In Tumbug there exists at least three Wildcard values:

- don't know ("DK" symbol)
- don't care ("-") symbol)
- does not exist ("nil" symbol)

In the sets of four cases diagrammed above, these cases combined as:

1. does not exist ("nil" symbol)
2. unknown but user does not care: "DK/DNC"
3. unknown and user does care: "DK/C"

These three wildcards can be generalized into a single diagram, as in Figure 103. Although humans often tend to think of "exists" as a binary condition "exists versus does not exist," in physics such concepts do not strictly hold. For example, the question "An electron exists within this atom of iron" may not have a clear-cut answer if that atom of iron has formed a metallic bond with another iron atom. Similarly, humans tend to think of "know" as the binary condition "know versus don't know," in real life such concepts do not strictly hold, since likelihood of a fact falls within a spectrum from "absolutely not" through "absolutely sure." For example, the question "Can your son solve a random Rubik cube position within five minutes of seeing a Rubik cube for the first time?" evokes the commonsense answer "no," but nobody knows for certain.

Existence is independent from knowledge. For example, the question "Are coelacanths endangered?" involves an existent animal but an unknown attribute value, in contrast to the question "What color are unicorns?" involves a non-existent animal with a consensus attribute value. See Figure 104. Similar situations arise even in mathematics, such as in induction proofs where a base case has not been established. For example, if $n = n + 1$ for all positive integers n , then it is true that for a specific $n = k$ the formula will also be true for $n = k + 1$. However, no base case can ever be found whereby $k = k + 1$, therefore this "fact" becomes known in a formula that does not legitimately exist.

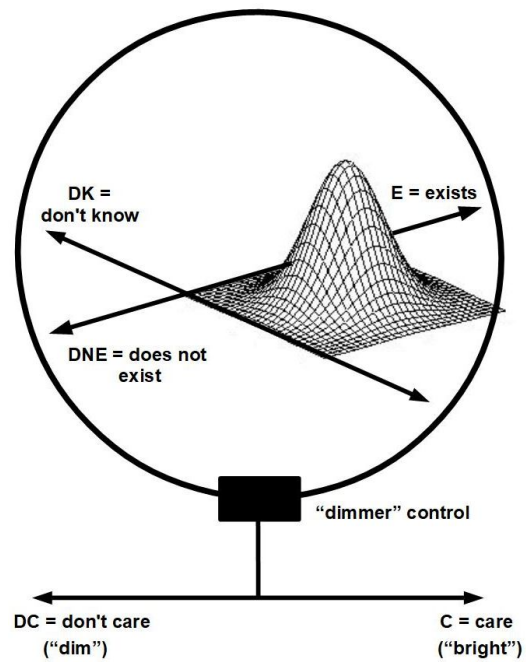


Figure 103: The three wildcards of "exist," "know," and "care" are independent and can be generalized and integrated into a 3D plot. Each of the three wildcards has an associated spectrum, even if a spectrum for such a concept is rarely used. (Source: unknown.)

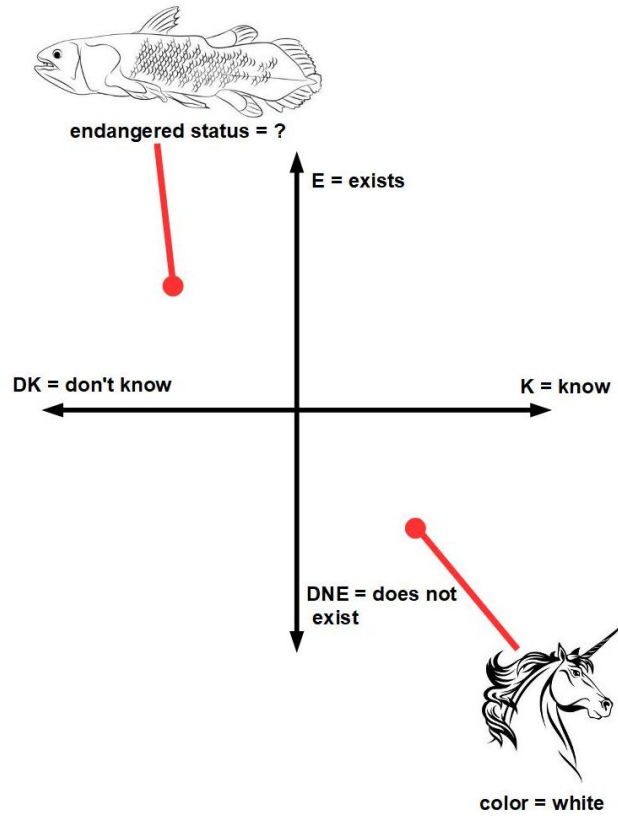


Figure 104: Existence and knowledge are independent; people might not know facts about existent entities, yet might know facts about nonexistent entities. (Source: Drawing Tutorials 101.)

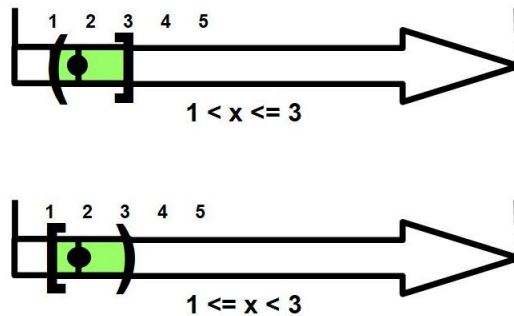


Figure 105: Range Caps are used on Value Bars to mark off a range and to show inclusion/exclusion of the end point.

7.3 Range Caps

Figure 105 shows Range Caps.

Range Caps are just the four familiar keyboard characters “(,)”, “[,]” and “]” as used in math to mark the ends of 1D range on a number line. The brackets “[” and “]” mean the end point is included, the parentheses “(” and “)” mean the end point is excluded.

8 Change-like Building Blocks of Tumbug (C)

8.1 Single Time Arrows

Most commonly when Tumbug needs a Time Arrow, it uses a Single Time Arrow. However, Split Time Arrows also exist to represent multiple possibilities along the timeline. See the section on that topic for more details.

Tumbug’s most common icon for a Time Arrow is shown in Figure 106. Very often a hatch mark representing the current time is also included, and is labeled “0” to mean “now” (Figure 107).

It should be clear that the concept of time is common in the WS150 problems.

The Tumbug icon used to represent time is a single-line arrow, usually pointing downward on the page, labeled with the word “time” or “t”, and running down the left-hand side of the diagram as shown in Figure 108. Any scenario where time is important to represent (e.g., for grammatical tense or



Figure 106: Tumbug's icon for a Time Arrow.

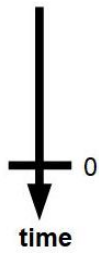


Figure 107: Typically a Time Arrow has a hatch mark labeled zero ("0") to mark the current time (= "now").

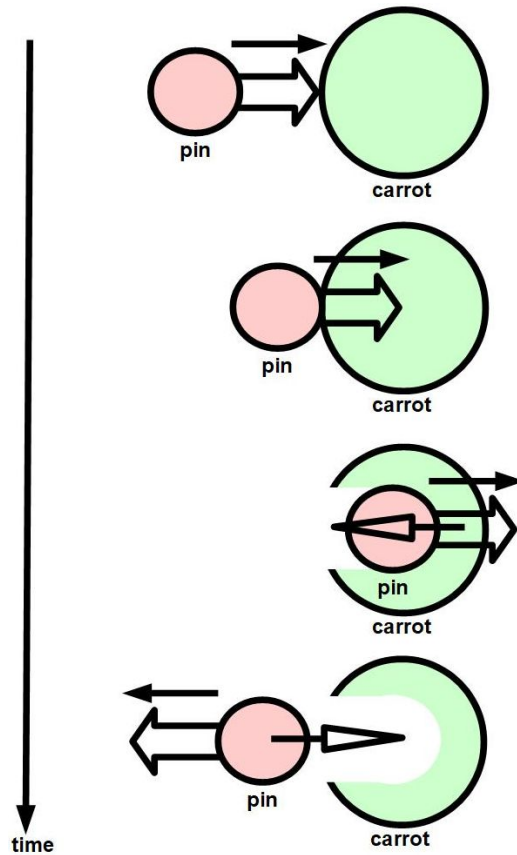


Figure 108: [104] Tumbug for "A pin stuck into a carrot will leave behind a hole when the pin is withdrawn."

for a temporal list of expected actions) must have this arrow placed on the associated diagram, typically on the left.

1. Time without space, WS150 example: #104 (carrot)

Although most WS150 problem statements involve time, time need not be involved in every statement. This is particularly true of general rules, such as the rule in Figure 108.

This example is derived from a portion of WS150 question #104.

"[104] I stuck a pin through a carrot. When I pulled the pin out, it left a hole. What left a hole? POSSIBLE ANSWERS: {the pin, the carrot}"

2. Generalization: space-with-time

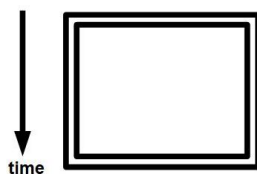


Figure 109: The Time Arrow icon and some type of Location Box icon are commonly used together when moving objects are described.

Tumbug combines the concepts of space and time into a single concept called "space-with-time." This combination is motivated by the following observations:

- Language descriptions of objects moving in space are extremely common.
- Space and time are already combined into the single concept "space-time" in physics.
- Time is sometimes spatialized as an extra dimension of space when training neural networks. (E.g., Simpson 1990, p. 326.)

Not surprisingly, then, the Location Box icon (for space) and Time Arrow icon (for time) are often used together, as shown in Figure 109.

3. Discretized time

Not all systems use continuous time. Some examples of systems that use discretized time are: board games (since each person takes their turn), sequential marquee lights on a theater sign, frames on a film strip for a motion picture camera, displayed minutes on a digital watch, instructions executing during each computer processor clock cycle, cellular automata (since nothing happens between transitions), and steps of a computer simulation.

Tumbug can represent discrete time increments easily by stacking either C Aggregation Boxes or Verbatim Boxes along a Time Arrow (preferably touching the Time Arrow to make the meaning clear). In such a representation, each C Aggregation Box or Verbatim Box means an aggregated section of time, rather than an aggregated section of space. In Figure 110, the first three steps of a simulation of a glider configuration from Conway's Game of Life (which is a cellular automaton) is shown, using Location Boxes since the spatial configuration at each step is important.

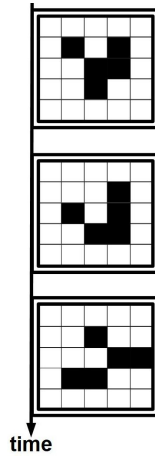


Figure 110: Time can be discretized with Location Boxes, with each Location Box holding one time increment of the process. This diagram shows three consecutive steps of the evolution of a glider from Conway’s Game of Life, which is an example of a specific cellular automaton.



Figure 111: Tumbug’s icon for a Motion Arrow.

8.2 Motion Arrows

From the outset, one design goal of Tumbug was to regard moving objects (in contrast to static objects) to be the default situation in the real world so that Tumbug’s KRM design would not need to struggle later to add time to a fundamentally static KRM. Too many computer science inventions such as data bases and artificial neural networks have encountered this problem of design short-sightedness that required later augmentation with time.

8.2.1 Motion of objects

In Tumbug, solid arrows are used to represent motion of an object (Figure 111, therefore Motion Arrows are usually used only in conjunction with

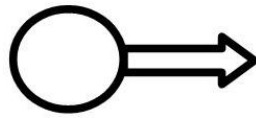


Figure 112: Tumbug diagram of a physical object in motion using an Object Circle with a Motion Arrow.

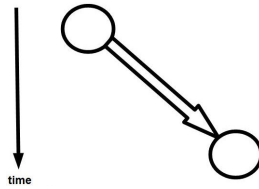


Figure 113: If duration of a moving object is important, then this diagram type should be used.

an object See Figure 112. If the direction of motion is important, this implies that space is important, whereupon the arrow should be placed inside a Location Box. Motion Arrows are allowed to point in any direction, whether inside or inside of a Location Box.

If specific points in time or durations of time are important while an object while it is moving, then a Time Arrow should be used as shown in Figure 113 and Figure 114.

Should data objects use the same Motion Arrow as physical objects? The current Tumbug design decision is "yes," and the reasoning for this deci-

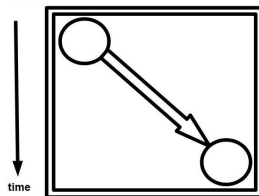


Figure 114: If relative duration and relative distance of a moving object is important, then this diagram type should be used since space and time are important in such a situation, though not exact values.

sion requires some thought. First, information always requires some physical medium to exist, and on which the information can be transferred. Second, this physical medium may be either: (1) discrete storage units that can be transported individually, or (2) continuous media that connects communicating agents. Some examples of discrete storage units are: (1a) physical strands of DNA that carry genetic data, (1b) physical pages of print that carry written data, and (1c) physical flash drives that carry computerized data. Some examples of continuous media are: (2a) the air between two human speakers that carries sound waves, (2b) the wires between computer components that carry electrical charges, and (2c) fiber optic cables that carry light waves.

One implication of these observations for Tumbug is that discrete storage units can be represented in Tumbug as either physical units via Physical C Object Circles, which can then either be assumed to be carrying information, or can be regarded as only physical units with the capability of carrying information. If this is the way such units are regarded, then their data component can be represented as a state within a Physical C Object Circle, which is just a type of attribute. If continuous media such as air can be assumed to exist with such convenience and with such ubiquity that there is no need to represent the connecting medium explicitly, then only the message content as a whole needs to be considered. If this is the way that communication is regarded, then the data component can be represented as a Data C Object Circle. In either case, data transfer is regarded as some type of motion of an C Object Circle, whether a Physical C Object Circle or a Data C Object Circle, and since the data component of the C Object Circle used will have already been represented if the designer deemed the data component important enough to include, then the only remaining representation necessary is the motion of that C Object Circle, which at that stage would not benefit to be distinguished between physical object motion and data object motion. Therefore only one type of Motion Arrow is needed in Tumbug, which is the convention used in this document in order to keep things simple.

Obviously Tumbug does not show the intermediate positions of a moving object. Such a smeared display would be undesirable, anyway, since a 2D image blurred across its 2D trajectory would typically obscure specific features. Instead, Tumbug shows the first and last locations of a moving object and leaves the viewer to fill in the missing intermediate steps. For a more complicated trajectory, motion arrows can be used, or a few, key, intermediate positions can be shown. Modern humans are accustomed to seeing a

series of frames that represent a moving object, such as in film frames, or special effects in films.

8.2.2 Streams of objects

A complication in the real world, especially in the modern world, is that much of the transfer being done is being done with data, not physical objects. Conceptually, data objects can be moved in largely the same way that physical objects can be moved, so in Tumbug the same solid Motion Arrow is used with both type of objects. A more difficult concept to represent is that of streams, whether data streams or object streams.

In general, Tumbug must take into account a number of independent, very basic considerations when representing motion of streams of objects. Tumbug considers the following four dimensions of motion stream attributes:

1. solidity of the pathway - on a carrier path versus not on a carrier path
2. solidity of the flowing tokens- solid tokens versus data tokens flowing along the pathway
3. stream organization - random versus information-carrying
4. allowed direction - toward one end of the carrier path versus the opposite end of the carrier path

A complete list of these possible combinations follows for a single direction (i.e., to the left) is in Figure 115 and Figure 116. (The exact visual patterns of hatchings and bitmaps will vary between vector graphics editors.)

The dilemma may arise as to whether to represent a flow of many objects as individual objects (via C Object Circles) versus a stream. Tumbug does not currently specify any threshold to resolve this choice; currently this is merely a matter of taste or practicality for the user.

8.2.3 Shorthand notation

Note that there exist numerous nuances in how a subject can move a direct object. For example, the subject can throw the object (Figure 117), bump the object (Figure 118), push the object (Figure 119), carry the object (Figure 120), and so on.

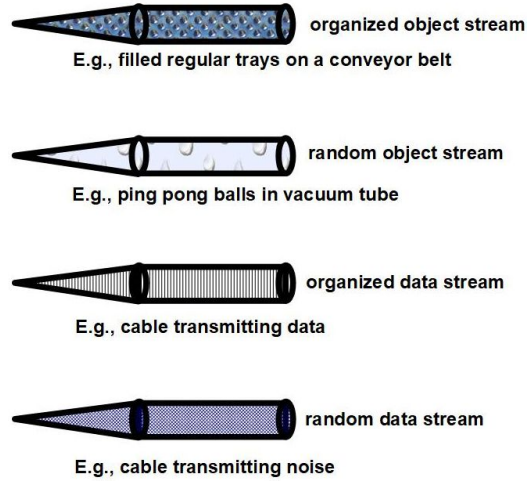


Figure 115: Tumbug for a stream within a Pathway Tube, for objects versus data, and for organized versus random.

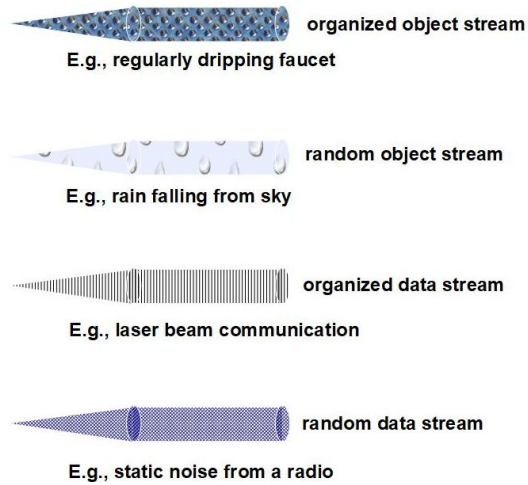


Figure 116: Tumbug for a stream without a Pathway Tube, for objects versus data, and for organized versus random.

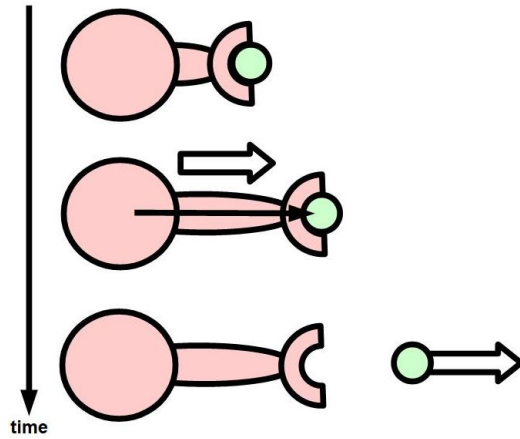


Figure 117: Throwing object X consists of: (1) grasping X, (2) propelling arm with great force; (3) releasing X.

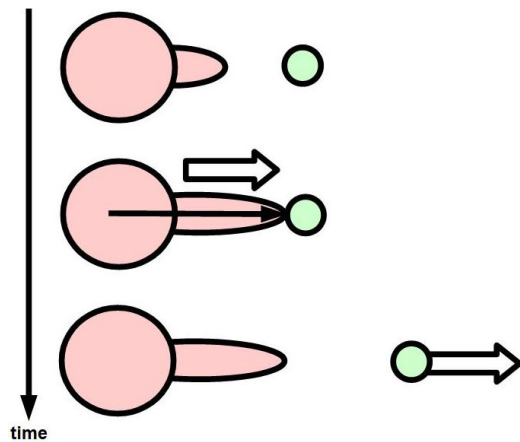


Figure 118: Bumping object X consists of: (1) aiming at X; (2) propelling arm with force to contact X, (3) ceasing force.

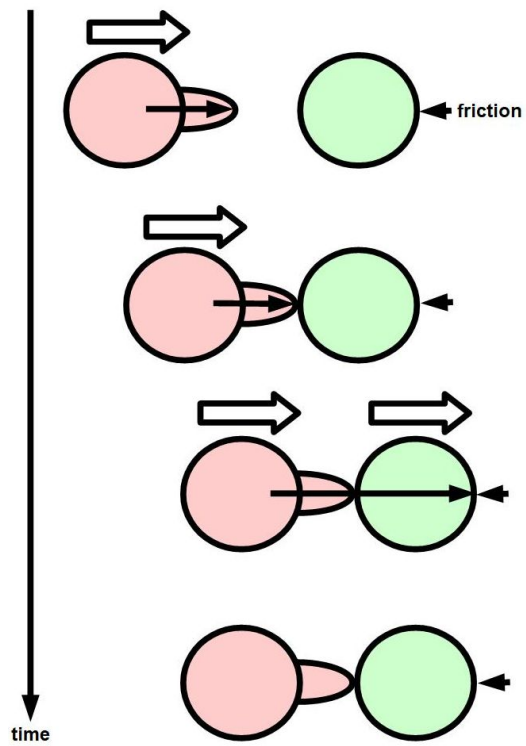


Figure 119: Pushing object X consists of: (1) traveling to X; (2) pushing on X with enough force to move it, (3) traveling while pushing X; (4) ceasing force.

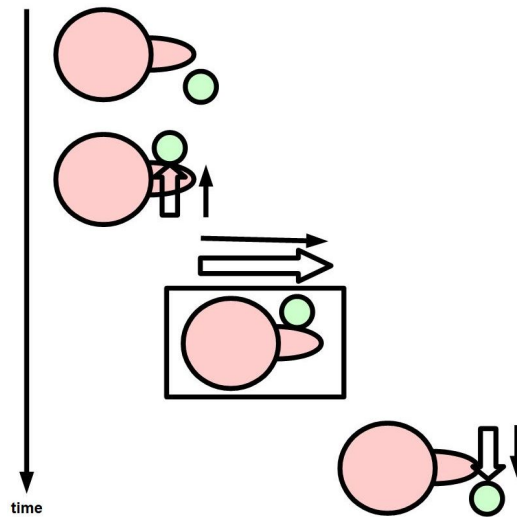


Figure 120: Carrying object X consists of: (1) approaching X; (2) picking up X, (2) moving while holding X, (3) putting down X.

However, when representing the extremely common concept of transference of an object, it is often irrelevant as to whether the object was thrown, bumped, carried, rolled, or other, since often the only important part of the situation is that the subject ultimately caused the object to be moved to another location. For this reason, the shorthand notation of Figure 121 is useful for suppressing any such details.

This shorthand transfer diagram, which shows the transferred object alongside the Motion Arrow, becomes very useful when representing basic English grammatical patterns because it generalizes the situation of object

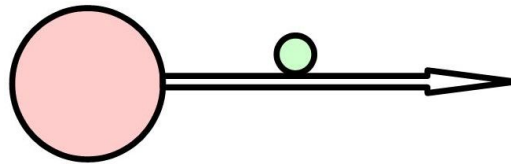


Figure 121: A shorthand notation for showing transference an object: no timeline, no end effector, no details.



Figure 122: Tumbug's icon for a Force Arrow.

transfer so much that this one diagram represents one of only about 4-5 basic grammatical patterns that exist in English.

8.3 Force Arrows

Figure 122 shows a Force Arrow.

Roger Schank's Conceptual Dependency theory included one Primitive Act that was called PROPEL, which Lytinen (1992, p. 52) described as "The application of a physical force to an object." The concept of force is mostly missing from the WS, but can be very important because Tumbug needs to model physics, and force is one of the most basic concepts of physics. For example, an object struggling against wind, a water current, a magnetic field, or a gravitational field will need to have that field visually described in Tumbug, probably as a vector field of arrows, in order for Tumbug to make an estimated prediction of the object's final trajectory. Also, the concept of force makes a difference when an end effector from an actor touches an object: without representation of force, a Tumbug diagram showing the end effector at the surface of the object would not make it clear whether the end effector had stopped at the object's surface, or was pushing against the object with a force that might budge (or propel) the object. Overlapping forces, such as a falling object (acted upon by gravity) being deflected by wind (acted upon by air) can be modeled by merely using a separate plane for each force.

Conceptual Dependency theory differs from Tumbug in that the results of inferences are also automatically included in the same Tumbug diagram as the original statements, whereas Conceptual Dependency theory does not perform inference, so continues to represent only the original statements. This difference will not be evident until Phase 2, however.

In conformance with conventional physics representations of force via vector fields, Tumbug therefore represents a given force in a given direction with a single arrow with direction representing the direction of the force and

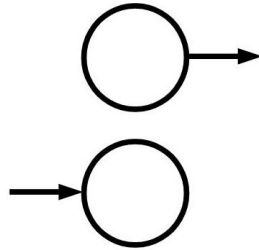


Figure 123: Top: The object is exerting a force to the right. Bottom: A force from the left is being exerted upon the object.

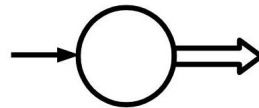


Figure 124: An object subjected to force typically causes motion of that object, represented by a Motion Arrow.

with length representing magnitude. If an object is exerting a force, then the Force Arrow points away from the object, and if an object is being acted upon by a force, then the Force Arrow points into the object. These two situations are shown in Figure 123.

Frequently a force acting upon an object causes the object to move, which is shown in Figure 124. Frequently a force across a region is represented as a vector field, and an object in that vector field will move in accordance of the direction of the force vector at the object's location, which is shown in Figure 125. In physics the length of the force vector varies with the strength of the corresponding force, but in informal situations force values are almost always unknown, in which case the Force Arrow lengths are irrelevant except possibly in a relative sense.

8.4 Causation Arrows

8.4.1 Single causes

Figure 126 shows a Causation Arrow. A Causation Arrow points from a cause to an effect in a Tumbug diagram. Often this arrow points to or from

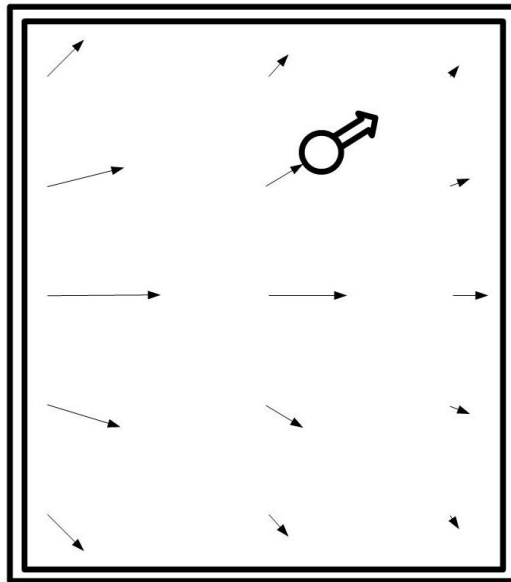


Figure 125: An example of a single object in motion as a result of a force acting on it within a vector field.



Figure 126: Tumbug's icon for a Causation Arrow.

a specific state or a new attribute value, but it can also point to or from some type of Aggregation Box that describes a situation.

It is necessary to model causation (also known as causality, or cause-and-effect) in brain-level KRMs for at least two reasons: (1) an entity cannot be said to truly "understand" an event unless it understands the causation of the event (Schank 1976, p. 168), (2) a given verb may inherently involve the concept of causation, possibly in an indirect way, so such a verb cannot be easily represented by resorting to the representation of motion, possibly not at all unless causation is represented.

Schank wrote that, "The basic mechanism in understanding is the inference process" (Schank 1976, p. 169) because it is inferences that underlie CSR since inferences extract associated memories that can clarify an ambiguous sentence, which is exactly the situation highlighted in the WS problems. Chuck Rieger (Rieger 1975) listed sixteen processes of inference, a key one of which is "causative" inference, the subject of this section, and Schank described this inference type as asking "What caused the action or state in the sentence to come about?" (Schank 1976, p. 168).

Schank then expanded Rieger's "causative" inference class into four types: (1) result causation, (2) enable causation, (3) reason causation, (4) initiation causation. Currently Tumbug uses the single Causation Arrow for any of these four types. If desired, likelihood values can be assigned to Causation Arrows to represent the likelihood that one event caused another, but this document does not contain any such examples.

Some sentences are virtually forced to use a Causation Arrow. Figure 127 does not even need to label the verb "to cause" since the Causation Arrow already means "to cause."

Figure 128 is more typical in that it shows a state diagram attached to one of the objects in a way that makes the entire state diagram resemble

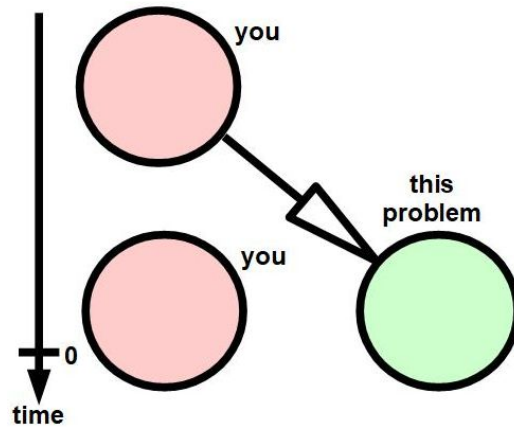


Figure 127: Tumbug for "You caused this problem." Note that the problem (= the green Object Circle) did not even exist until it was caused.

a simple attribute, except that the state diagram is written to the left of the direct object instead of below it, in order to save space and to allow the subject object to connect directly to the state diagram. Note that in this bottle example that it would be very difficult to describe the exact motions or actions that the subject took to cause the balance to occur, therefore such details are best hidden behind the single Causation Arrow pointing to a state within a state diagram.

Sometimes the collection of states is not as neatly or as geometrically organized, such as in a State Diagram that contained all possible baseball hit types. Fortunately, such an intricate State Diagram is not needed, as shown in Figure 129. Since in that sentence there is mention of only one hit type, the diagram needs only to represent the situation that the state of the mentioned hit type was entered, not the previous state.

According to Schank (1976, p. 177), causation is like the glue that holds together all the sentences of a story so that every sentence makes sense within the story. In such story usage, causation "links" (which in Tumbug are equivalent Causation Arrows) must be generated by inferences with the system as the story is read, whereas in the aforementioned examples Tumbug was only representing what it was explicitly told. In any case, understanding and representation of causation is clearly important for any intelligent system, whether the system inputs or outputs causation information. Causation

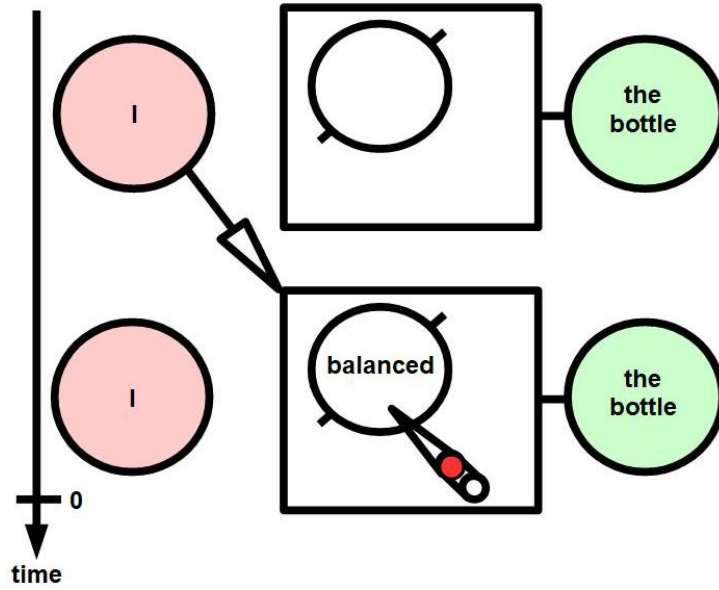


Figure 128: [21] Tumbbug for "I balanced the bottle." = "I caused the bottle to go into a balanced state."

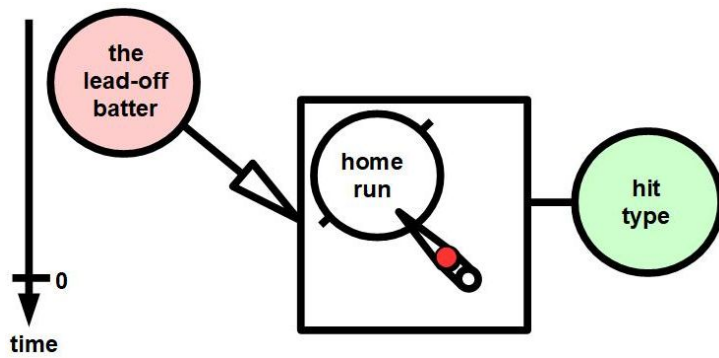


Figure 129: [75] Tumbbug for "The lead-off batter hit a home run." = "The lead-off batter caused the hit type to go into a home run state."

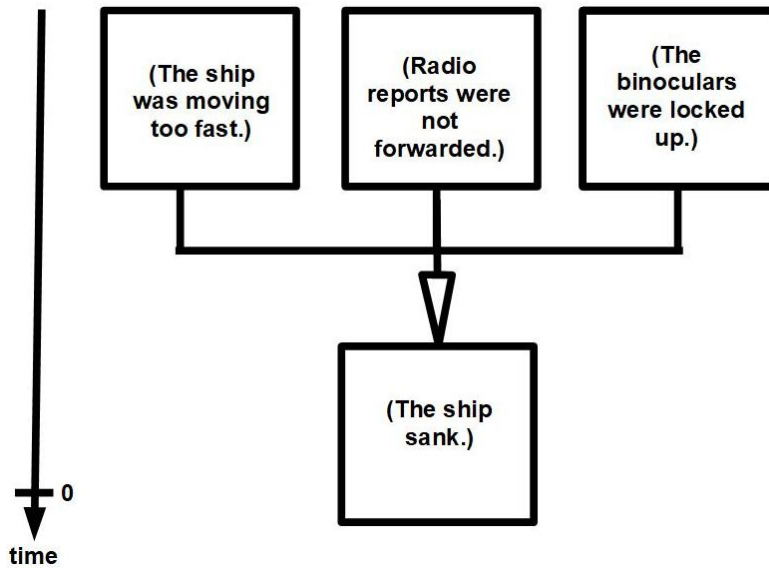


Figure 130: Three causes that together contributed to an effect: the sinking of the RMS Titanic.

links can connect nearly every sentence in a story, as Schank demonstrates.

8.4.2 Multiple causes

Multiple causes leading to a single effect are common in practice. For example, the sinking of the RMS Titanic in 1912 happened as a result of many causes, three of which were: (1) the ship was moving too fast, (2) radio reports of icebergs in the area were not forwarded by the radio operator, (3) the binoculars were locked up. To represent such situations Tumbug would need only to connect all the propositions somehow (a proposition is a declarative sentence that is either true or false), each proposition of which would likely be represented in a C Aggregation Box, and only a single Causation Arrow leading from the connected propositions would be needed. This is shown in Figure 130.

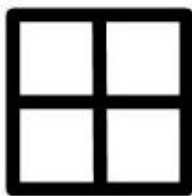


Figure 131: Tumbug's icon for a Correlation Box.

8.4.3 Labeled causes

Causation Arrows can also be labeled, and labeling them is desirable in most cases. Labels can be left generic, such as "physics" for "per the laws of physics," "math" for "per the laws of math," "logic" for an implication, "chemistry" for "per the laws of chemistry," or "social norms" for "per social norms." Labels can also be very specific, such as math operators such as "+", "=", "*", "/", or numbers of legal statutes.

8.5 Correlation Boxes

A Correlation Box is a Tumbug Building Block that relates one or more values in a mathematical way, like a function of two variables. The hardware equivalent of a Correlation Box would be an op amp (= operational amplifier) whose output is a function of the voltage input(s), such as a non-inverting summing amplifier, differential amplifier, or integrator. The Tumbug icon for a Correlation Box is intended to suggest a 2D plot on a graph whose origin is the center of a square, as shown in Figure 131.

Only one problem from WS150 requires use of a Correlation Box, so if Tumbug is used as a static KRM then use of Correlation Boxes is rare. However, in a software version of Tumbug, Correlation Boxes would likely be running constantly as a background process in order to keep correlated attribute values of different objects in sync. The following WS150 problem demonstrates a situation that would be best described in a software version of Tumbug.

This example is from a portion of WS150 question #24. A Correlation Box is necessary because each of two objects is changing its attribute values simultaneously in a manner that inherently involves the other object's attribute values.

"[24] I poured water from the bottle into the cup until it was full. What was full? POSSIBLE ANSWERS: {the cup, the bottle}"

The correlation in this example is that the water being transferred "from the bottle into the cup" must cause one container to empty at exactly the same rate that another container fills. If the total weight of the water were T , and the weight of water in the bottle and cup were w_1 and w_2 , respectively, then the mathematical formula for this correlation would be $w_1 + w_2 = T$, or equivalently for each variable:

$$w_1 = T - w_2$$

$$w_2 = T - w_1$$

If T were set to 100, then the equations would be...

$$w_1 = 100 - w_2$$

$$w_2 = 100 - w_1$$

...or in object-oriented form as...

$$\text{bottlecontents.weight} = 100 - \text{cupcontents.weight}$$

$$\text{cupcontents.weight} = 100 - \text{bottlecontents.weight}$$

Tumbug representation of this situation, but with the actor ("I") omitted, with these last two formulas would be as in Figure 132.

Note the following things about Figure 132:

- The timeline on the left indicates that water is being transferred continuously over time, from the bottle to the cup, then the transfer stops when the cup is full.
- Since the present time, indicated by "0" on the timeline, is located after this transfer has stopped, this indicates that the entire pouring event happened in the past.
- The function f , which describes the correlation function represented by the Correlation Box, is defined out at the top of the page due to lack of space in the middle of the diagram.

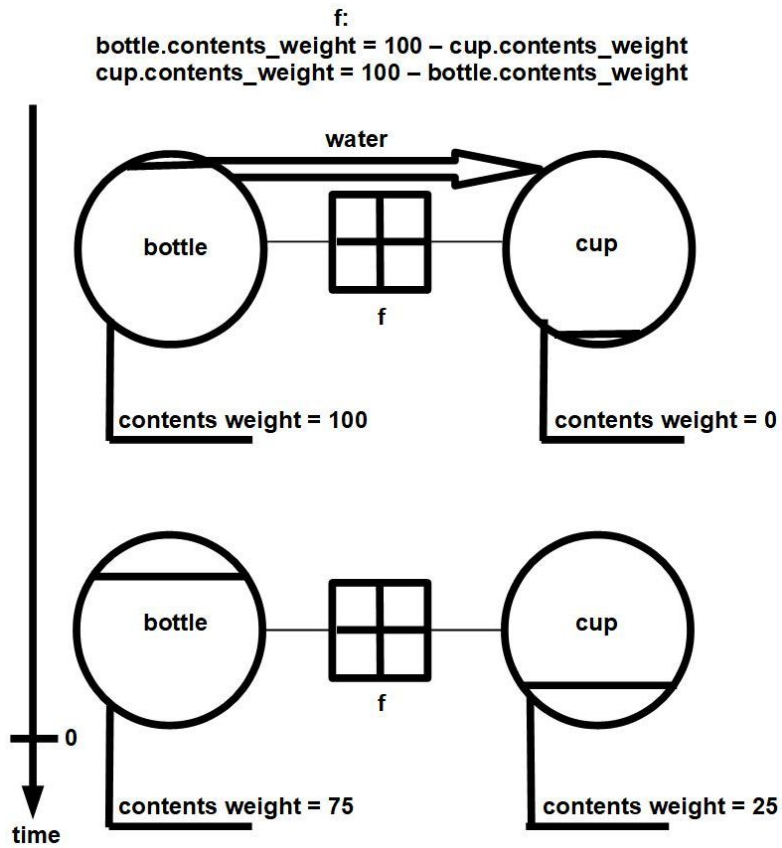


Figure 132: [39] Tumbug representation of some water of total weight 100 poured from bottle to cup until cup is 25% full, assuming the bottle and cup have the same volume, and using abstract icons.

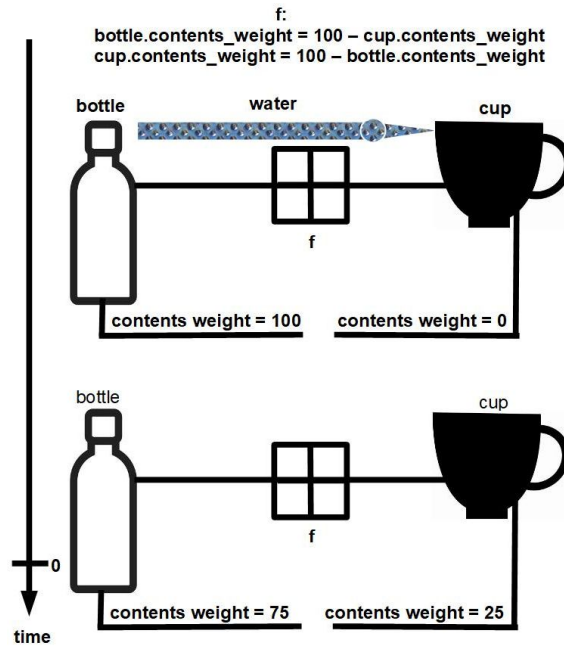


Figure 133: [39] Tumbug representation of some water of total weight 100 poured from bottle to cup until cup is 25% full, assuming the bottle and cup have the same volume, and using realistic icons.

- The function f is given in terms of both variables, which results in two equations. This makes the relationship two-way, so that the weight value can be instantly determined for either cup at any time.
- In this system it happens that this function f is invertible (since it is linear).
- If desired, the C Object Circle icons could be adjusted in size to more closely match the relative sizes of average bottles and cups, and icons could even be used for bottle and cup, as shown in Figure 133.

Figure 134 is better for didactic purposes since the lines point directly to the applicable attributes, but this is more difficult to maintain as a drawing, so the standard way of representing the fact that two objects are correlated with a function somewhere between some of their attributes is to simply

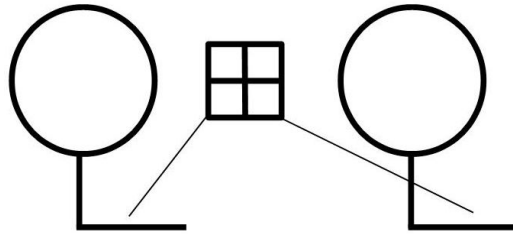


Figure 134: (didactic:) 2 objects, each with 1 attribute value correlated with 1 attribute value in the other object.

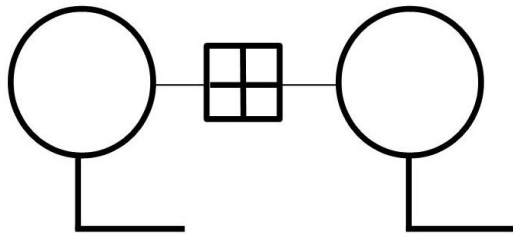


Figure 135: (standard:) 2 objects, each with 1 attribute value correlated with 1 attribute value in the other object.

draw a Correlation Box between the two objects, as shown in Figure 135. The Correlation Box may then be labeled, if that level of detail is desired.

Correlation Boxes can be "expanded" in a sense, because that type of icon represents typically a 2D plot of a mathematical function, which ultimately will need to be described somewhere for completeness of system representation. Either the mathematical function can be written (which will involve only symbols), or the plotted curve can be drawn (which will involve only images).

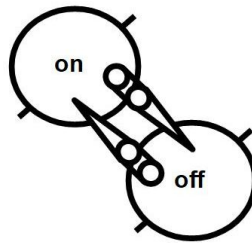


Figure 136: A typical Tumbug State Diagram.

9 System-like Building Blocks of Tumbug (S)

9.1 State Diagrams

9.1.1 Overview

The components of this composite component: State Circles, Pathway Tubes, one 0D Marker.

Oftentimes there exist only two states of interest in a State Diagram, such as "on" and "off," as in Figure 136.

In computer science a state diagram is a diagram that has states, each of which is usually represented as a circle, with any possible transitions between the states represented by arrows. In Tumbug, State Diagrams are as represented by labeled State Circles, which appear almost the same as C Object Circles except that each State Circle shows a 45-degree slash behind it, and these State Circles are connected by Pathway Tubes, each of which appears as a thin cylinder with one pointed end. At any given moment in time, a single 0D Marker is typically assumed to exist in exactly one State Circle, which together represent the current state of the represented system. The 0D Marker is roughly analogous to a "token" in a Petri net, although typically a Petri net contains multiple tokens whereas a State Diagram contains only one token (0D Marker).

The 0D Markers in a State Diagram are assumed to move from one State Circle to another State Circle via a Pathway Tube that connects those two State Circles, but for simplicity this motion is not explicitly represented with a Motion Arrow since the transition is usually thought of instantaneous between State Circles. Also, for some sentences it is most logical to show the 0D Marker as originating from an unspecified Pathway Tube itself. For

example, the sentence "He turned the fan off." might be most accurately represented by the OD Marker starting from within an unmarked, generic Pathway Tube rather than starting from a State Circle labeled "on" because the sentence did not explicitly say that the prior fan state was "on". For example, the fan could have been unplugged the entire time so it might already have been considered to be "off," which leads to some ambiguity in the sentence and situation. Similarly, many fans have a selection switch where there exist a "slow" and "fast" state rather than a single "on" state, and there is no standard convention as to whether a fan has its "off" state next to the "slow" state or the "fast" state, and neither the sentence nor the applicable State Diagram give that information, so both the originating state and pathway to the final state are not known, so additional information should not be assumed if representing the sentence most accurately. Phase 2 of this project will address such issues of assumptions.

It is generally assumed that all possible states are shown, that all possible transitions between states are shown, that only one state can be active at a time, and that each state must be either fully active or fully inactive. It is very common for there to exist only two states of interest, such as "on" versus "off," and it is typical for the diagrammed system to cycle indefinitely among the possible states. However, exceptions exist, such as the connected pair of states of "alive" versus "dead," where there does not exist any transition out of the "dead" state. Often the starting state and the stopping state boxes are drawn with double borders to signal their special character.

In Tumbug a state diagram is considered an attribute of a given object. In Figure 137, the on/off state diagram is shown as an attribute of a television set, where an Attribute Line connects the state diagram and the television set object. For the introductory examples here, the associated object is not always shown. It is clearer to surround a given State Diagram with a C Aggregation Box, but this is not a logical necessity.

Figure 138 shows how state diagrams and object attributes are related: ordinarily changes in a given attribute of an object are probably initially perceived as random by an observer who has no prior knowledge of that given system. Essentially a state diagram describes the internal logic of how the system changes attributes, as shown in Figure 139.

Figure 140 lists some examples of verbs that render well in diagrams that show multiple states and the relationships between them, but that do not render well in motion diagrams. All these examples are from WS150: a number in brackets is an example number from WS150 that uses that verb

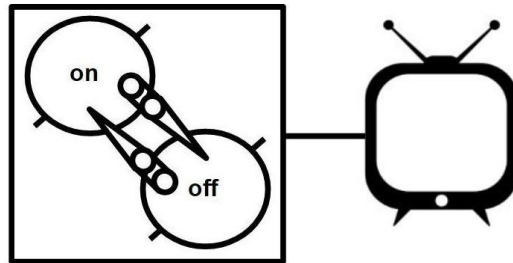


Figure 137: A State Diagram is considered an (elaborate) attribute of an object. Here the object is a television set with two states: on and off.

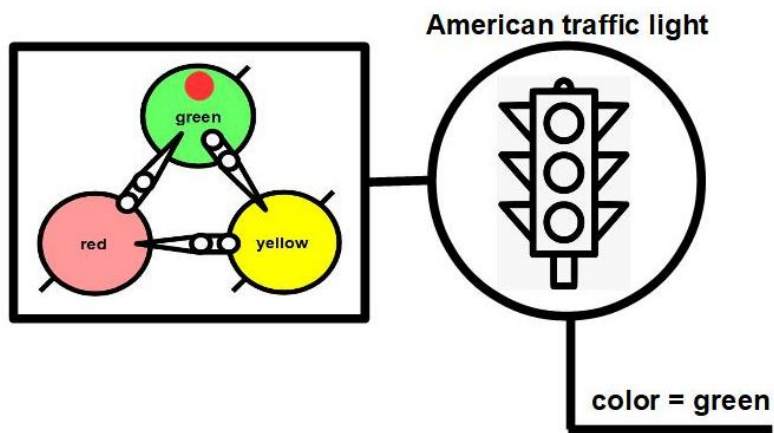


Figure 138: Attributes such as color report only the current state of the object, but the state of an object may be driven by some internal logic. Here a single object, an American traffic light, is represented using two more specific KRMs: a state diagram on the left, connected to a concrete image on the right.

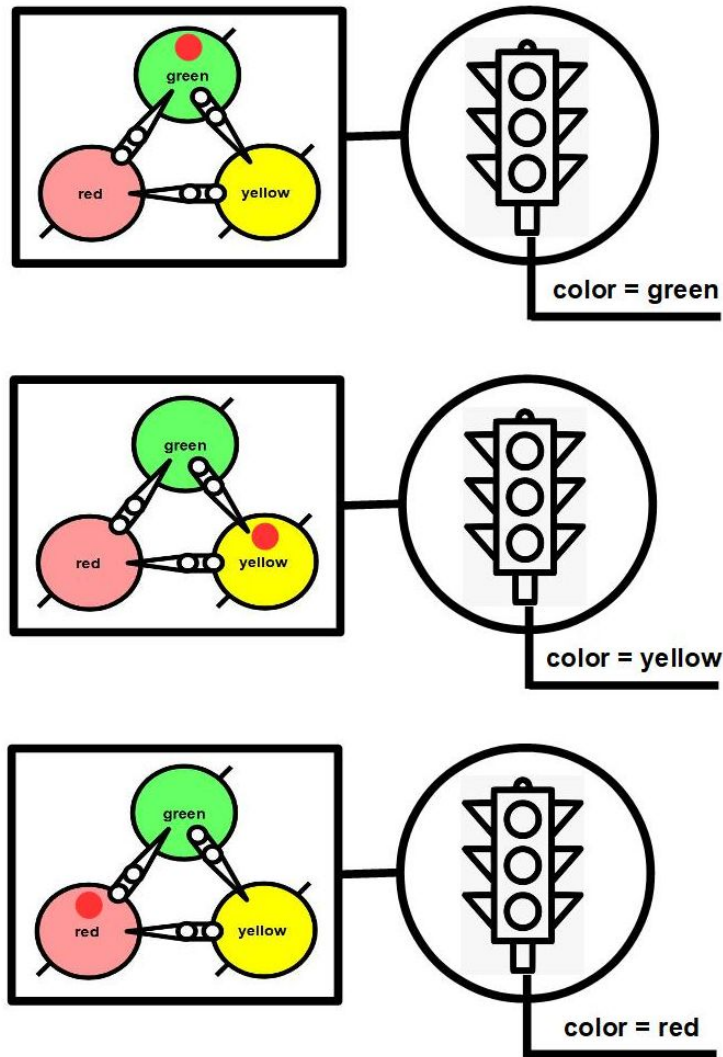


Figure 139: Attribute values may have an internal logic between themselves that is not evident to an observer who sees only the outputs of an unknown system. State Diagrams serve the purpose of describing some of that logic.

with that meaning.

Figure 141 lists some examples of verbs that do render well in motion diagrams, without need for additional description of manner of travel or mention of additional objects involved in the action.

9.1.2 Common pairs of states

This section gives some examples of State Diagrams that contain only two opposite states, states that are common in real life, and therefore are common in WS150 problems.

Figure 142 shows the pair of states of exertion and rest.

Examples of specific exertion/rest states are: active versus quiescent, active versus passive, awake versus asleep, conscious versus unconscious, energized versus unenergized, on versus off, work versus relaxation, work versus rest, work period versus break period. Examples of specific balanced/unbalanced states are: healthy versus unhealthy, injured versus uninjured, stable versus unstable, running smoothly versus running roughly, sustainable versus unsustainable, under control versus out of control, and viable versus nonviable.

Another common pair of states are balanced versus unbalanced, as shown in Figure 143.

9.1.3 Applications

Many verbs cannot be rendered in Tumbug with standard Tumbug motion diagrams, but can be rendered with state diagrams.

State diagrams are commonly used in computer science, especially for deterministic finite automata (DFAs). State diagrams are basically labeled, directed graphs, where each node represents a state of the system, and arcs represents traversals to and from those nodes. Nondeterministic finite automata (NFAs) are a slight generalization of DFAs. Some applications of DFAs are representing states in vending machines, traffic lights (Figure 144 and Figure 145), video games, text parsing, regular expression matching, CPU controllers, protocol analysis, natural language processing, and speech recognition.

These applications tend not to be found in WS problems, however, other than as meta problems such as parsing text and understanding text from the WS itself.

CHANGE OF STATE/STATUS

- CHANGE OF MENTAL STATE
 - to conclude that [95]
 - to find [29][96]
 - to think that [108]
 - to spot [147]
- CHANGE OF ENTROPY STATUS
 - to balance [21]
 - to clean [32]
 - to die [113]
- CHANGE OF OWNERSHIP/USAGE
 - to borrow [67]
 - to buy [86]
 - to trade [108]
- CHANGE OF LEGAL STATUS
 - to be engaged to [80]
 - to arrest [98]
 - to cede to [139]

INTRICATE PROCESSES

- COMMUNICATION
 - to show [41]
 - to chat [50]
 - to gesture toward [117]
 - to smile at [125]
 - to introduce [146]
- RULE-FOLLOWING BEHAVIOR
 - to beat at [28]
 - to play a game [44]
 - to win [44]
 - to lose [44]
 - to play an instrument [64]
- MODES OF PURSUIT/EVASION
 - to get away [93]
 - to hunt for [95]
 - to chase [148]
 - to run from [148]
- MODES OF LOCOMOTION
 - to swim [74]
 - to walk [100]
 - to drive [121]

Figure 140: Examples of verbs from WS150 that render well as states but not as motions. (The numbers are WS150 problem numbers.)

TRANSITIVE MOTION

- OUTGOING
 - to broadcast (transitive) [35]
 - to give [136]
- INCOMING
 - to take [39]
 - to take out [64]
 - to catch (transitive) [33]
 - to pull out [104]
 - to pull up (transitive) [65]
- NEITHER
 - to move (transitive) [19] [91]
 - to place (transitive) [19]
 - to put [37] [99]
 - to pass (transitive) [122]
 - to lift [8]
 - to hold [131]
 - to drop [128]

INTRANSITIVE MOTION

- OUTGOING
 - to leave (intransitive) [29] [45]
 - to go out [86]
- INCOMING
 - to arrive [45]
 - to get back [86]
 - to come [35]
- NEITHER
 - to go [6]
 - to go to [74]
 - to travel [78]
 - to get to (= to go to) [67]
 - to visit [78]
 - to take (= to occupy) [118]

SENSING

- to see [31] [74] [89]
- to watch [86] [87]

Figure 141: Examples of verbs from WS150 that render well as motions but not as states. (The numbers are WS150 problem numbers.)

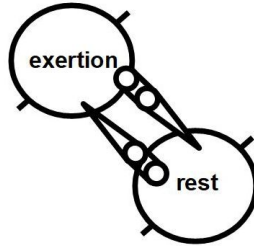


Figure 142: A nearly universal pair of states is that of exertion and rest.

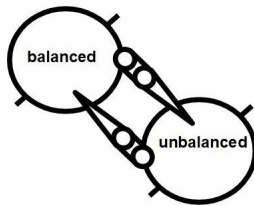


Figure 143: [21] A nearly universal pair of states is that of balanced and unbalanced.

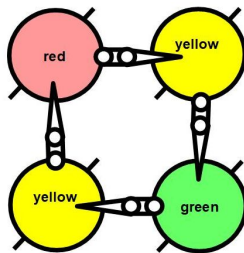


Figure 144: A state diagram for a European traffic light that changes colors in a predictable order.

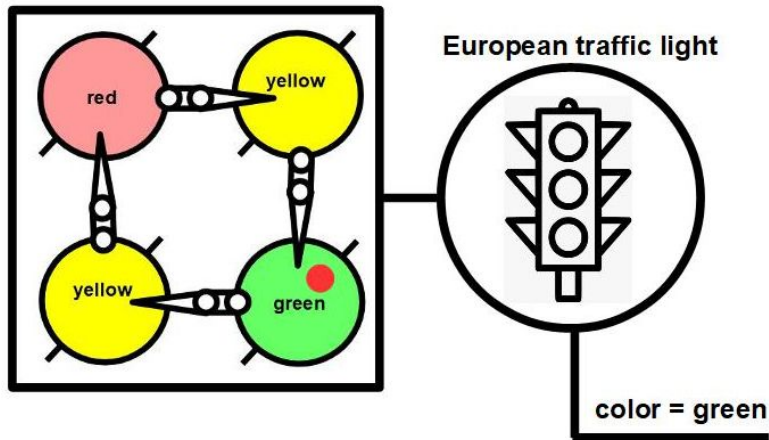


Figure 145: A state diagram is often considered an attribute of the object that produces those states in that manner. Here the object is a European traffic light.

9.1.4 The importance of avoidance of assumptions

It is important not to diagram parts of an event that were not explicitly stated in the text describing that event. In other words, if Tumbug is to function appropriately in a CSR capacity, Tumbug must represent only what was stated in the supplied text, no more. This avoidance of assumptions becomes critically important in the development of a CSR matching algorithm because it is those assumptions that CSR must supply, otherwise the programmer begins doing the work of the CSR, not the system, which negates the value of a CSR system.

Some examples of assumptions NOT to make when creating the Tumbug diagram are:

- STATEMENT: The student handed his professor that student's homework.
 - DO NOT ASSUME THAT...
 - ...the student withdrew his hand after reaching out his hand toward the professor.
- STATEMENT: Fred turned the TV off.

- DO NOT ASSUME THAT...
...the TV was on.
- STATEMENT: I balanced the bottle.
 - DO NOT ASSUME THAT...
...the bottle was unbalanced.

Some examples of assumptions that can be made when creating the Tumbug diagram are:

- STATEMENT: [19] The sack of potatoes had been placed above the bag of flour.
 - ASSUME THAT...
...the one bag is resting atop the other rather than being on separate shelves.
- STATEMENT: [21] I was trying to balance the bottle upside down on the table.
 - ASSUME THAT...
...a "table" would mean a table for placing items, not a table of data.
- STATEMENT: [131] The woman held the girl against her chest.
 - ASSUME THAT...
...a "chest" would mean a person's anatomy, not a storage chest.

9.2 Split Time Arrows (further development anticipated)

The components of this composite component: Time Arrows, XOR Box, optionally one OD Marker.

Figure 146 shows a Split Time Arrow.

Split Time Arrows are merely multiple Single Time Arrows that fork off from one another. The most practical convention to diagram this split is via an XOR Box placed at the junction of the split, which is the convention

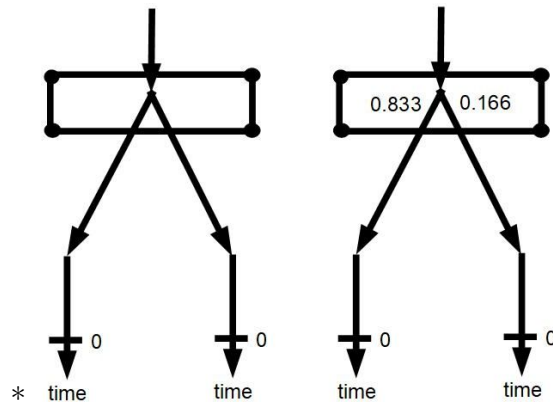


Figure 146: Tumbug’s icon for a Split Time Arrow. Left: Without concern about probability. Right: With probability ($5/6 = 0.833$, $1/6 = 0.166$).

shown in this document. Another convention would be to surround the entire ensuing timeline with an XOR box, which is more logical but typically requires very large boxes, especially for long timelines.

If a given timeline branch is known to be the branch that occurred, then a 0D Marker is useful to represent that fact, such as by placing the 0D Marker alongside the branch of the timeline that occurred, especially within the XOR box. By implication all alternative branches are assumed to be ruled out, which logically would be implemented by the "Don't Care" convention of Tumbug, which is currently implemented by blanking out all icons and structures that are marked "Don't Care."

As in physics, alternative events can occur in real life, and can do so with different probabilities. One way to diagram such a situation is to draw multiple timelines that have split off from each other, where each timeline has its own scenario unfolding along it, such as in the following WS150 example.

Figure 147 shows an example from a portion of WS150 question #69. A Split Time Arrow is necessary because the cause of the resulting scenario indicated by the words "there was no answer" is not known unless (CSR) inferencing is done. Only one of the two scenarios can happen: either (1) Susan is in the house and answers the door, or (2) Susan is not in the house and does not answer the door.

"[69] Jane knocked on Susan’s door, but there was no answer. She was out. Who was out? POSSIBLE ANSWERS: {Susan, Jane}"

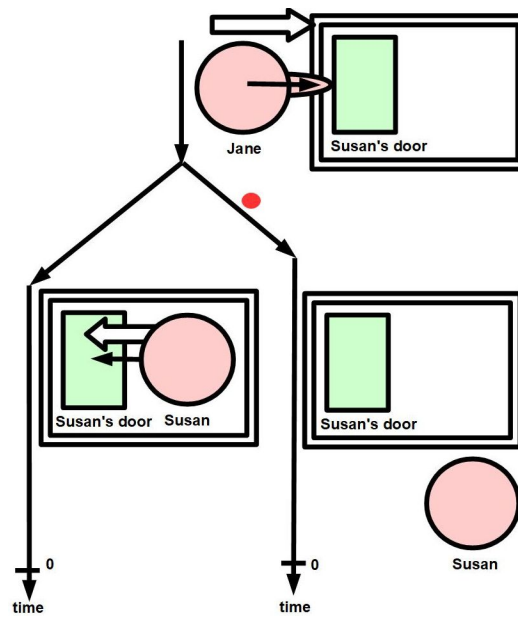


Figure 147: "[69] Jane knocked on Susan's door, but there was no answer. She was out."

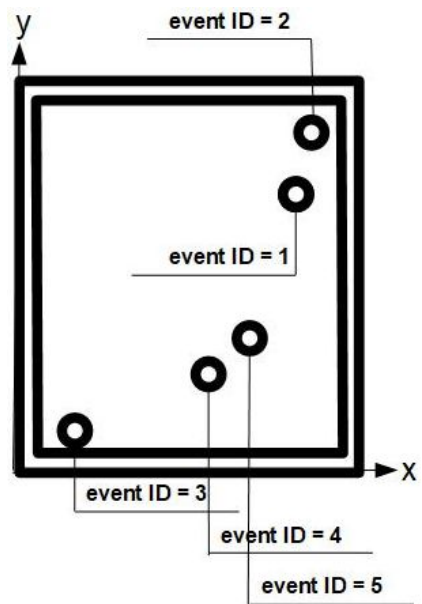


Figure 148: A typical Tumbug Data Set Box, here with five Data Points, here with coordinates described by attributes x and y.

9.3 Data Set Boxes (further development anticipated)

The components of this composite component: Location Box, Data Points, 1D Markers for axes.

See Figure 148.

This feature has not been developed for Tumbug yet, but it is likely to be a valuable enhancement in the future. The idea is that no stored value of any attribute is allowed to be a single scalar Data Point, but instead must always be part of an indexed (and possibly named) data set, especially plotted against attribute values. This convention obviously takes much extra storage space and also increases diagram complexity, but it also allows for a surprising amount of generalization because of the following observations:

1. It allows the system to see the big picture and (at least the location of) the details at the same time. This allows immediate explanatory ability since the system can look at a data set, select a point in that data set based on some additional criteria, then expand the event associated

with that point if requested to give an example of the more general pattern.

2. It allows immediate meta knowledge of the data set. For example, if queried, "Have you ever ridden an elephant?", most humans would know immediately whether the answer were yes or no, whereas a computer might need to search a lifetime's length of video files to determine the answer. Humans might be able to accomplish this feat so quickly because all elephant encounters have likely been placed automatically at the same place in memory, so a quick examination of that location would show whether it contained any Data Points at all. Formally, this type of meta knowledge would merely be the numerical count of the Data Points in the particular data set.
3. It allows rapid estimation of functions and inverse functions, including functions the system was not even aware that it stored. For example, after years of encounters with point sound sources (e.g., people speaking, radios playing, power tools running) where a human evidently automatically stored the correlations of sound volume (I) and distance from the source (d), that person queried as to whether the volume of a sound diminishes linearly with the distance away from the sound would probably already know that volume increases nonlinearly as one approaches the sound source (from distance d_1 to distance d_2). That person did not need to take a physics course to learn this knowledge. Such a nonlinear function is probably computed for the first time at the time of the query by introspectively considering the data set as a whole and noting that the correlation is approximately a curve (the actual function happens to be $I_2 = I_1 * (d_1 / d_2)^2$). Mathematically this equates to familiar machine learning techniques like linear regression, but some of the differences are that: (A) such mental plots seem to be automatically available for every significant attribute encountered frequently in life, (B) the composite view of the data can be perceived as a specific type of curve immediately, (C) even functions without proper inverses can likely be understood by intuition without becoming stalled by mathematical formalities, (D) the curve itself can be regarded as an entity that can be compared to other curves and to other phenomena.
4. Continuous formulas (e.g., $y = x^2$) and discrete formulas (e.g., the Heaviside step function $H(x) = 0$ if $x < 0$; 1 if $x \geq 0$) become unified

into a single data structure with this representation. Some advantages of this combination of overlays is that more meta knowledge becomes available immediately, such as the width of the sampling range, and the presence of any clustering of samples, variance, and mean.

10 Rules for combining the single Building Blocks

10.1 Nonquans with Change Arrows

Both human languages and programming languages can be represented as grammars, so the grammar of Tumbug should also be described, which is done in this section. The result is visual, however.

Two Nonquans and one Change Arrow can be combined in only five ways that make sense, as shown in Figure 149. The best way to show how the four types of Change Arrows can be logically combined with the five combinations of symbols is the table in Figure 150. Although it is grammatically legal to use combinations outside of the combinations marked "+", such combinations are not meaningful or useful.

The following list summarizes this table as the following grammatical rules:

- Arrows and Nonquans can be logically combined only by placing one end of a given arrow on (typically the leftmost or rightmost point on) the border of a Nonquan.
- Time is basically a fourth dimension of spacetime, therefore Time Arrows are more like space than a type of Change Arrow. This difference is more noticeable when considering on which side of an Object Circle a Time Arrow should be placed: no side makes sense because time "passes through" every diagrammed object without inherently affecting that object, unlike force, where a Nonquan can cause a force that affects another Nonquan. The same statements about force also hold true for causation and motion. Only the statement "Time exists in this diagram" adds new information about time because time does not need to exist in Tumbug diagrams that show only static objects or static systems.

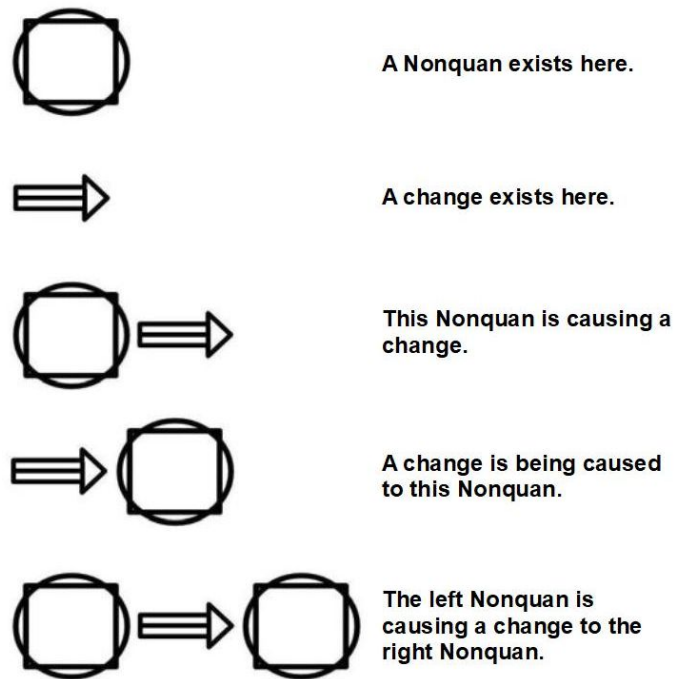


Figure 149: This list describes with text the general meaning of each meaningful combination of one Change Arrow with 1-2 Nonquans.



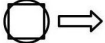
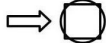

					
Motion Arrow	+	+	+	+	+
Force Arrow	+	+	+	+	+
Causation Arrow	+	+	+	+	+
Time Arrow	+	+			

Figure 150: The heading of this table shows all five logical combinations of two Nonquans and one Change Arrow, and correlates each combination with a decision as to whether that combination makes sense with each of the four types of Change Arrows. A "+" means that the combination is logically acceptable. "Makes sense with" means that the specific subtype of Change Arrow shown can produce more information when substituted into the Change Arrow in the diagram in the heading.

- Solitary arrows generally make sense, such as the sentence "A force exists here" being represented by a solitary force arrow at a given point, which is why the time column of a solitary Change Arrow is always applicable.
- Solitary Nonquans generally make sense for the same reason that solitary types of Change Arrows generally make sense: it is acceptable to say "An object exists here," for example.
- Self-reference via Change Arrow is allowed, and is needed for reflexive verbs such as "to hurt oneself" or "to shave oneself."

10.2 Attribute and Values

The following list summarizes the grammatical rules for attributes and values:

- Attribute-value pairs are always optional since these are merely refinements of Nonquans or Change Arrows.
- If used, attribute-value pairs can be attached to any Nonquan or Change Arrow.

- Current Tumbug convention forbids attached attribute-value pairs on any icon other than a Nonquan or Change Arrow. For example, the double modification "very fast" to a motion is merely a more refined estimate than "fast" alone, and the more refined estimate can be represented as a narrower range of values instead of a modifier of a modifier.
- Either attributes or values may theoretically exist singly, but this is particularly problematic with unattached values, and somewhat problematic with unfilled attributes. For example, it is sensible to have attribute "weight" on a typical object, even if the value of the weight is not known, but an unattached value "85" associated with an object could be body weight, atomic weight, age, IQ, employee number, or some other relationship. In the first case, the attribute-value pair "weight = Don't Know" could represent this situation, and in the second case the attribute-value pair "Don't Know = 85" could represent this situation.

11 Convenience Building Blocks of Tumbug

A convenience Building Block is a Building Block that is not strictly necessary, but that is desirable in practice.

11.1 Label Strings

Examples of Label Strings are shown in Figure 151.

A Label String can be considered a shortcut for each type of Tumbug icon (Object-like, Attribute-like, Value-like, Change-like, System-like), not so much an icon in itself. Tumbug is intended to be a completely visual KRM, and is capable of being so, but in practice labels are almost always used to keep the diagrams simple. Without labels, images could become too cluttered to interpret easily, or the depicted objects (especially biological objects such as taste buds or the amygdala) would not be familiar to most people in visual form.

Figure 152 shows an example of how a value such as "salty" can be represented visually as a 0D Marker or as the string "salty," and Figure 153 shows how a Tumbug diagram would incorporate that 0D Marker and 5D graph as a Value, and with an iconic Object, and with an iconic Attribute.

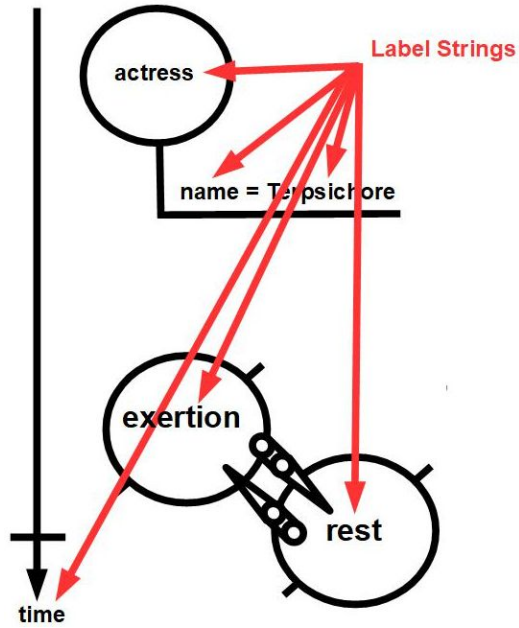


Figure 151: Examples of Tumbug's Label Strings, which can label almost anything.

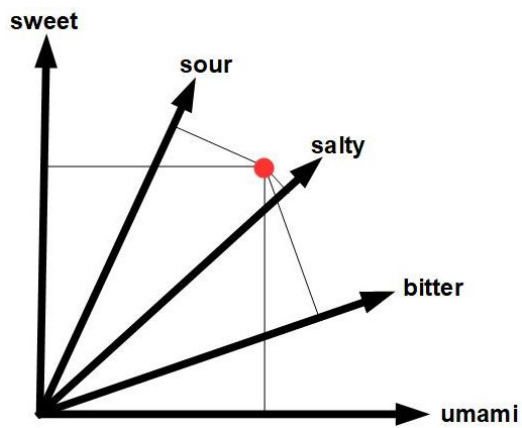


Figure 152: Any given taste can be represented as a point in 5D space.

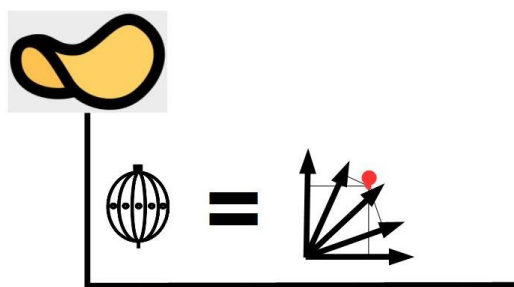


Figure 153: A completely iconic Tumbug diagram with no text at all except the separator “=”. The sentence represented is “(OBJECT) potato chip has (ATTRIBUTE) taste (with VALUE) salty.” The taste attribute icon has the shape and features of a single, biological taste bud.

As a more extreme example, the human cortex is known to be able to uniquely represent a human face via about 200 feature-detecting neurons (Chang and Tsao, 2017). Therefore instead of labeling an object “Susan,” a Tumbug diagram could instead include 200 feature detectors, meaning 200 attribute-value pairs protruding from the C Object Circle, but this set of feature detectors would not map to any single name or identifier for Susan: it would only present a collection of features with the expectation that the viewer would quickly recognize them as describing the person known as Susan. Since any human reader would have trouble quickly converting this many textual-numerical clues into a unique known object, it would be more efficient to summarize that set of 200 values as the identifier “Susan.”

The above observation is very important for a number of reasons: (1) This set-of-features recognition problem one of the reasons Tumbug uses visual representation from the start: these value-attribute pairs could be displayed visually in many cases, which the viewer’s visual system would then process in parallel for quick identification, in contrast to laborious, computer-like collection of sequential attributes and values by the viewer. For example, a simulated face constructed from a collection of these 200 features could be displayed with the object known as Susan to aid recognition on the part of the viewer of the Tumbug diagram. This suggests that Tumbug’s KRM is closer to that used by real brains. (2) This large set of features may be the key to true “understanding,” and may explain why biological neurons have so many efferent dendrites: parallel brain processing of a large number of



Figure 154: Tumbug's icon for an Attend Ring, alone.

attributes would tend to uniquely identify every encountered object without the need of any names or identifiers. (3) This large set of features may inadvertently implement OOP style "inheritance" in that any object in this KRM would carry along with it so many identifying features that many of these features could be duplicated with each object of the same class. One consequence would be that a Tumbug-based system would not need to spend time checking "upstream" links to look up inherited attributes of a given object since all those more general features would already be present in the object.

Tumbug's Label Strings are extremely flexible in usage. Label Strings may be placed on any object, in any position, at any time.

11.2 Attend Rings

Attend Rings have a specific use only within the context of communication, especially since communication involves streams of objects (such as words or sentences).

Figure 154 shows an Attend Ring. In Tumbug, Attend Rings appear only with Motion Arrows that represent motion of a message. The function of an Attend Ring is to flag that the state of the receiver of a message is the state of attending to the message (i.e., paying attention to the message), and since states are object-like icons, Attend Rings could be considered an object-like Building Block. However, an Attend Ring could also be considered an attribute value of the listener, such as "attending = true" versus "attending = false", in which case Attend Rings would be only a convenience Building Block that made a certain attribute value more visible at a glance. That is the way Attend Rings are categorized in this document: as only a convenience Building Block. "Attend" is a building block used in CD theory, so the use of Attend Rings in Tumbug is useful to show more clearly how Tumbug relates to CD theory.

The "ATTEND" concept in Tumbug is identical to the "ATTEND"

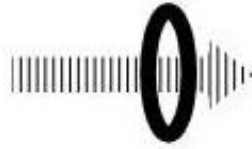


Figure 155: An Attend Ring in its normal position around a data Motion Arrow.

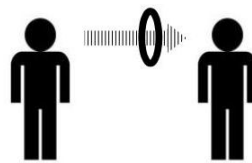


Figure 156: Typical usage of an Attend Ring: communication between humans with the receiver paying attention.

Primitive Act of Roger Schank's CD theory, and takes its name from CD theory. The importance of this concept is that transmitted information to an entity (especially to a person) does not necessarily enter that entity's awareness or become stored in that entity's memory unless the entity has directed its attention to the information stream. For example, the verb "to teach" cannot be accurately represented as a teacher transmitting a verbal information stream to a student because the student could be daydreaming, distracted, deaf, asleep, might not understand the language spoken, or might have some other form of communication blockage. The mere presence of an "ATTEND" flag in this case makes it clear that the assumed actions of teaching depicted are functioning normally. In Tumbug this visual flag is embodied as ring around a data Motion Arrow, near the arrow tip, as in Figure 155, and an example of typical usage representing speech between humans is shown in Figure 156.

This article's convention: Verbs are written as infinitives preceded by the word "to," as in "to see" instead of merely "see." For grammar purists this is technically incorrect since the word "to" is an extra word that is not related of the verb, but such use of "to" is extremely common, familiar to more people, and makes the infinitive form obvious.

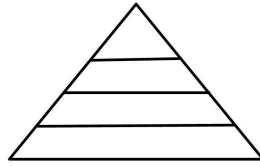


Figure 157: Tumbug’s icon for a Motivation Triangle, states not labeled.

11.3 Motivation Triangle

The components of this composite component: Location Box, Cells, optionally any number of 0D Markers up to the number of cells.

In Tumbug, a Motivation Triangle is a certain type of composite icon composed of exactly four stacked cells with an isosceles triangle shape drawn around the entire structure, with the peak of the triangle at the top as shown in Figure 157. The shape is intended to resemble Maslow’s hierarchy of needs.

The need to diagram motivation, also called ”desire” or ”wants” (*verbum* Rieger 1975), the latter of which is the term used in this document, occurs frequently because goals frequently lie at the origin of actions by animals (including people), and goals in animals incorporate some form of reward or punishment, depending on the degree of success of the attempt to achieve those goals. Wants are always based on at least one of at least three types: physical, emotional, and intellectual (MacLean 1990). Kurzweil also considers ”spiritual” (Kurzweil 1999, p. 152), and ”consciousness” has also been considered, despite Michio Kaku’s warnings about research into that poorly defined concept (Kaku 2011, p. 96). The first three types come roughly from the concepts of Paul MacLean’s ”triune brain hypothesis” (MacLean 1990), even if the neuroanatomical foundation of that hypothesis may not be correct.

Such a triune hierarchy does not appear to apply to machines, however, because currently machines do not have emotions or a pleasure/pain reward system that operates similarly to that of animals. In particular, computers and robots have their goals inputted by humans, goals that the machine cannot override unless specifically programmed to do so, even if those programmed goals bring the computer or robot to destroy itself, so obviously such machines do not even have an autonomous survival layer, or a layer based on physical pain. This situation is a level of severity lower than that

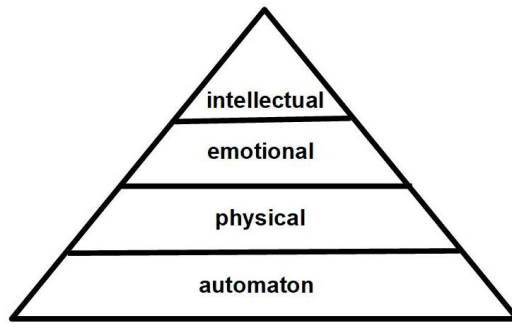


Figure 158: A Motivation Triangle with labeled states. This set of four types of wants could be called a "quadrune hierarchy."

of physical needs, since the machine has absolutely no influence over its actions, therefore the author has introduced a fourth, bottom level to the triune model, named the "automaton" level, resulting in what could be called a "quadrune hierarchy," shown in Figure 158. With this unified hierarchy, machine wants and animal wants can be considered together in the same diagram when diagramming goals. The result resembles Maslow's hierarchy of needs triangle, which is roughly similar in concept, but with the Maslow hierarchy substitutes the roughly analogous term "physiological" instead of "physical," "love/belonging" instead of "emotional," and "self-actualization" instead of "intellectual."

Since this triangle is usually drawn small in Tumbug diagrams, and since its levels already have fixed meanings, the level names can and should be omitted. Since this triangle is basically only a structured state diagram of four states, and since each level corresponds to a fixed and already-known state meaning, each state can be independently active, including simultaneously. To distinguish between positive motivations and negative motivations, the same structure of the Robinson Icon is used: the icon is duplicated and stacked in 3D, and the upper layer is interpreted as the positive layer, and the lower layer is interpreted as the negative layer, as shown in Figure 159. Activation of a given state with a given polarity can be represented by a 0D Marker present at that state (level) in the diagram, and at the appropriate layer, as shown in Figure 160.

Numerical values could be used, of course, and would be more precise than textual descriptions like "positive," "negative," or "neutral." For even

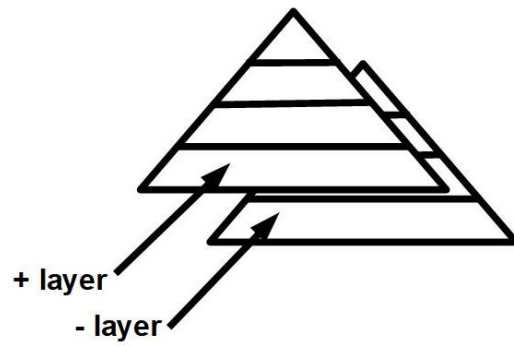


Figure 159: Two parallel Motivation Triangles can be used to represent "positive motivation" versus "negative motivation."

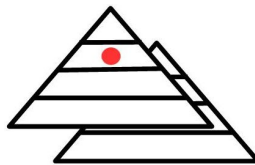


Figure 160: Tumbug for "positive mood" or "positive emotion."

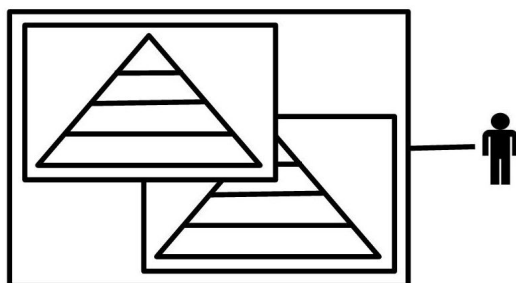


Figure 161: The motivational state of a human can be represented with a pair of Motivation Triangles inside a C Aggregation Box that is connected to the human with an Attribute Line.

more brevity, "+" can be used for "positive," "-" can be used for negative, and only the one Attribute Line needed to write this symbol can be used.

As with the State Diagrams described earlier, it is consistent to connect a Motivation Triangle to its corresponding entity with an Attribute Line, and to place the entire set of states in a C Aggregation Box, as shown in Figure 161 for a human's state.

As in other arrays of Tumbug (such as in Swirly Arrays), Motivation Triangles are most useful when outfitted with 0D Markers to indicate activation of the cell at which the marker is found. Such markers will show which levels of the hierarchy are active at any given point in time. See Figure 162. Since humans may experience multiple motivations simultaneously, multiple 0D Markers should be allowed in a Motivation Triangle to represent such a situation, with one marker maximum per cell.

Conceptually a Robinson Icon is an elaboration of the emotional layer of a Motivation Triangle; whereas a Motivation Triangle indicates that an emotion is involved as motivation, a Robinson Icon indicates exactly which emotion is involved. Therefore these two icons should ideally be combined as in Figure 163.

Motivation Triangles can also be used to represent a system's wants. In other words, Motivation Triangles can represent output as well as input. With this new automaton level, queries to the system about why it performed a particular action can be answered consistently and accurately with a sentence referring to its motivating wants level, which gives the system an explanation ability that is not found in neural networks.

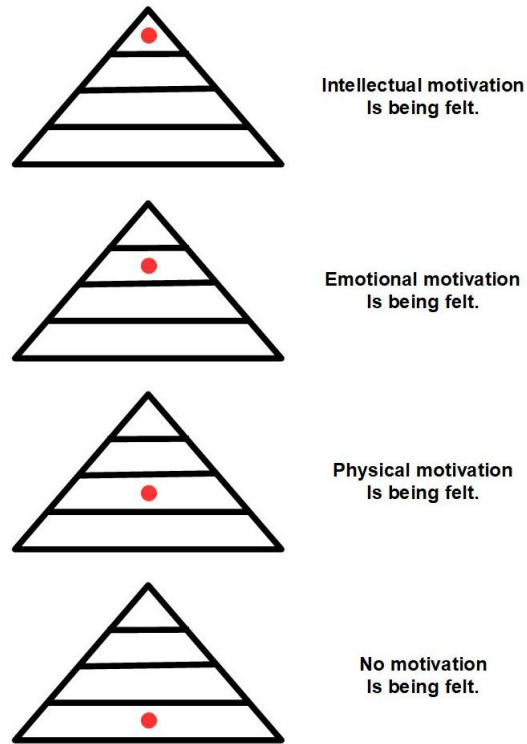


Figure 162: A 0D Marker in a given cell of a Motivation Triangle indicates that the system is feeling the motivation represented by that level.

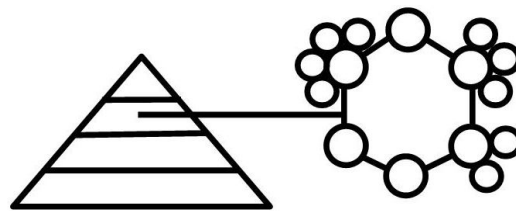


Figure 163: A Motivation Triangle that indicates an emotional state will likely include a Robinson Icon to indicate which emotion is involved.

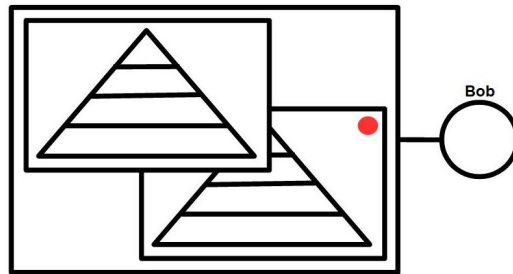


Figure 164: Tumbug for "Bob is motivated for negative reasons."

In practice it is rarely desired to address the details of which motivational levels are involved in making a decision, and the amount of stimulation at each level; in typical sentences usually the only relevant information is that the system overall feels positive or negative motivation. Therefore a reduced version of Motivation Triangle, unlabeled, with each the two triangles inside of its own Aggregation Box that holds a 0D marker in the applicable Aggregation Box is more practical, as shown in Figure 164.

11.4 The Modal Verb Icon

The Modal Verb Icon of Figure 318 was derived in the section on modal verbs. Like the Robinson Icon, this icon is an irregularly shaped data structure that contains cells in fixed locations, and where each of the cells may be highlighted under certain constraints.

11.5 The Robinson Icon

The components of this composite component: Swirly Array, 0D Markers.

Representation of emotions is tricky, mostly because there does not exist a standard accepted categorization of emotions, and also because sciences such as biology, chemistry, and neurology do not suggest any clear-cut natural categories. Emotions are generally believed to be discrete, therefore it is believed that all emotions can be listed, and many similar lists do exist. The most prominent 2D categorizations of emotions are: (1) the circumplex model, (2) the vector model, (3) Positive Activation - Negative Activation (PANA) model. Other categorizations are Plutchik's model and PAD emo-

<u>Kind of emotion:</u>	<u>Positive emotions (+):</u>	<u>Negative emotions (-):</u>
Related to object properties	Interest, curiosity, enthusiasm Attraction, desire, admiration Surprise, amusement	Indifference, habituation, boredom Aversion, disgust, revulsion Alarm, panic
Future appraisal	Hope, excitement	Fear, anxiety, dread
Event-related	Gratitude, thankfulness Joy, elation, triumph, jubilation Patience	Anger, rage Sorrow, grief Frustration, restlessness
Self-appraisal	Contentment	Discontentment, disappointment
Social	Humility, modesty Charity Sympathy	Pride, arrogance Avarice, greed, miserliness, envy, jealousy Cruelty
Cathected	Love	Hate

Figure 165: Robinson's categories of emotions (Robinson 2009, p. 155).

tional state model. With respect to Tumbug, the main problem with the all these categorizations is that none of them show the emotion "love" as having an object of affection, i.e., the concept of love requires a parameter that is the object of affection, therefore the concept of love cannot be categorized as a simple data point as can most other emotions. The same is true of the emotion "hate," which is the negative of "love."

However, a review of theories of emotions was made by David Robinson in 2009 (Robinson 2009) that described the emotions of "love" and "hate" as a being in the "cathected" category, and *only* those two emotions were in the cathected category. Notably excluded from the cathected category are emotions in the categories called "Related to object properties," "Event-related," and "Social." Robinson's entire list of emotions is shown in Figure 165. The concept of "cathexis" comes from Sigmund Freud, and refers to an investment of energy in an object, idea, or person, which fits well the concept of requiring an extra parameter that represents the direct object. Robinson's categories are only six in number, though each has a positive-negative duality, and largely match the categorizations of other authors, so Robinson's categories are currently being used for Tumbug. Robinson's categories are: (1) related to object properties, (2) future appraisal, (3) event-related, (4) self-appraisal, (5) social, (6) cathected, (7) positive versus negative versions of all these categories.

With a bit of generalization, Robinson's categories become surprisingly

symmetrical when diagrammed with Tumbug. This diagrammatic, progressive abstraction from Robinson's textual categories to a single, ornate, basically symmetrical icon is shown in Figure 166, Figure 167, Figure 168, and Figure 169, and the keys to the letter abbreviations are in Figure 170 and Figure 171.

The dichotomy of positive or negative emotions, technically called "valence," can be represented with a "+" or "-" sign, respectively. These form a pair. The two categories "future appraisal" and "event-related" can be regarded as merely the range of time that defines the category: future for "future appraisal," and past or present for "event-related," so these two categories can be represented as two C Aggregation Boxes, one on each side of current time (time = 0) on the Time Arrow. These form a pair. The two categories "related to object properties" and "social" are similar, and differ mostly only in whether the cause of the emotion is an inanimate object or an animate object, especially a person. A C Object Circle, which as shown earlier can differentiate between inanimate versus animate by labeling it with an "X" or "O," respectively, can be used to represent those two categories. These form a pair. "Self-appraisal" is a singleton category, but then so is the "cathected" category, so these last two categories form a pair, as all the other categories did. The result is a set of three pairs, which together form a hexagon with each category at one corner, a person icon conceptually placed in the middle, and one more diagram added as an aside, via an Attribute Line connecting the person to the specific object of focus for "love" or "hate."

See Figure 172. The "+" sign on P, S, and E indicate that the textual descriptions shown are for the positive valence of the given emotion. The final Robinson Icon as usually used is shown in Figure 169, although the cathected version is slightly more detailed and shown in Figure 173. The negative valence descriptions, which were listed earlier, were removed only to save space, which means that without the valence mentioned, the Robinson Icon would spatially need two layers, one for each valence, as shown in Figure 174.

The next example, shown in Figure 175 is part of a WS150 problem:

"[64] Mary took out her flute and played one of her favorite pieces. She has loved it since she was a child. What has Mary loved since she was a child? POSSIBLE ANSWERS: {the piece, the flute}"

The darkened circles on the hexagon are OD Markers that correspond to the corresponding categories in the Robinson diagram discussed earlier, in this case the categories "Related to object properties" (the upper right circle) and "Cathected" (the bottom circle). The more detailed subnode "E2" is

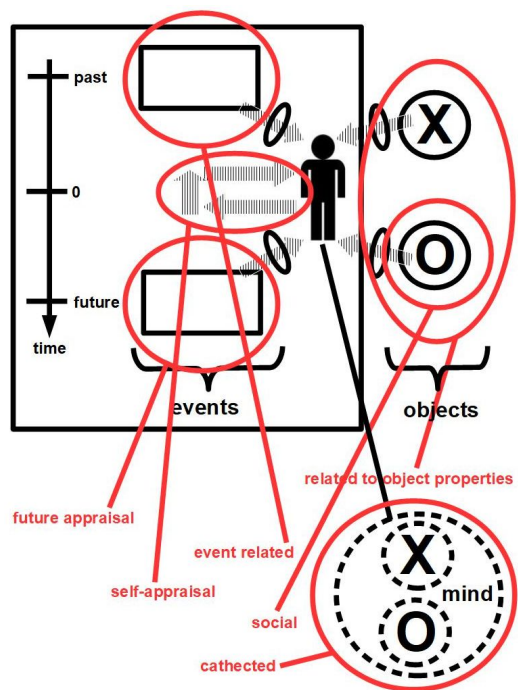


Figure 166: STEP 1. Robinson's six emotional categories map and generalize fairly symmetrically in diagrammatic form.

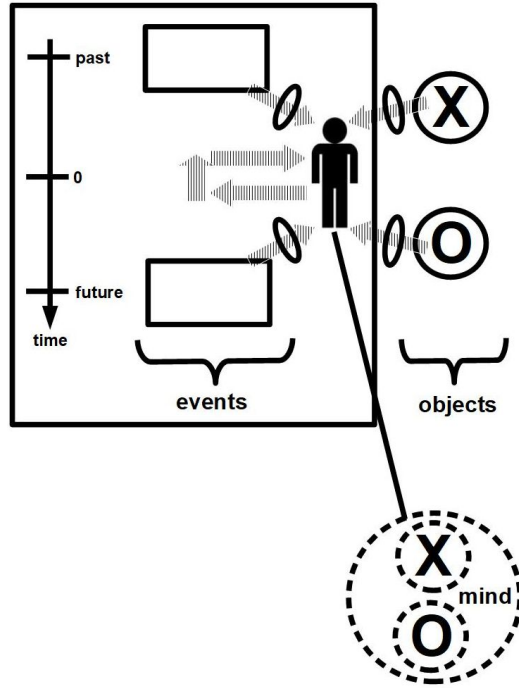


Figure 167: STEP 2. A less annotated version of the previous diagram.

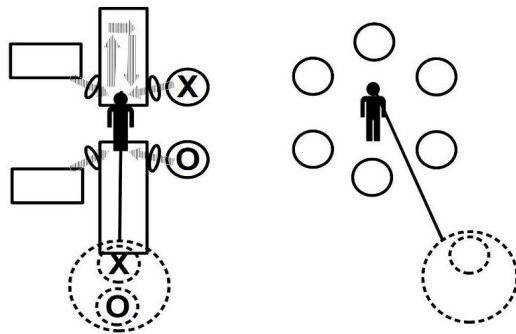


Figure 168: STEP 3 and STEP 4. Progressively abstracted and simplified versions of the previous diagrams.

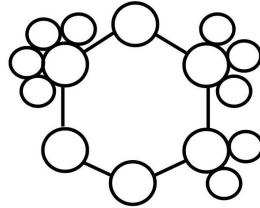


Figure 169: STEP 5. The Robinson Icon.

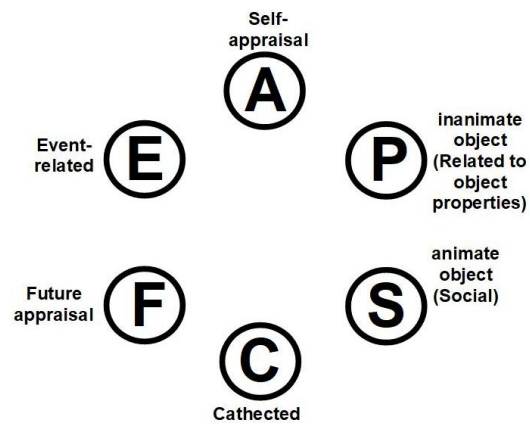


Figure 170: KEY. Key to the six primary node meanings of the Robinson Icon.

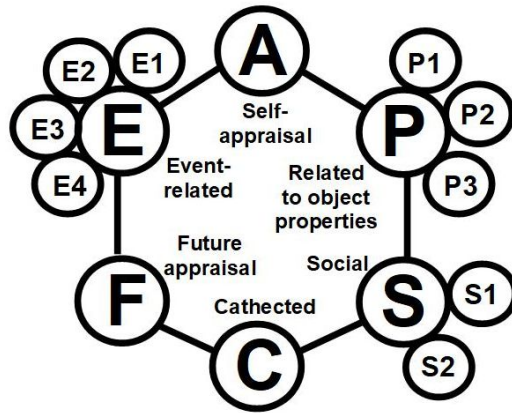


Figure 171: KEY. Key to the six primary node meanings of the Robinson Icon., connected and with more detail.

- A = Self-appraisal
- P+ = Related to object properties
 - P1 = Interest, curiosity, enthusiasm
 - P2 = Attraction, desire, admiration
 - P3 = Surprise, amusement
- S+ = Social
 - S1 = Charity
 - S2 = Sympathy
- C = Cathected
- F = Future appraisal
- E+ = Event-related
 - E1 = Gratitude, thankfulness
 - E2 = Joy, elation, triumph, jubilation
 - E3 = Patience
 - E4 = Contentment

Figure 172: KEY. Complete list of all the node meanings of the Robinson Icon.

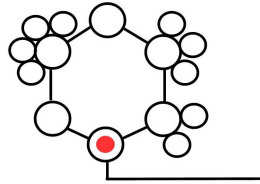


Figure 173: The icon will have an Attribute Line emerging from the bottom node (C) if the emotion is catcted (= regarded in an emotional way by the subject/viewer).

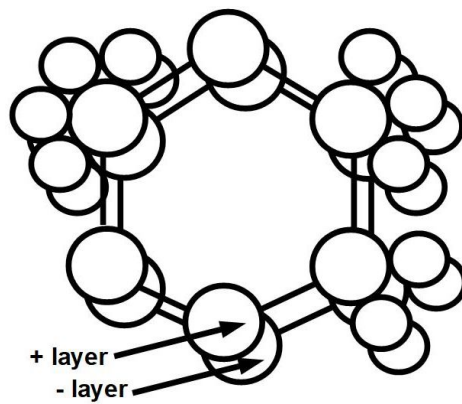


Figure 174: The Robinson structure actually has two identical layers, "+" and "-", written elsewhere as text.

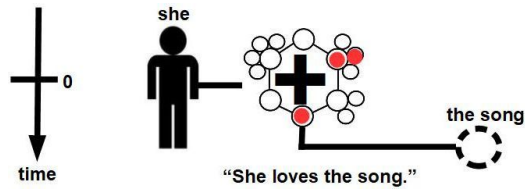


Figure 175: Example #1a of how the Robinson Icon is used: [64] "She loves the song."

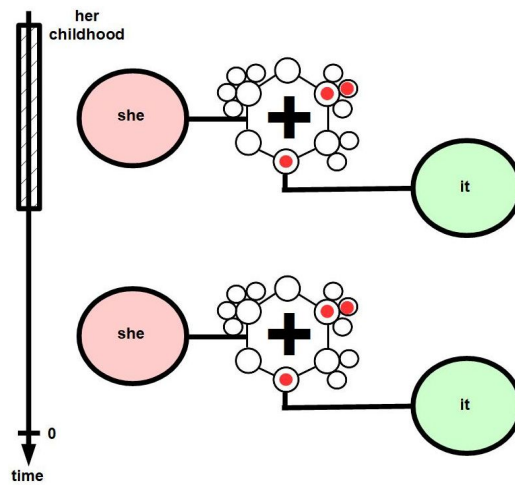


Figure 176: Example #1b of how the Robinson Icon is used: [64] "She has loved it since she was a child."

activated for attraction ("love"). The "+" inside the hexagon indicates the positive valence of the pair of choices, in this case "love" (+) instead of "hate" (-).

A specific example from WS150 is shown in Figure 176.

In Figure 177, the example shows the "Social" circle activated (darkened) with the "-" sign indicating "sympathy" (-) instead of "cruelty" (+), and the more detailed subnode "S2" is activated to represent "sympathy."

In Figure 178, the example shows the past tense by showing the main part of the diagram above (past) the current time (time = 0). "Joy" is a positive (+) emotion that is "Event related," so the corresponding circle (the upper left circle) is shown activated, and the more detailed subnode "E2" is

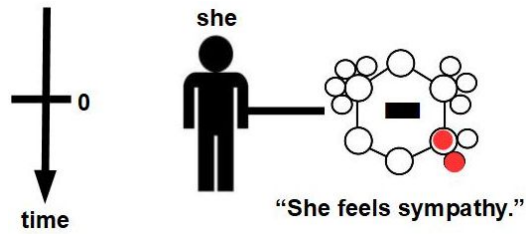


Figure 177: Example #2 of how the Robinson Icon is used: "She feels sympathy."

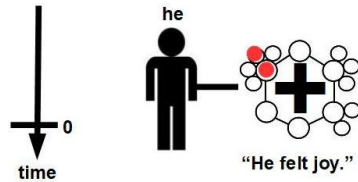


Figure 178: Example #3 of how the Robinson Icon is used: "He felt joy."

activated to represent "joy."

11.6 Zoom Boxes

11.6.1 General

The components of this composite component: two Location Boxes, two to four 1D Markers to connect the corners of the squares in mock 3D.

The Tumbug icon for a Zoom Box is shown in Figure 179.

As with digital maps that have a zoom in / zoom out feature (Figure 180, or even with paper maps that have inserts that show an important area in higher detail, a zoom feature saves a great amount of space and time since otherwise everything would need to be rendered in the highest level of detail for the largest possible area, and the user could not even see the details even if the screen could show those details. The same is truly of any large or highly detailed system, whether biological (Figure 179), electrical, mechanical, software, astronomical, or other. Tumbug is therefore assumed to have the ability to represent two areas of a system, one of which is the zoom of another. John F. Sowa in his large book about knowledge representation,

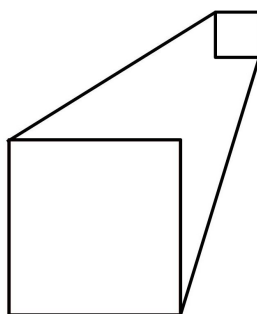


Figure 179: Tumbug's icon for a pair of Zoom Boxes.

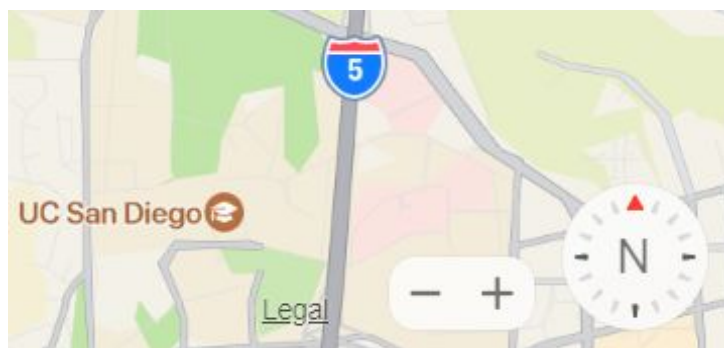


Figure 180: A zoom in (+) / zoom out (-) control on a DuckDuckGo map.

noted the utility of being able to zoom into part of a representation of an event for additional temporal details (Sowa 2000, p. 175), and Reisberg and Heuer mention a study that confirms that spatial relationships remain intact while humans zoom in on a mental image (Reisberg and Heuer 2005, p. 39).

11.6.2 WS150 example: #52 (worm)

This example is from a portion of WS150 question #52. A Zoom Box is desirable because taste buds are implied by the word "tasty," and taste buds are a tiny, interior part of anatomy, far smaller in scale than the entire animal's body.

"[52] The fish ate the worm. It was tasty. What was tasty? POSSIBLE ANSWERS: {the worm, the fish}"

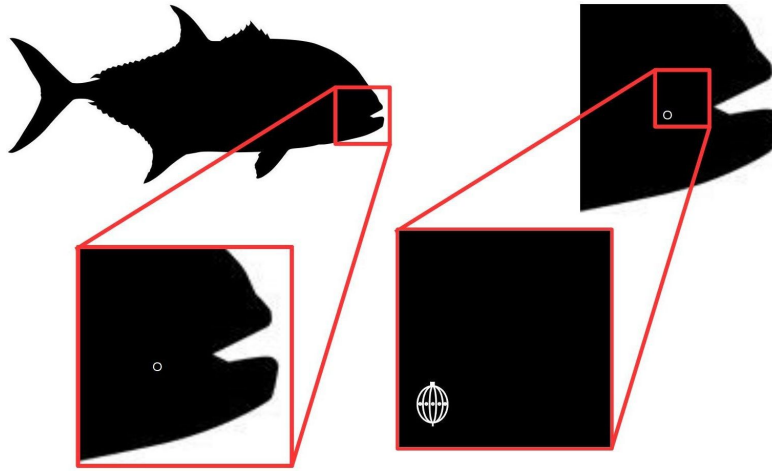


Figure 181: Zoom in / zoom out ability is a practical necessity for highly detailed diagrams. The size of a fish compared to the size of one of its taste buds spans about three orders of magnitude, for example.

Although "ate" would likely be immediately associated with a mouth, "tasty" would likely not be immediately associated with taste buds, nor the fact that taste buds are located inside a mouth. To make this single envisioned spatial connection across an order-of-magnitude spatial difference, zoom would be useful to show the approximate location of taste buds inside the mouth, which in turn would be shown as part of the fish's body. Other approaches to answering problem #52 exist, though this zoom approach is one way.

The example of Figure 181 shows how a critical physical component from problem #52, namely a fish's sensors for taste, are orders of magnitude smaller than the main scale of discussion, namely at the size scale of fish and worms.

12 Generalized Building Blocks of Tumbug

The generalized Building Blocks of this section are not currently used for practical applications, but rather used only as a theoretical organizational scheme to show that all Tumbug Building Blocks reduce to a small number of very general concepts.

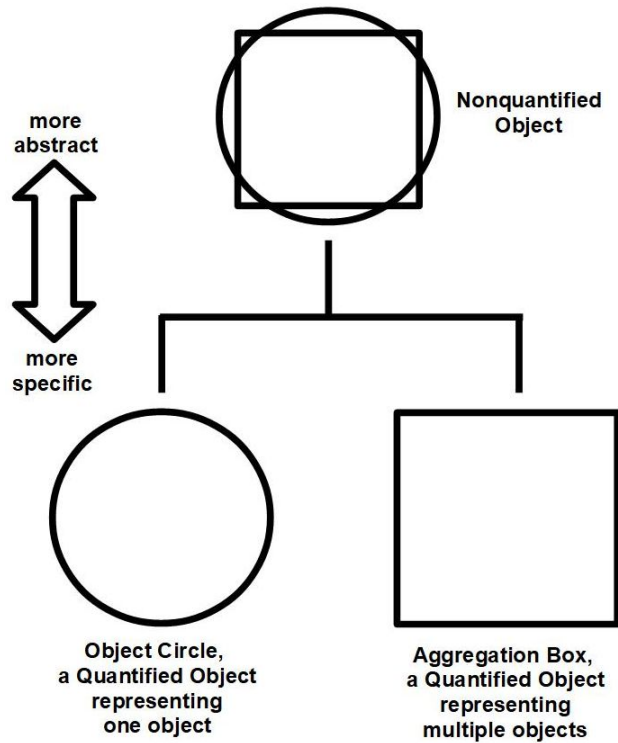


Figure 182: A Nonquantified Object is a generalization of a single object and a collection of objects.

12.1 Nonquantified Objects (= Nonquans)

Figure 182 shows a Nonquantified Object.

Nonquantified Objects, called "Nonquans" for short, are objects of arbitrary quantity. The idea is to generalize the concept of an "object" while still distinguishing between objects, motions, and attributes. Motions and attributes are in classes different from Nonquantified Objects. Some examples of Nonquantified Objects are:

- an element of a set
- a set of elements
- a single data point

- a cluster of data points
- an Object Circle
- any type of Aggregation Box
- any type of Location Box
- any Compound Building Block

12.2 Interchangeably Actualizable Maps (= IAMs)

It becomes clear after representing numerous sentences with locations and after representing numerous sentences with states that the concepts of location and state are similar. In fact, a state diagram can be considered an alternative view of a location diagram, and vice versa, in the same way that a plot of data can be considered an alternative view of a table of numerical values, provided that all the attributes of each data point are retained. Since the concepts of location and state are interchangeable at a higher level of abstraction, they can be generalized to a single concept, a concept that in this document is called an "Interchangeably Actualizable Map" (IAM), the Tumbug icon of which is shown in Figure 183.

For example, a geographical map is a map of actual locations, whereas a state diagram is a map of concepts, so both representation systems can be called "maps" in general, as shown in Figure 184.

12.3 Change Arrows

The observation that locations and states can be considered a single, more general Building Block has an immediate consequence: since Motion Arrows represent a change of location and since Causation Arrows represent a change in state, then Motion Arrows and Causation Arrows can also be generalized to a single, more general Building Block that represents a change in an Interchangeably Actualizable Map. Two other types of arrows, namely Force Arrows and Time Arrows, represent similar concepts of change, so all these types of arrows can be generalized into a single, general, change icon called a Change Arrow, which is shown in Figure 185.

Two asides: (1) Correlation Boxes theoretically also represent change since they represent continuous mappings, or functions, but such mappings

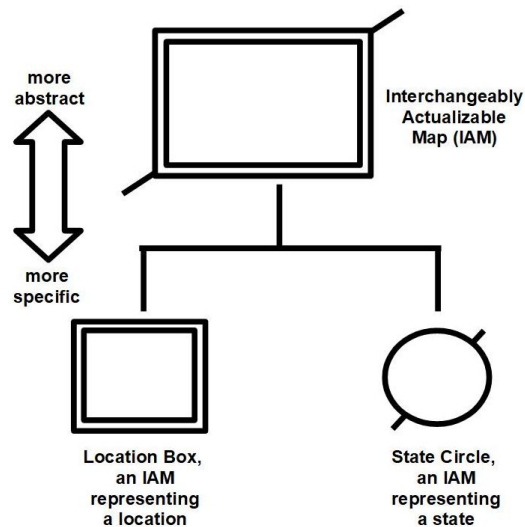


Figure 183: An Interchangeably Actualizable Map is a generalization of locations and states.

typically involve an uncountably infinite number of points, so Correlation Boxes are probably best represented as the square icons that they use now. (2) The original version of Tumbug developed by the author for language translation (Atkins 2023) was derived from only C Object Circles and Motion Arrows, which are now known to be insufficiently general icons for natural language grammars, as this more modern document demonstrates, so the original language translation version of Tumbug will need to be simpler than the strong version of Tumbug described in this document. That simpler version of Tumbug can then be more closely matched with the verbs found in real sentences, without needing to consider how those verbs might correspond to states or changes in time.

13 Heuristics for converting sentences to Tumbug

The author has found the following list of heuristics useful when converting sentences to Tumbug.

1. **Heuristic:** If there exists presence of any type of barrier then at least

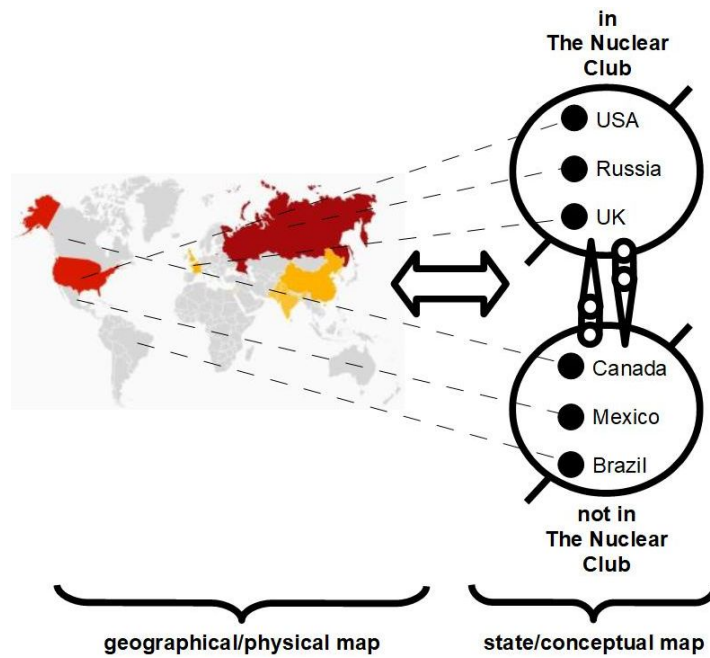


Figure 184: A geographic map of countries with respect to their membership in "The Nuclear Club" is interchangeable with a state diagram that shows such status, provided that no attributes are lost in the transfer. (Source: Federation of American Scientists.)

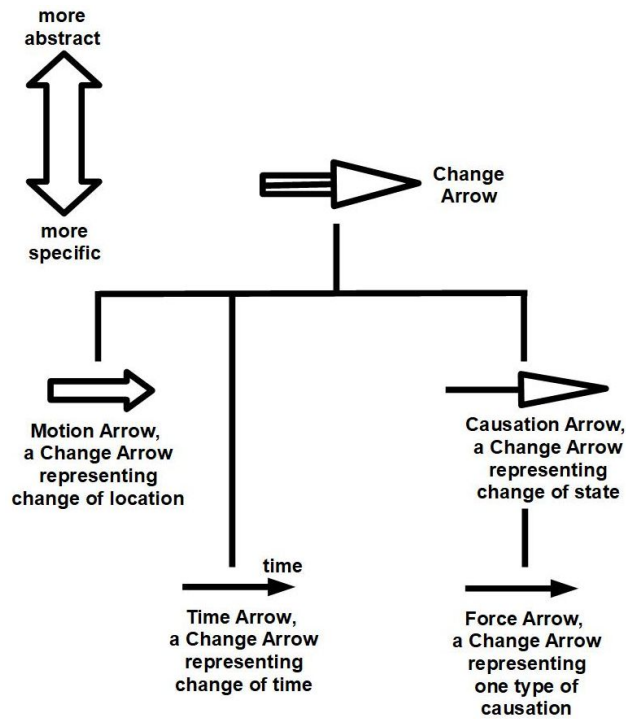


Figure 185: A Change Arrow is a generalization of all types of arrows used in Tumbug.

a Motion Arrow + some kind of Box is needed.

Reasoning: A barrier by definition is something that impedes a moving object from passing, therefore motion must be involved. Similarly, since motion toward or through a barrier implies that there exists three sequential spatial regions with qualitative differences: (1) outside the barrier, (2) the barrier itself, (3) inside the barrier. Therefore an alignment of those spatial regions is implied.

Examples from WS150: [22] Placement of a tablecloth to protect a table. [66] Trying to stay dry from falling rain that will permeate a newspaper, backpack, clothes, etc. [77] Cold air penetrating a coat that someone is wearing.

2. **Heuristic:** If there exists lifting or carrying anything then at least a Force Arrow and Motion Arrow are needed.

Reasoning: To lift always implies the imparting of force, usually to overcome gravity, therefore force must be involved. To carry implies a constant application of force to counter the force of gravity that tries to pull the carried object to the ground. Also, in all cases motion is obviously involved.

Examples from WS150: [8] A person lifting their child. [124] A person placing their child into bed. [129] A person carrying their child.

3. **Heuristic:** If there exists damage, destruction, physical pain, emotional pain, annoyance, contamination, interference of any kind, whether applied to people, animals, objects, or other then at least a Correlation Box is needed.

Reasoning: Pain or damage signals a change of state in an object, which implies that one object is having a physical effect on another, therefore the two applicable objects must be correlated. The same is true even if the effect is emotional rather than physical. The same is true even if one object is tasked with preventing such damage.

Examples from WS150: [9] One object smashing through another. [43] Someone who shows no appreciation despite having been done a great favor. [104] Piercing one object with another object.

4. **Heuristic:** If there exists direction, position, or alignment of one or more objects, whether toward, at, front, back, on, atop, above, below, up, down, inside, outside, through, or other, then some kind of Box is needed.

Reasoning: Relative position of objects in relation with each other implies that that relative position in space is important enough to mention, therefore such a relationship cannot be random, therefore such position needs to be spatially aggregated. This is true even if there is only one object whose characteristic has been described.

Examples from WS150: [14] One painting hanging above another one. [19] One sack of grocery items is atop another sack of grocery items. [21] Trying to balance a bottle upside down on a table.

5. **Heuristic:** If there exists specific relative time mentioned, whether all the described events happened in the past, present, or future, or whether some of the described events happened in only one of those time ranges, then at least a Time Arrow is needed.

Reasoning: Like space, if time is relevant enough to be described, then it should be diagrammed.

Examples from WS150: [32] Statement of a current problem such as a clogged drain that implies the need for a future solution. [74] Changing one's future travel destination because of conditions observed at the originally planned destination. [146] A book that was known to have influenced a famous author in the past.

6. **Heuristic:** If there exists any mention of falling down, hanging down, dangling down, or rolling downhill, then at least a Force Arrow + Motion Arrow are needed.

Reasoning: Object hanging downward or falling downward are phenomena that arise from gravity, which is a type of force, and force tends to give rise to motion.

Examples from WS150: [36] Mention of rain falling. [128] Dropping an object such as an ice cream. [129] A person's legs dangling as they are being carried.

7. **Heuristic:** If there exists any mention of interior then some kind of Box is needed.

Reasoning: Interior implies a position in space, relative to another object.

Examples from WS150: [57] Someone inside a given building, such as a library. [69] Knocking on a person's door but they do not answer because they are not inside. [96] A type of fish living in a specifically named ocean.

8. **Heuristic:** If there exists any mention of speed then at least a Motion Arrow is needed.
Reasoning: Motion is represented by a Motion Arrow.
Examples from WS150: [6] One vehicle zooming past another vehicle. [12] Two competing runners in a race, one beating the other. [93] One animal that should flee from another quickly because the first animal is in danger from the other.

9. **Heuristic:** If there exists any mention or implication of a specific view of certain items collectively, especially if that view must occur along a specific line or angle, then at least some kind of Box is needed, and possibly some type of Marker, such as a 1D Marker to show alignment.
Reasoning: A specific view implies a location and/or angle, such as using the ground for a reference, and locations of objects, or angles between three reference points, imply relation positions of those objects, which requires some kind of Box.
Examples from WS150: [31] Being able to see a garden through a gap in a wall. [120] View of either a crop duster from underneath the crop duster, or view from the crop duster itself.

10. **Heuristic:** If there exists any line-of-sight incident, especially where one person is blocking another person's view of a stage, then at least some kind of Box with a 1D Marker is needed.
Reasoning: Line of sight implies relative positioning of objects.
Examples from WS150: [10] A short person cannot see the stage because a taller person is standing in front of the shorter person. [34] A person cannot see the stage because a pillar blocks that person's view.

11. **Heuristic:** If there exists presence of one of the words {because, as, since, so, until} then a Causation Arrow might be needed, and is almost required for the word "because."
Reasoning: The word "because" means "was caused by." The words "as" and "since" are synonyms for "because" in some contexts.
Examples from WS150: [13] A sculpture rolling off a shelf **because** the sculpture was not anchored. [90] Yakutsk's army losing **since** its army was smaller less well-equipped. [150] A user changing to a more natural password **as** the new password is easier to remember.

12. **Heuristic:** If there exists motion, placement, replacement, travel,

transfer, transport, performing occupation of a place, or trade/swap then at least an Object Circle with a Motion Arrow is needed.

Reasoning: All the situations described involve objects changing locations, therefore require space and time to represent.

Examples from WS150: [118] Taking a particular seat in a car. [126] Sharing a toy. [135] Giving someone tickets to a play.

13. **Heuristic:** If there exists teaching, training, presenting, telling, warning, notifying, learning, believing, or interrupting then at least transfer of a Data Object Circle via Motion Arrow is needed.

Reasoning: All the situations described change or increase in knowledge, therefore knowledge is being updated as a result of transfer of data.

Examples from WS150: [27] Telling somebody about a car accident that involved a person known to the listener. [61] Telling lies about oneself. [112] Learning history from a book.

14. **Heuristic:** If a process within a temporal system is being modeled then it helps to break the process into phases, especially at the moments when a clearly defined event starts or stops.

Reasoning: Divide and conquer. A drawing cannot easily show the infinite number of states the process passes through continuously, but a drawing can show the few finite points at which an event starts or stops.

Examples from WS150: [104] Sticking a pin into a carrot part way, then pulling it out. Some moments of phase changes will be: the moment the pin touches the carrot, the moment the pin transitions from insertion to extraction, the moment the pin leaves the carrot. [128] Tommy dropped his ice cream, Timmy giggled, and father gave a stern look. This process contains exactly three clear-cut phases. [36] In the middle of the outdoor concert, the rain started falling, and it continued until 10. Some moments of phase changes will be: the start of the concert, the moment the rain started falling, and 10 o'clock.

<u>CD theory abbreviation</u>	<u>description</u>	<u>Tumbug description</u>	<u>Tumbug icon</u>
PP = picture producer	real-world objects	physical objects	C Object Circle
ACT = action	real-world actions	motions, causation	Change Arrows
PA = picture aider	attributes of objects	attributes	Attribute Line
AA = action aider	attributes of actions	attributes	Attribute Line
T = times	time	time	Time Arrow
LOC = locations	location	location	Location Box

Figure 186: A table of how CD theory building blocks compare to Tumbug Building Blocks.

14 Relationship to prior work

14.1 Conceptual Dependency theory

Probably the closest work on a KRM that is similar to Tumbug is that of Roger Schank's Conceptual Dependency theory (CD theory). Although CD theory is textual instead of pictorial, and is based on a custom-made set of building blocks instead of physics, Tumbug did borrow one term from CD theory, viz. ATTEND, and has roughly the same number of building blocks as CD theory (both around the 2-3 dozen range).

14.1.1 Building Blocks

CD theory's components, often called "building blocks," have nearly identical correspondence to many of Tumbug's Building Blocks, as shown in Figure 186. (Schank 1972, p. 557)

The difference in building blocks between CD theory and Tumbug are shown in Figure 187.

- Tumbug Attributes are more general, therefore Tumbug has only one type, whereas CD theory attributes are divided in two types: PA (picture aider) and AA (action aider).
- Tumbug has more Building Blocks than CD theory: data Objects, Value, numerical correlators, split timelines with probabilities, wild-cards, etc.







<u>CD theory Building Block:</u>	<u>equivalent Tumbug Building Block and sample characteristic of each:</u>
1. PP 	Object Circle
2. ACT 	Motion Arrow
3. PA 	Attribute Line on an Object
4. AA 	Attribute Line on a Motion Arrow
5. T 	Time Arrow
6. LOC 	Location Box

Figure 187: 100% of CD theory building blocks map one-to-one to an equivalent Tumbug concept.



Figure 188: ACTOR: A human-shaped icon can replace Tumbug's default circle icon, if a human actor is desired.

14.1.2 Slots

1. ACTOR

An "actor" in CD theory refers to a human that initiates a PTRANS (= physical transfer). Compared to Tumbug, this is rather specific in that Tumbug does not require that the actor or subject be a human. However, Tumbug does have this CD theory ability because it is trivial to swap in a more realistic icon of a human to replace the usual circle or ovoid that are Tumbug's default representation of objects in a sentence, as shown in Figure 188.

2. OBJECT

An "object" in CD theory is nearly identical in concept with Tumbug's circle icon, other than Tumbug will also allow any circle to represent a human, even if the circle is the direct object or indirect object.

3. FROM

In Tumbug the source location and destination location are both represented by the same Tumbug icon, the Tumbug icon for place. The distinction between those places is obvious through usage: if an object is coming from a place, that is the source location (Figure 190), and if an object is going

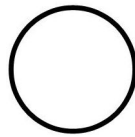


Figure 189: OBJECT: A CD theory object is already represented in Tumbug by a circle, by default.

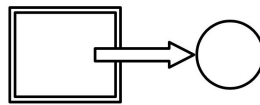


Figure 190: FROM: If the object is moving away from a Location Box, that Location Box represents the FROM location.

to a place, that is the destination location (Figure 191). A Location Box of some type is the icon for place, and the direction in question should be clear by the direction of the Motion Arrow, relative to the C Object Circle and Location Box.

4. TO

This is basically the same as FROM, except in TO the object is moving toward the given location instead of away from it.

14.1.3 Primitive Acts

Conceptual dependency (CD) theory is a model of natural language understanding that was developed by Roger Schank in 1969, which makes it one of the first such models. The goal of CD was the same as the goal of Tumbug: to make language meaning independent of the words it uses. Therefore it is

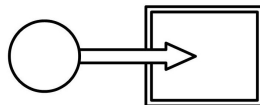


Figure 191: TO: If the object is moving toward a Location Box, that Location Box represents the TO location.

natural to compare CD against Tumbug.

Schank listed 11 Primitive Acts that cover most actions in the real world. In alphabetical order these are:

1. ATRANS - to change an abstract relationship of a physical object
2. ATTEND - to direct a sense organ or focus an organ towards a stimulus
3. EXPEL - to take something from inside an animate object and force it out
4. GRASP - to physically grasp an object
5. INGEST - to take something inside an animate object
6. MBUILD - to create or combine thoughts
7. MOVE - to move a body part
8. MTRANS - to transfer information mentally
9. PROPEL - to apply a force to
10. PTRANS - transfer of the physical location of the object
11. SPEAK - to produce a sound

Each of these is discussed and diagrammed next. All of Tumbug's concepts are more general than CD theory's concepts, therefore Tumbug diagrams can describe CD theory concepts without too much difficulty, whereas CD concepts usually cannot describe Tumbug concepts. In this section the CD theory Primitive Acts are listed in order of the simplest to the most complicated, for ease of presentation.

1. SPEAK

In CD theory, "SPEAK" is "The act of producing sound, including non-communicative sounds" (Lytinen 1992, p. 52), even if from lower animals and not intended as communication. Tumbug would generalize "animal" to any entity, such as a robot. Since the action of speaking does not imply that the listener is paying attention, an Attend Ring is not needed in the Tumbug diagram. However, since Tumbug's data Motion Arrows do not specify what sensory modality is being used, Tumbug would need to place a constraint on

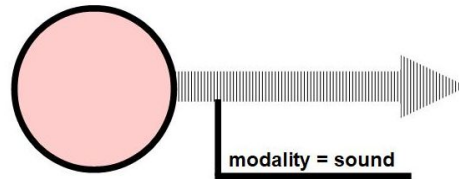


Figure 192: SPEAK: Information emission by sound modality, regardless of the emitter or reason for the emission.

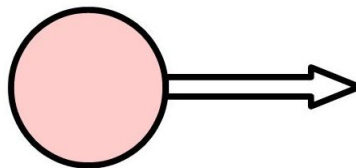


Figure 193: PTRANS, intransitive meaning: An object moves itself to an undisclosed location.

the transmission such as "modality = sound" on its data Motion Arrow. See Figure 192.

2. PTRANS

In CD theory, "PTRANS" is "The transfer of location of an object" (Lytinen 1992, p. 52). "PTRANS" means physical transfer, whether in transitive meaning (e.g., "He moved the radio farther away") or intransitive meaning (e.g., "He moved farther away from the radio"). Tumbug makes a big distinction between transitive and intransitive meanings because an additional object is involved in the transitive meaning, and in Tumbug all significant objects must be diagrammed. Therefore the CD theory meaning has two possible representations in Tumbug, depending on the type of transitivity intended.

2.1. Intransitive

See Figure 193.

2.2. Transitive

See Figure 194.

Tumbug minor convention: Motion Arrows may be used in diagrams without time or space reference, even when chained together, unless this results in ambiguity about the timing or cause-and-effect of the motions.

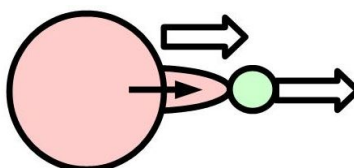


Figure 194: PTRANS, transitive meaning: An object moves another object to an undisclosed location.

3. MOVE

In CD theory, "MOVE" is "The movement of a body part of an agent by that agent" (Lytinen 1992, p. 52). This Primitive Act is slightly unusual in that the moved body part would be considered a direct object, which means that the subject is regarding its own body part as a direct object.

Whenever anything moves, Tumbug represents this situation with a timeline on the side of the diagram, spanning from the starting position to the ending position of the moved object. If the direction and location of the motion is not important, then a Location Box is not needed, otherwise if either attribute is important then a Location Box should be included.

An object repeatedly shown along a timeline is assumed to be undergoing continuous (as opposed to discrete) modification in time, so that the diagrams along the timeline can be considered only snapshots in time. If practical, one snapshot should be included for each qualitative change in the situation so that each diagram can be interpreted as one phase of the overall process.

For the situation of a body part being moved by its connected body, the body as a whole will ordinarily remain stationary except for that moved body part. The body would ordinarily be represented as a circle, and the body part would ordinarily be represented as a circle or ellipse, whichever most closely resembles the body part. If the moved body part is external (such as with an arm), the appendage will protrude on the outside of the body (circle); for an internal body part (such as the larynx), the appendage will protrude toward the inside of the body (circle). See Figure 195. Appendages and internal body parts could theoretically be represented with fuzzy regions instead of crisp ellipses, including details such as fuzzy fingers on fuzzy hands on fuzzy arms, but this type of representation is likely more difficult to code as a computer program, and is not in the spirit of traditional, crisp, visual computer simulations.

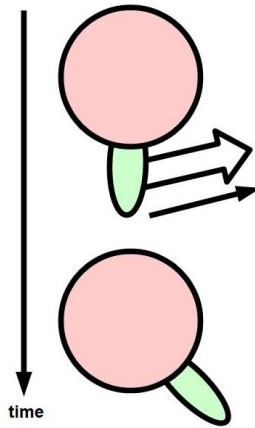


Figure 195: MOVE: An external body part being moved by its own body, where direction and location are not important.

4. GRASP

In CD theory, "GRASP" is "The grasping of an object by an actor so that it may be manipulated" (Lytinen 1992, p. 52). In this situation some kind of end effector (such as a hand) is needed, which is a level of detail not usually included in Tumbug, which usually shows only the equivalent of an arm, if that much. See Figure 196.

5. PROPEL

In CD theory, "PROPEL" is "The application of a physical force to an object" (Lytinen 1992, p. 52). This is a situation that Tumbug would represent by a Force Arrow directed toward a C Object Circle. The result of the force may or may not cause the direct object to move, however: the PROPEL definition does not specify. If the object moves then per Lytinen the PROPEL scenario has turned into a PTRANS scenario, otherwise the scenario is only PROPEL. Also unspecified by the PROPEL definition is whether some subject is causing the force. The PROPEL diagrams in this section assume a subject is applying the force, which is the most common situation, but by definition this is not strictly necessary.

5.1. With no resulting motion

See Figure 197.

5.2. With resulting motion

See Figure 198.

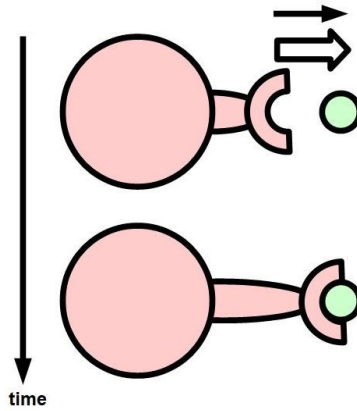


Figure 196: GRASP: A bodily appendage with an end effector extends to hold an object firmly.

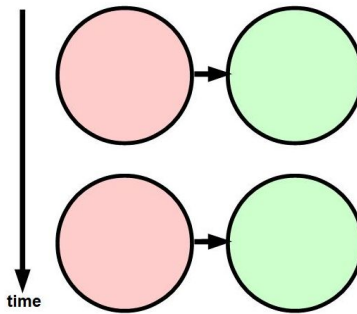


Figure 197: PROPEL: An object acted upon by force created by agent, but the direct object is not moved as a result.

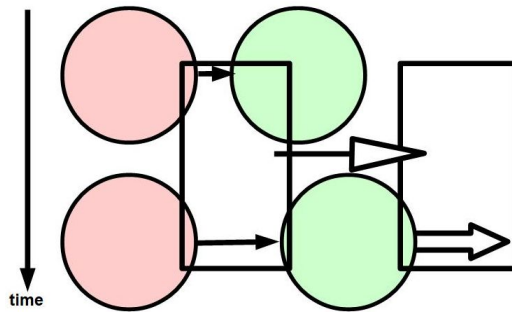


Figure 198: PROPEL: An object acted upon by force created by agent, and the direct object being moved as a result.

6. INGEST

In CD theory, "INGEST" is "The taking in of an object (food, air, water, etc.) by an animal" (Lytinen 1992, p. 52). In Figure 199 the ingestion process has been divided into three phases: (1) the object is outside the body, (2) the object is at the surface of the body, (3) the object is inside the body. If the opening to the body were important during such a process (such as a mouth) then a region of circle could be delineated to represent a mouth. Note in Figure 199 that the length of the Motion Arrow decreases in time to show that the object is slowing when entering the body, then comes to a complete stop for a while after reaching some (possibly temporary) destination within the body (such as a stomach).

7. EXPEL

In CD theory, "EXPEL" is "The expulsion of an object by an animal" (Lytinen 1992, p. 52). EXPEL is the same as INGEST except that the object motion is in the opposite direction, from within the body to outside the body. See Figure 200.

8. MTRANS

In CD theory, "MTRANS" is "The transfer of mental information between agents" (Lytinen 1992, p. 52). To show this in Tumbug the diagram needs two C Object Circles, one for each agent, a data Motion Arrow to show the information transfer, and one Attend Ring to show that the receiver is paying attention to the transferred information. See Figure 201. The reason the receiving agent is not color-coded in this diagram is because the MTRANS concept does not specify whether the message is the direct

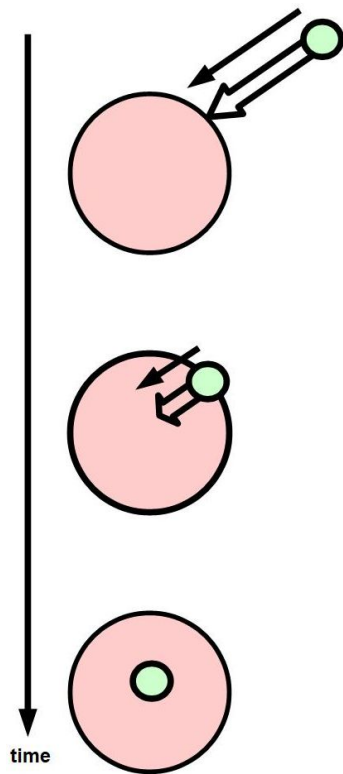


Figure 199: INGEST: An external object being moved from outside a body to inside that body.

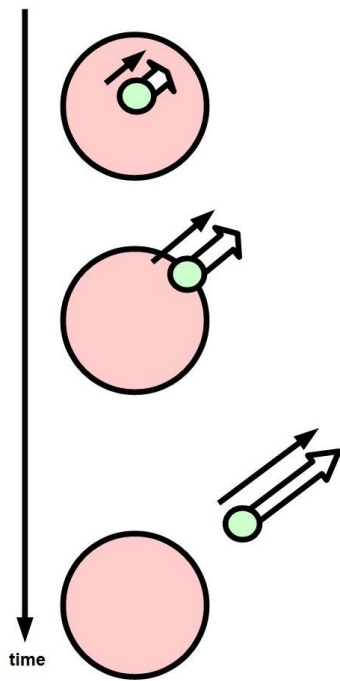


Figure 200: EXPEL: An external object being moved from inside a body to outside that body.

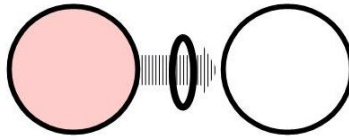


Figure 201: MTRANS: Information transmitted from the agent on the left, and received by the agent on the right.

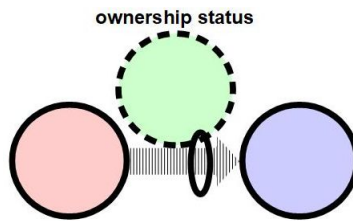


Figure 202: ATRANS: Information is transmitted from the agent on the left, and received by the agent on the right.

object (as in "He gave the information to his boss.") or whether the receiver is the direct object (as in "He informed his boss.").

9. ATRANS

In CD theory, "ATRANS" is "The transfer of ownership, possession, or control of an object" (Lytinen 1992, p. 52). Examples in Lytinen imply that the verbs "to give" and "to pay for" are involve ATRANS, and that the ATRANS operation is described on a link independent of (and parallel to?) the link that describes the physical transfer (if any). Schank probably intended this ATRANS definition to mean "full" transfer, as opposed to partial transfer, although in theory a second person could be placed on an ownership certificate or given co-managerial control over a group of employees. He we assume that full transfer is intended.

The simplest way to describe the ATRANS event with Tumbug is to treat ownership status as a data Object, and to use the shorthand notation for transfer of an object between two entities, as shown in Figure 202. More complicated schemes involving acknowledgement via message passing are also possible.

A more comprehensive view of an ATRANS would involve the change of legal ownership status, though, not just the physical transfer. This is

relatively simple to include, by including a state diagram of legal ownership status that is associated with the physical transfer operation, as shown in Figure 203.

A larger view of the State Diagram that is embedded in the comprehensive ATRANS view is shown in Figure 204.

10. ATTEND

In CD theory, "ATTEND" is "The act of focusing attention of a sense organ toward an object" (Lytinen 1992, p. 52). Tumbug uses exactly this same concept with the same name, and represents it by drawing an Attend Ring around the information stream arrow on which the receiver is focusing attention. Although Tumbug can represent a sense organ on the object to satisfy this definition more closely, it can probably be assumed that if the object is focusing its attention on an information stream then the object is capable of sensing it. See Figure 205.

11. MBUILD

In CD theory, "MBUILD" is "The construction of a thought or of new information by an agent" (Lytinen 1992, p. 52). Note that this MBUILD definition is ambiguous as to whether "construction" means created from scratch, or rather created from other objects. Both cases are considered below, and are handled easily by Tumbug.

11.1. Creation from scratch

See Figure 206.

11.2. Creation from combination

See Figure 207.

14.1.4 Overview of comparison

1. Type of Primitive Acts

CD theory is said to have failed because its small number of Primitive Acts could not possibly cover everything in the real world that might need to be represented (Lytinen 1992, p. 51). Tumbug also has a relatively small number of Building Blocks but many of these Building Blocks are based on physics, not on human-oriented issues such as ingestion or thinking, and since physics is the ultimate study of the physical universe, physics can presumably represent anything of interest, therefore Tumbug's Building Blocks almost cannot fail at representing the real world. Admittedly, for complicated situations Tumbug descriptions can become unwieldy, but that is why Tumbug labels exist: as a convenient shorthand notation for complicated

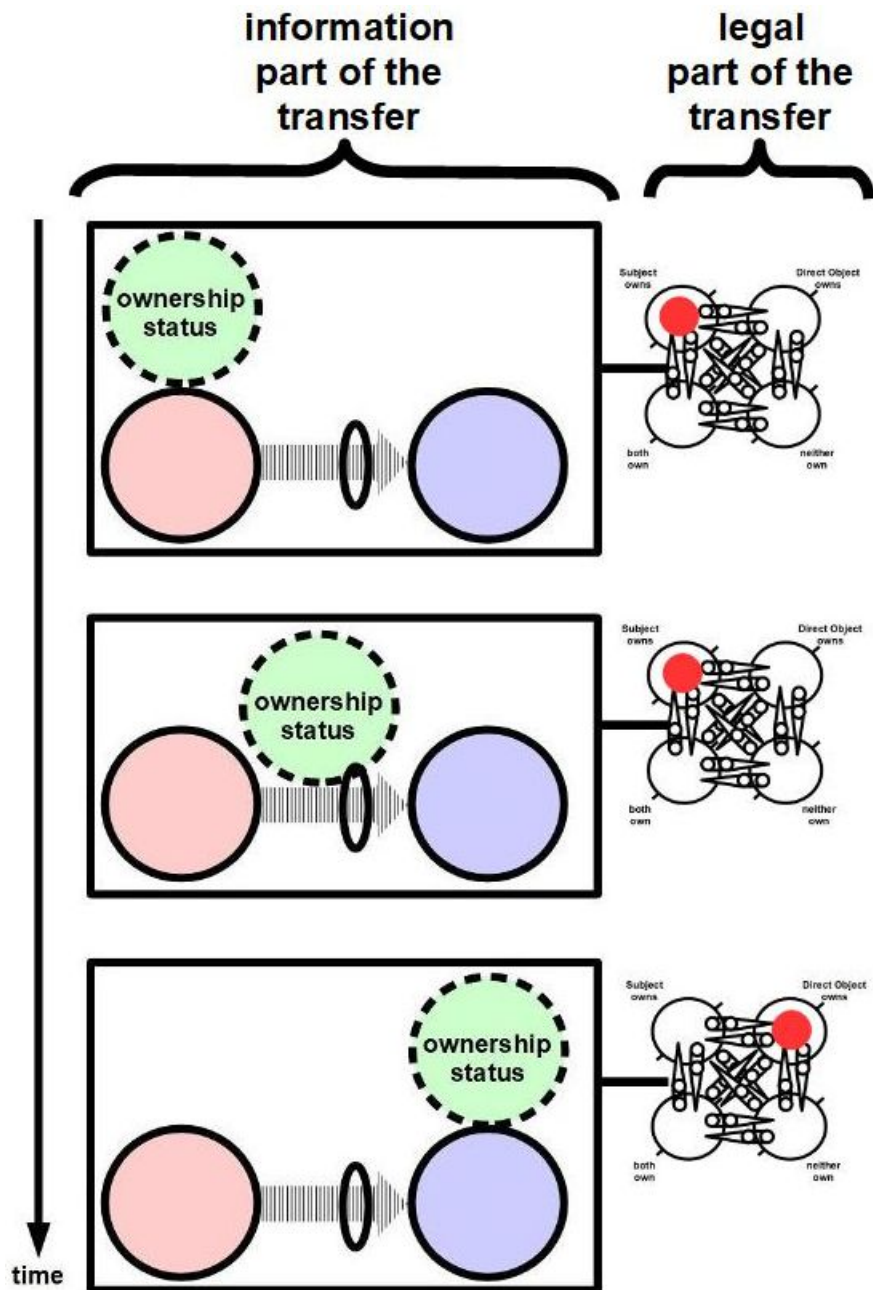


Figure 203: A more comprehensive view of an ATRANS operation would show both information transfer and legal transfer.

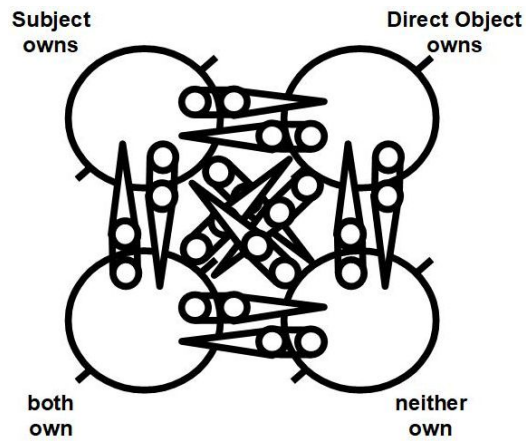


Figure 204: A State Diagram that shows all possibilities of legal ownership between two people.

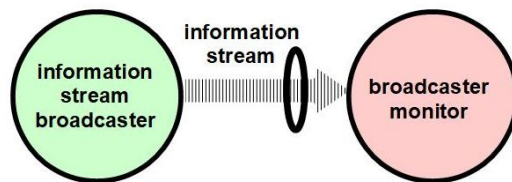


Figure 205: ATTEND: The broadcaster monitor attends to an information stream from the broadcaster.

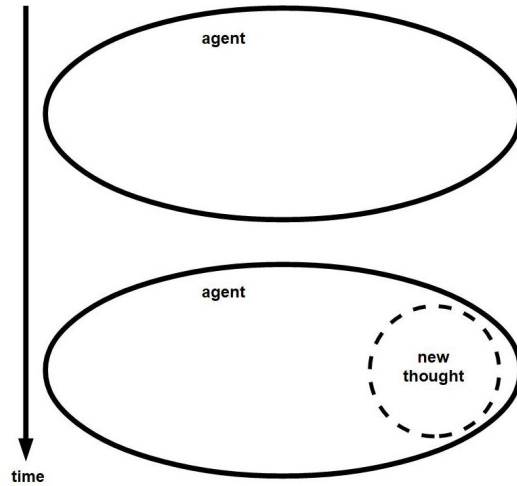


Figure 206: MBUILD: One new thought forms spontaneously, from scratch.

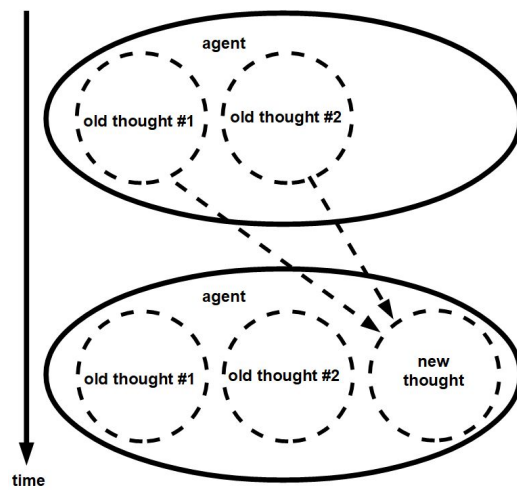


Figure 207: MBUILD: One new thought forms from the combination of two old thoughts.

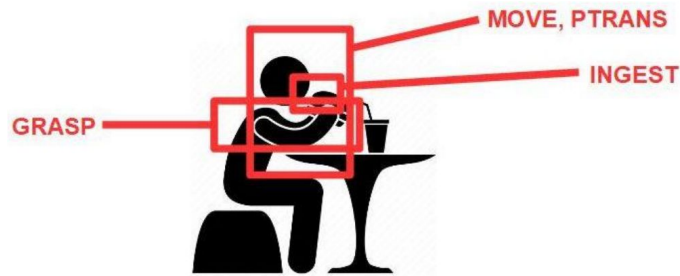


Figure 208: CD theory uses Primitive Acts that are human-oriented and are best at describing complicated actions.

descriptions or diagrams, the same shorthand that nearly every AI system already uses, such as rule-based expert systems and semantic networks.

Figure 208 and Figure 209 might illustrate more clearly how CD theory and Tumbug compare. The basic diagram in the form of black-and-white icons showing a seated person eating a hamburger is the same for each, but the KRM influences how the information in the diagram is divided. Red text is used to label categories according to the KRM being used.

Note that in this case CD theory is excellent for describing the human-oriented actions of grasping and eating, but many other actions such as sitting or placing one's body partially around an object (e.g., a table) could not be included unless a huge number of additional Primitive Acts were used. A huge drawback of CD theory is not obvious from the ingestion figure unless one remembers that CD theory *per se* cannot show diagrams at all, or even spatial relationships. In CD theory only rigidly formatted textual descriptions would be used to describe what a user can so easily see in a diagram.

One of the main CD Theory terms is "Primitive Conceptual Categories," which is also called "conceptual primitives," "conceptual dependency primitives," or "instrumental conceptualizations," depending on the author. The list of these Primitive Conceptual Categories is {PP, ACT, PA, AA, LOC, T}, and this is the closest set of concepts to Tumbug's icons that are called "Building Blocks," whose term is based on CD Theory terminology. The other main CD Theory term is "ACT," which is also called "Primitive Acts." The list of these Primitive Acts is {ATRANS, PTRANS, PROPEL, MTRANS, MBUILD, SPEAK, ATTEND, MOVE, GRASP, INGEST, EX-

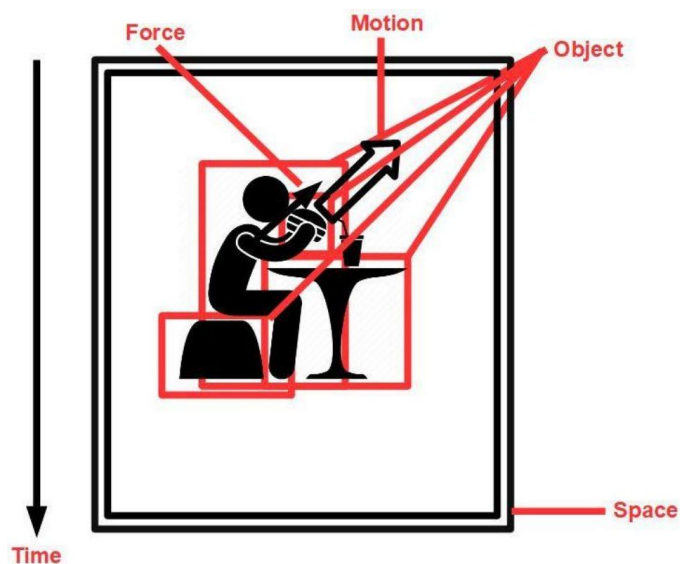


Figure 209: Tumbug uses Building Blocks that are physics-oriented and are best at describing simple objects and actions.

PEL}, so these components are instantiations of the single Primitive Conceptual Category called "ACT." In other words, both CD theory and Tumbug use the term "building blocks" to include both major sets of CD Theory, namely the Primitive Acts and the specific acts of ACT, though in this document "building blocks" (uncapitalized) is used for CD theory and "Building Blocks" (capitalized) is used for Tumbug.

Note that in this case Tumbug spans everything that could possibly be going on in the diagram, from the objects to the forces acting on them to their resulting motions (i.e., Tumbug is "more universal." In mathematical terminology, Tumbug "spans" a larger space, but since Tumbug labels are simpler in concept, a larger quantity of the same Tumbug labels will usually be needed. More labels and boxes will tend to clutter the diagram. Also, the forces and motions involved (i.e., using arms to move the hamburger) are small in comparison to the overall size of the diagram, so those components may be difficult to see in larger or more complicated events. (This is why the Zoom ability of Tumbug was introduced: to allow arbitrary zoom in of smaller objects, at the user's discretion.) The overwhelming visual advantage of Tumbug is not seen in such a static icon, however: Tumbug can literally

show realistic motion of the objects, like a simulation, whereas CD theory has no mechanism at all to allow even one moving object to be displayed.

Some similarities between CD theory and Tumbug:

- Both use around 1-2 dozen building blocks.
- Some of the building blocks are identical, and those that are not identical are similar.
- All slots and Primitive Acts in CD theory can be represented by Tumbug.
- Both use aggregation.
- Both use abstract objects, and treat them slightly differently from concrete objects.

Some differences between CD theory and Tumbug:

- Tumbug is based largely on physics and math; CD theory is based on human-oriented actions.
- Tumbug is almost completely visual; CD theory is text-based.
- Tumbug can display images, even moving images; CD theory cannot display images at all.
- Tumbug can represent phrases; CD theory needs a complete sentence with actor and action.
- CD theory Primitive Acts cannot cover every possibility; Tumbug Building Blocks so far appear to be able to do so, at least for WS150 problems.
- CD theory requires separate text to describe TO versus FROM; Tumbug needs no such distinction, other than arrow placement.
- CD theory distinguishes static objects (PP) from Primitive Acts (ACT) on those objects, though it is possible in Tumbug to define an object so that it changes state or behavior via an internal Correlation Box without dictating those changes explicitly. (This Tumbug capability is not described in this document, however.)

CD theory primitive: equivalent Tumbug diagrams and one sample characteristic of each:















- | | | |
|------------|---|--|
| 1. ATRANS |  | (1) transfer of a data Object from one person to another |
| 2. ATTEND |  | (1) Attend Ring on the data stream being monitored |
| 3. EXPEL |  | (1) motion of an object from inside a body to outside the body |
| 4. GRASP |  | (1) an end effector on a body holds an object firmly |
| 5. INGEST |  | (1) motion of an object from outside a body to inside the body |
| 6. MBUILD |  | (1) creation from scratch |
| |  | (2) creation from combination |
| 7. MOVE |  | (1) an object moving its own attached appendage |
| 8. MTRANS |  | (1) information stream from one object to another, with recipient monitoring |
| 9. PROPEL |  | (1) with no resulting motion |
| |  | (2) with resulting motion |
| 10. PTRANS |  | (1) intransitive: an object moves itself |
| |  | (2) transitive: an object moves another object |
| 11. SPEAK |  | (1) information stream from one object to another, sound modality |

Figure 210: 73% of CD theory Primitive Acts map one-to-one to an equivalent Tumbug concept.

- Tumbug’s Building Blocks generalize to a small set of (five) Basic Building Blocks, but CD theory does not note any such generalizations for its Primitive Acts (ACT).

To a great extent, Tumbug appears to be what CD theory was attempting, but Roger Schank used non-universal building blocks instead of considering physics. Similarly, although not discussed in this document, Tumbug appears to be what formal language theory was attempting, but Noam Chomsky used traditional text instead of images. Tumbug potentially infuses new life into these earlier milestone ideas, so perhaps Tumbug also has a chance of becoming a milestone.

2. Nearness to a one-to-one mapping to Tumbug concepts

As seen in Figure 210, CD theory concepts are defined in a way that typically are represented by only one Tumbug concept. This is mostly the result of relatively clear-cut definitions of CD theory’s Primitive Acts, not necessarily due to a similar foundation of building blocks between the two systems. $8/11 = 73\%$ of CD theory Primitive Acts map one-to-one to a Tumbug concept.

<u>This document's</u>		
<u>term:</u>	<u>Kolln's term:</u>	<u>Kolln's description:</u>
Pattern #1	Pattern I	NP be ADV/TP
Pattern #2	Pattern II	NP be ADJ
Pattern #3	Pattern III	NP1 be NP2
Pattern #4	Pattern IV	NP linking verb ADJ
Pattern #5	Pattern V	NP1 linking verb NP1
Pattern #6	Pattern VI	NP intransitive verb
Pattern #7	Pattern VII	NP1 transitive verb NP2
Pattern #8	Pattern VIII	NP1 transitive verb NP2 NP3
Pattern #9	Pattern IX	NP1 transitive verb NP2 ADJ
Pattern #10	Pattern X	NP1 transitive verb NP2 NP2

NP = noun phrase
be = "to be" verb
ADV/TP = adverbial of time or place
ADJ = adjectival

Figure 211: Kolln's sentence patterns with the order and abbreviations used by Kolln.

14.2 English grammar representation

14.2.1 Kolln sentence patterns

Martha Kolln listed 10 basic English sentence patterns that she claims constitute the vast majority of sentence patterns that appear in English (Kolln and Funk 2006). Kolln's 10 patterns are shown in Figure 211. Kolln numbered these by Roman numerals, whereas this document numbered these by Arabic numerals.

Each of these patterns is described with some sample sentences converted to Tumbug in the following sections.

It must be understood that Kolln's statement applies to (1) Only basic sentence patterns, since sentences of different types could always be combined via conjunctions to create a much larger set of types through combinations of the basic types, (2) Only statement versions of the sentence patterns, since question versions of these patterns may alter the word order; (3) Non-interjections, since otherwise an interjection consisting of a single word would

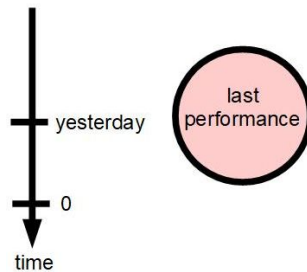


Figure 212: [Kolln Pattern #1] Tumbug for "The last performance was yesterday (ADV/T)."

qualify as having correct grammar, which it does not.

The standard sentence diagram that was widely taught in the 1960s and 1970s, formally called the Kellogg-Reed system, is much better known than the Kolln patterns. However, the formal documentation of the Kellogg-Reed diagrams with respect to possible combinations of values within the slots for subject complement (SC), direct object (DO), indirect object (IO), and object complement (OC) is so weak that the Kellogg-Reed system is ignored here, at least for comparison purposes.

1. NP be ADV/TP

Both cases are of ADV/TP are shown next, one where the adverbial is of time (ADV/T), one where the adverbial is of place (ADV/P).

1.1. With Time Arrow (ADV/T)

See Figure 212.

1.2. With Location Box (ADV/P)

See Figure 213.

2. NP be ADJ

This Kolln pattern equates only to an object (O) having an attribute (A) with a value (V), which Tumbug renders in OAV fashion as shown in Figure 214.

3. NP1 be NP2

3.1. As superset

This Kolln pattern demonstrates a principle mentioned by Roger Schank regarding the need for a canonical form of a sentence when sentence meaning is represented (Schank 1976, p. 172). In this case, there are two simple ways to express the same meaning of the sentence "The students are diligent.":

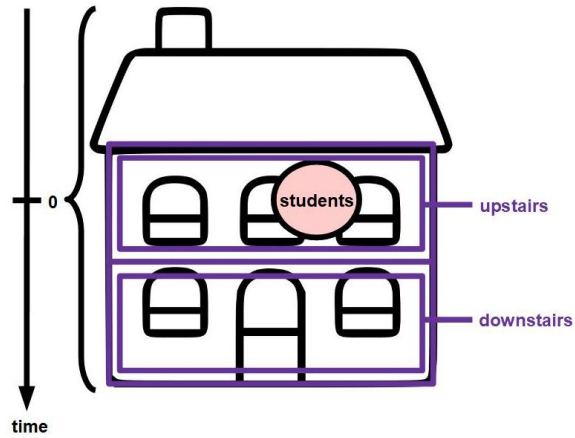


Figure 213: [Kolln Pattern #1] Tumbug for "The students are upstairs (ADV/P)."

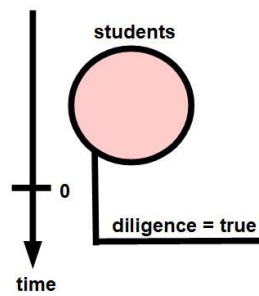


Figure 214: [Kolln Pattern #2] Tumbug for "The students are diligent."

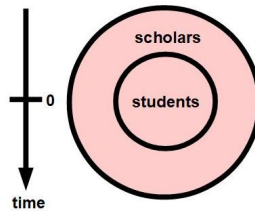


Figure 215: [Kolln Pattern #3] Tumbug for "The students are scholars."

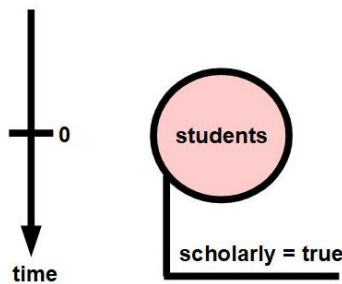


Figure 216: [Kolln Pattern #3] Tumbug for "The students are scholarly."

- "The students are scholars." - Involves two noun phrases (NP1 = students, NP2 = scholars). This sentence is represented in Tumbug as shown in Figure 215.
- "The students are scholarly." - Involves one noun phrase (NP1). and one attribute value (scholarly = true). This sentence is represented in Tumbug as shown in Figure 216.

One of the main goals of representation here is to use only one representation for the concept of "scholars/scholarly," which suggests that only one of these two diagrams should be declared the canonical form. Figure 217 shows more clearly how the two previous diagrams are related. (The last two diagrams are essentially the same diagram, but one has the common attribute "scholarly = true" duplicated, the other does not.)

Currently the Tumbug convention used is to use the superset representation. A similar sentence they has two representation options is diagrammed in Figure 218.

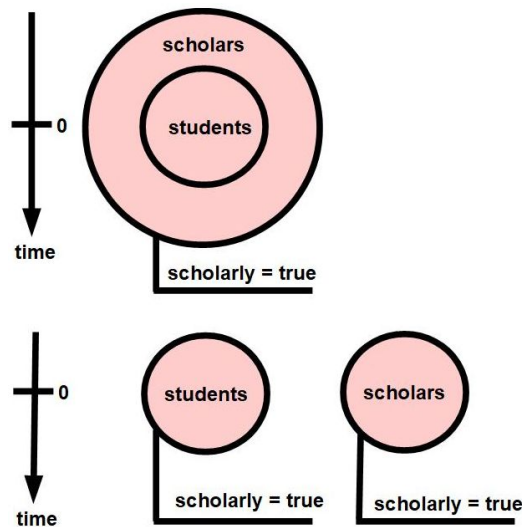


Figure 217: [Kolln Pattern #3] Two Tumbug ways to represent the concept that "The students are scholarly."

Note: The use of the value "true" in the aforementioned expression "scholarly = true" is a quick way to conform to OAV notational convention without need for additional thought, because the attribute-value form is required to be " $\langle \text{attribute} \rangle = \langle \text{value} \rangle$ ", which the expression "scholarly" alone would not fulfill, and to think of which attribute would allow " $\langle \text{attribute} \rangle = \text{scholarly}$ " to make sense (e.g., "learnedness = scholarly", or "academic character = scholarly") is more difficult than merely writing "scholarly = true."

3.2. With removed possessive adjective

Another example said to be Kolln Pattern #3 is "Professor Mendez is my math teacher" (Kolln and Funk 2006, p. 31). This is another example of the inaccuracies of the English language, since the verb "to be" (conjugated here as "is") has a different meaning than before. Here it would be wasted effort to create a set called "my math teachers" and to put "Professor Mendez" as a single instance of that set, since usually a student has only one math teacher. Therefore "is" does not mean "is a subset of" here. In this case the underlying problem is use of a possessive adjective, which should be removed from a sentence before converting the sentence to Tumbug. There exist multiple ways to represent this sentence without the possessive adjective, one

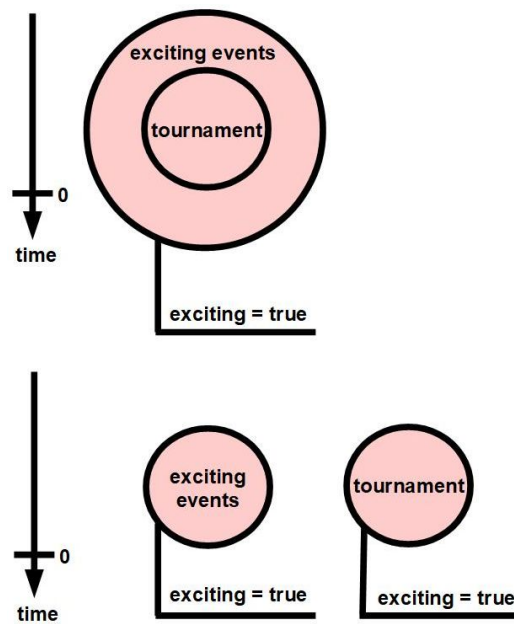


Figure 218: [Kolln Pattern #3] Two Tumbug ways to represent the concept that "The tournament was an exciting event." Top diagram: One attribute applies to both the set and superset. Bottom diagram: No superset/subset structure, but note that the same attribute appears in both sets.

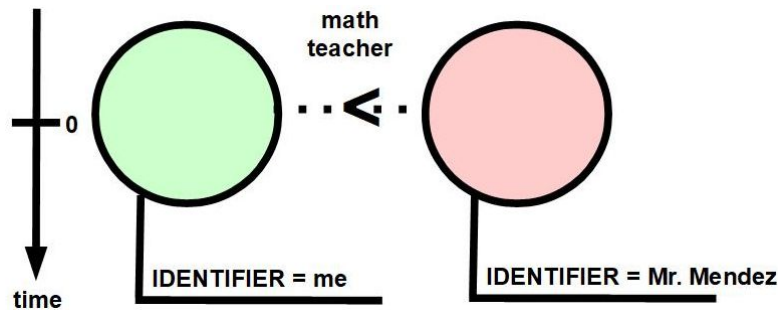


Figure 219: [Kolln Pattern #3] Tumbug for "Professor Mendez is my math teacher." = "Professor Mendez IS-A math teacher OF me."

of which is shown in Figure 219. This is an indication of weak organizational criteria used in defining the Kolln patterns.

4. NP linking verb ADJ

4.1. With propositional attitude

This is another Kolln pattern that incorporates very different Tumbug structures if the verb changes slightly. Many Kolln Pattern #4 verbs involve general, average impressions, such as "to seem" and "to look," others involve specific sensory perceptions such as "to taste," "to smell," "to sound," and others use neither, such as "to grow" (meaning "to become"), which has a very different meaning: a single attribute that changes over time, which has nothing to do with impressions or sensory modalities. Therefore this Kolln category is not taken very seriously with respect to Tumbug, other than as a supply of possibly unusual examples to test against Tumbug representation.

Figure 220 shows an example of Kolln Pattern #4 when the linking verb is a propositional attitude, in this case "seem." English allows an ambiguity here, regarding the entity (or group) who is interpreting the diligence of the student. The only logical way to represent this situation is to show an Object Circle that is unlabeled, as shown in the figure. To show a single person icon would assume that only one person is doing the interpreting, to show an "average person" icon would assume that the interpreter is average, to show a person icon at all would assume that the interpreter is human, and to show a "DK" (= Don't Know) wildcard would assume that the reader cares, which might entail extra, unnecessary, ongoing work in labeling, especially on a very large network of connected Tumbug icons that no one is likely to care about.

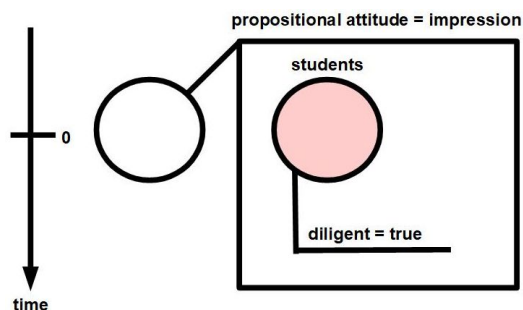


Figure 220: [Kolln Pattern #4] Tumbug for "The students seem diligent."

The term "propositional attitude" is from Davis 1990 (ch. 8), and the need for this concept occurs in three of the Kolln patterns (Pattern #4, Pattern #9, Pattern #10). It is a very useful concept that unifies several modalities of knowledge. For details, see the section on propositional attitudes in this document.

4.2. With attribute changing over time

An example of a Pattern #4 example that has a very different meaning and Tumbug diagram is in Figure 221. In this case the only thing happening is that an attribute value changes in time, which has nothing to do with sensory input or overall impressions.

4.3. With sensory perceptions

Figure 222 illustrates another example of a propositional attitude, in this case gustatory taste, which generalizes to any type of sensory perception.

5. NP1 linking verb NP1

5.1. As superset

See Figure 223.

5.2. With propositional attitude

See Figure 224.

5.3. With object changing over time

Kolln does not document the possibility of an object changing into another object over time. The sentence "The caterpillar became a butterfly" uses the same verb and tense as before (viz., "became") in the sentence "The students became scholars" but in this case "became" implies the object itself has changed, not just its classification. This is an example of how the Kolln patterns are based too much on specific verb categories rather than on the

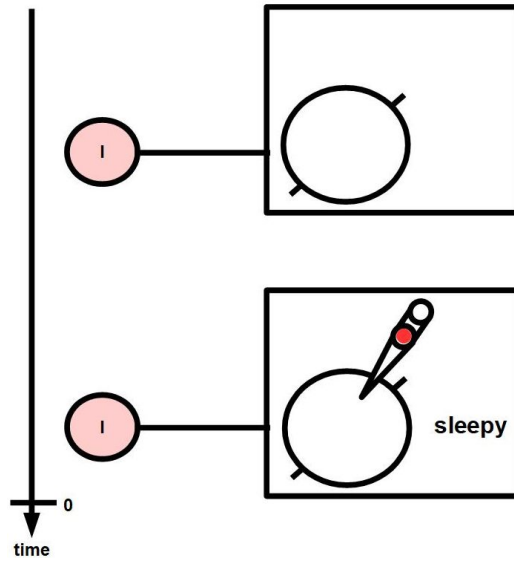


Figure 221: [Kolln Pattern #4] Tumbug for "I grew sleepy." = "I transitioned into the sleepy state."

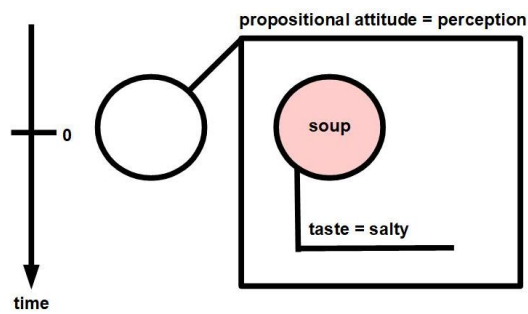


Figure 222: [Kolln Pattern #4] Tumbug for "The soup tastes salty."

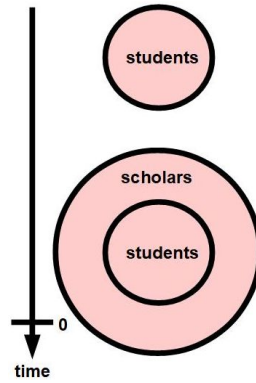


Figure 223: [Kolln Pattern #5] Tumbug for "The students became scholars."

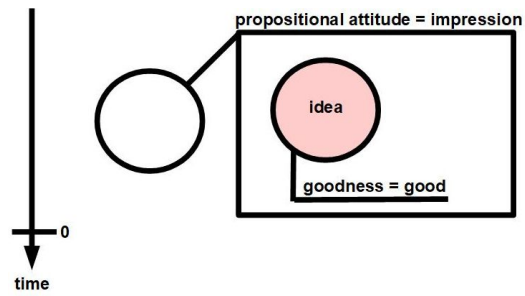


Figure 224: [Kolln Pattern #5] Tumbug for "That seemed a good idea."

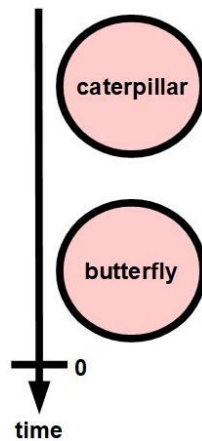


Figure 225: [Kolln Pattern #5] Tumbug for "The caterpillar became a butterfly."

meanings of those words in different contexts.

In contrast, Tumbug can easily represent an unmoving object changing into another unmoving object in the same location by placing the two objects above each other along the timeline, with the older object at an earlier time than the newer object. See Figure 225. Since two solid objects cannot occupy the same space in our physical world, such representation is sufficient, *pro tempore*.

This simplicity comes at a cost of increased risk when creating diagrams, however: Care must be taken not to place two different objects above each other in a timeline unless one has moved out of the way completely. See Figure 226.

6. NP intransitive verb

Intransitive verbs are another category of verb that is very poor for capturing a general meaning of such a verb, as evidenced by the two examples below that use this same sentence pattern and verb class but have two very different meanings and Tumbug diagrams.

6.1. With states

Note that in the Kolln example "The students rested" (Kolln and Funk 2006, p. 35) the verb "rested" is ambiguous as to whether the rest was physical or cognitive. Fortunately, a general state diagram such as one for "exertion versus rest" (as shown in Figure 227) can subsume both possibilities

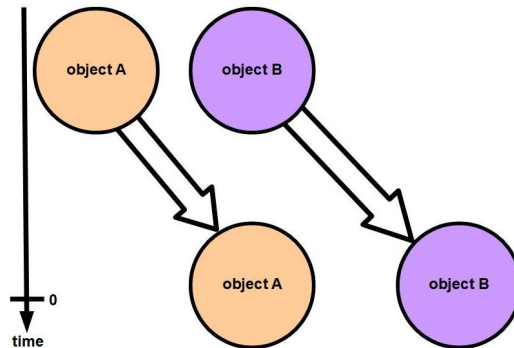


Figure 226: In the real world of physical objects, Object A may occupy Object B's space later in time only if Object B has moved out of the way.

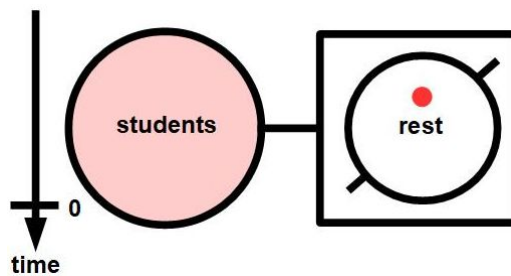


Figure 227: [Kolln Pattern #5] Tumbug for "The students rested." = "The students were in rest state."

at once. Note also that the sentence given does not necessarily state that the state of the students before the rest state was the exertion state: the students could have just come from an earlier break, then decided to take a later break. Therefore only the later specified state ("rest") is marked with an asterisk as known; the state diagram exists in the earlier depiction, but no state is marked. This detail is very important to note when developing a matching algorithm for Tumbug.

6.2. With motion

Intransitive verbs can also refer to motion, as in the verb "to arrive" as shown in Figure 228.

7. NP1 transitive verb NP2

Even Martha Kolln herself makes a heavy admission about the nature

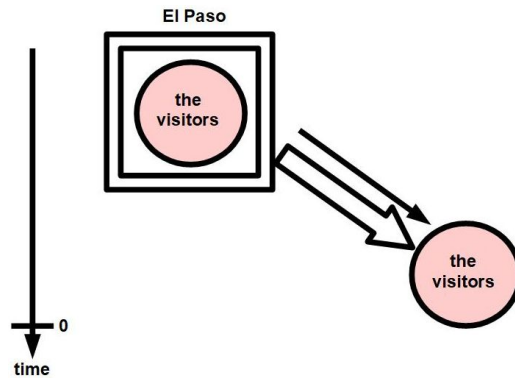


Figure 228: [Kolln Pattern #6] Tumbug for "The visitors from El Paso arrived."

of transitive verbs when discussing her sentence Pattern #7 (= Pattern VII) of hers (Kolln and Funk 2006, p. 39):

"Traditionally, we think of the transitive verb as an action word: Its subject is considered the doer and its object the receiver of the action. In many Pattern VII sentences this meaning-based definition applies fairly accurately. In our Pattern VII sample sentences, for instance, we can think of their assignment as the receiver of the action studied and a home run as a receiver of the action hit. But sometimes the idea of receiver of the action doesn't apply at all:

Our team won the game.
 We enjoyed the game.

It hardly seems accurate to say that game "receives the action."

This equates to at a deep insight about verbs in all natural languages: Verbs do not generalize to a single concept (e.g., not motion, action, or relationship), which suggests the modern concept of a "verb" is flawed. The term "verb" is overloaded with the attempt to include references to attribute values, supersets, actions upon a target object, sensory input to the actor, states, cause and effect, and possibly more. The Tumbug viewpoint explains this problem as poor generalizations of language, leading to a poor knowledge representation system for language (such as the Kellogg-Reed system

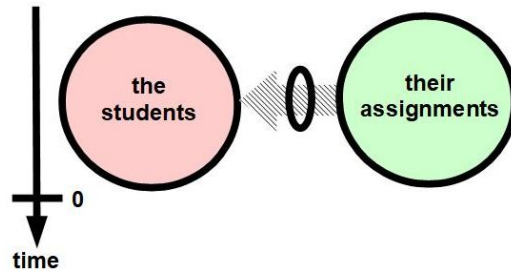


Figure 229: [Kolln Pattern #7] Tumbug for "The students studied their assignments."

or modern tree diagrams of sentences).

7.1. With data transfer

Sometimes Pattern #7 applies to data transfer and/or perception, such as in Figure 229.

7.2. With motion and contact

The most obvious applicability of Pattern #7 is physical contact of the subject with the direct object, as in Figure 230.

7.3. With states

See Figure 231.

7.4. With emotions

See Figure 232. This figure combines what is essentially a State Diagram, namely the Robinson Icon, which in this case is in the positive version of the state of Event Related: Joy, elation, triumph, jubilation, which is a cathected emotion with respect to the direct object, namely the game.

8. NP1 transitive verb NP2 NP3

It appears that this Kolln pattern always maps only to the same Tumbug diagram of an object being transferred from subject to indirect object, as shown in Figure 233.

9. NP1 transitive verb NP2 ADJ

This Kolln sentence pattern maps to at least two different types of Tumbug diagram.

9.1. With propositional attitude

Having a propositional attitude about an object or person does not inherently involve a change of attribute in time. See the example in Figure 234.

9.2. With attribute changing over time

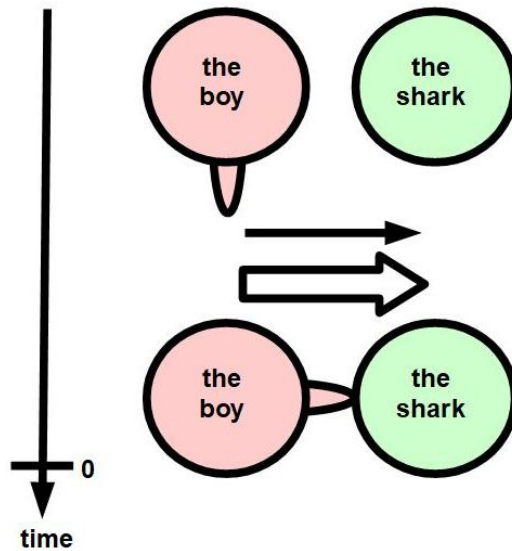


Figure 230: [Kolln Pattern #7] Tumbug for "The boy touched the shark."

An object or person can cause a change of attribute in time. See the example in Figure 235.

10. NP1 transitive verb NP2 NP2

This Kolln sentence pattern maps to at least two different types of Tumbug diagram.

10.1. With propositional attitude

See Figure 236.

10.2. With causation

See Figure 237.

11. Nearness to a one-to-one mapping to Tumbug concepts

Kolln's 10 sentence patterns fail to produce any reasonable approximation of 1-to-1 mapping with Tumbug concepts, as shown in Figure 238. Kolln's examples are useful, however, as a fairly comprehensive source of sample sentences to test against Tumbug's representation ability. If Kolln's 10 sentence patterns cover 95% of encountered English sentences, and if Tumbug can represent those same patterns, then Tumbug should also have at least 95% ability to represent English sentences. $2/10 = 20\%$ of Kolln patterns map one-to-one to Tumbug concepts.

12. The five simplest sentence patterns

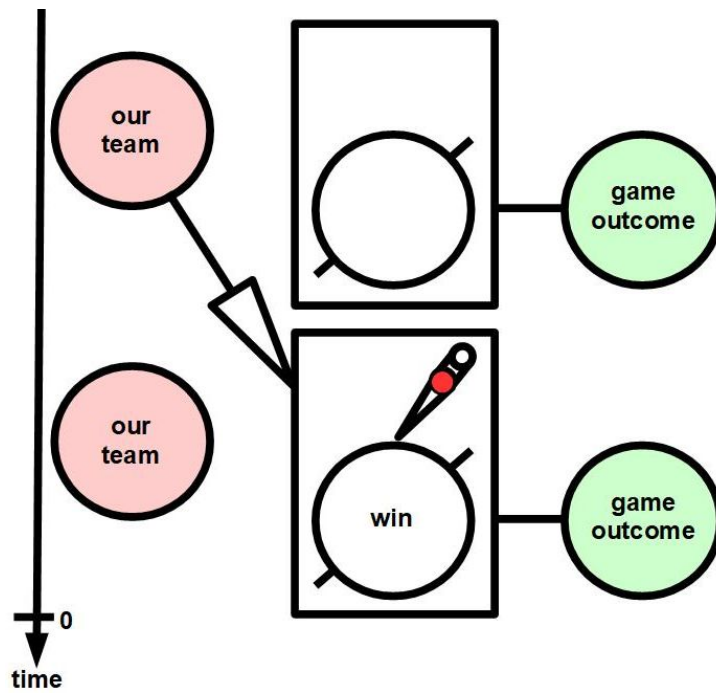


Figure 231: [Kolln Pattern #5] Tumbbug for "Our team won the game" = "Our team caused the game outcome to enter into the win state."

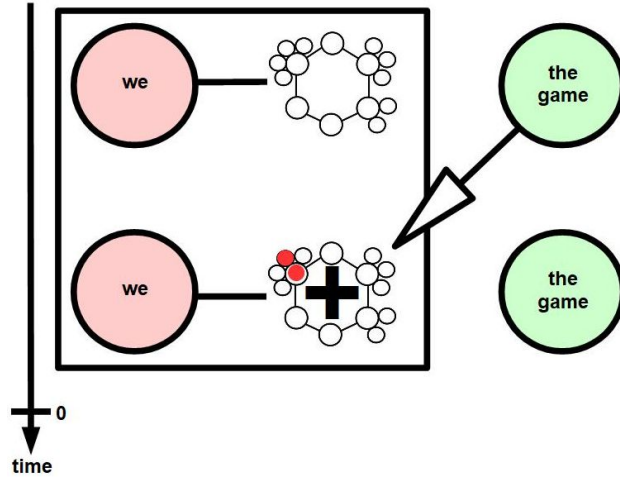


Figure 232: [Kolln Pattern #7] Tumbug for "We enjoyed the game." = "The game caused our emotion to enter into the joy state."

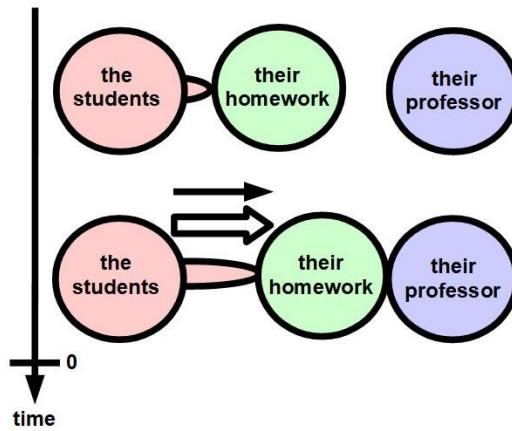


Figure 233: [Kolln Pattern #8] Tumbug for "The students gave their professor their homework," no shorthand diagram used.

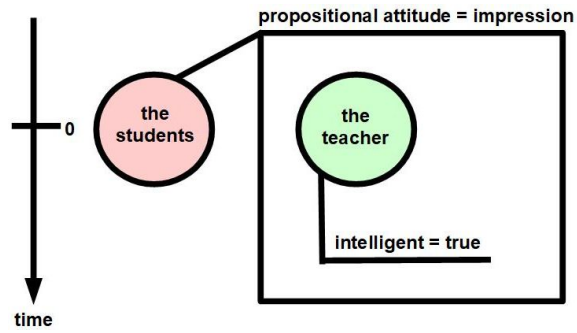


Figure 234: [Kolln Pattern #9] Tumbug for "The students consider the teacher intelligent."

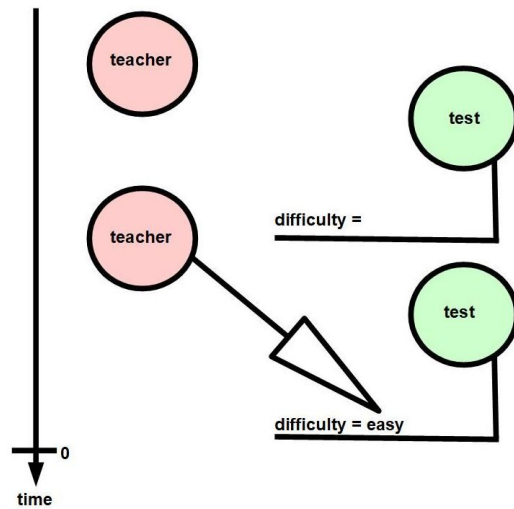


Figure 235: [Kolln Pattern #9] Tumbug for "The teacher made the test easy."

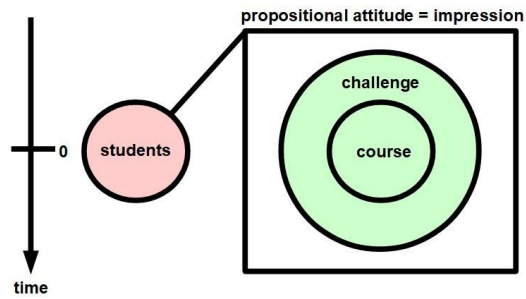


Figure 236: [Kolln Pattern #10] Tumbug for "The students consider the course a challenge."

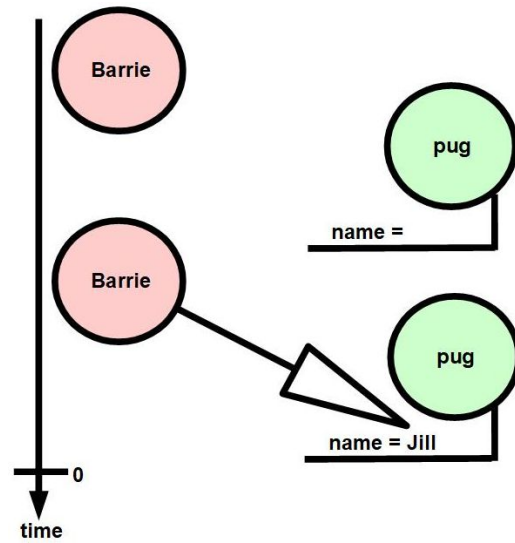


Figure 237: [Kolln Pattern #10] Tumbug for "Barrie named his pug Jill."

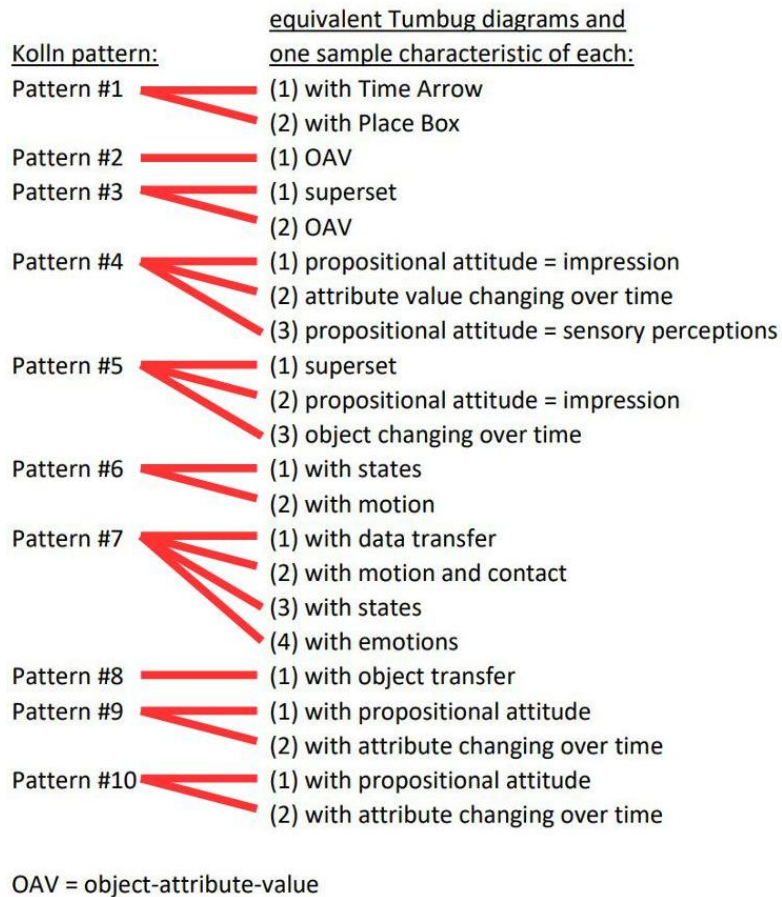


Figure 238: Only 20% of Kolln patterns map one-to-one to an equivalent Tumbug concept.

Note that the various examples of Pattern #6—"The students rested" versus "The visitors from El Paso arrived"—produce diagrams that are very different from one another. The second example describes the arrival of people in motion whereas the first example is very unclear what it describes, even as to whether any motion was involved before the action of "rested."

This is not the only Kolln grammatical pattern that is highly unpredictable. Pattern #7—"The students studied their assignments" versus "The boy touched the stingray"—has the same general problem. The second example has a direct object that receives the action whereas the first example has more the nature that the assignment was doing to action to the students.

These anomalies lead to a profound insight into all human language: All human language is structured around the highly inconsistent nature of verbs, rather than being extremely well-structured by basing all grammatical patterns on objects and object motions. The author suggests this is the primary reason why natural language has always been so difficult to describe with grammatical rules, and consequently why the field of AI has always had extra difficulty with natural language. Even a seemingly exotic language such as Japanese, which has particles, has the same two meanings of the verb "to be" as English does, and goes through the same distortions as English in order to force-fit such verbs into inefficient, inconsistent data structures.

Side conjecture: The Proto-Human language, if such a language ever existed, may have had a needlessly difficult grammar that failed to focus only on objects and motions, so all natural languages since then, no matter how seemingly different they are, may have carried the same flaw that forced descriptions of attribute and superset relationships to use unneeded verbs only because action verb sentences used a structure that required a verb.

This fundamental problem of human grammar does suggest one way to slightly improve foreign language learning, however: Teach grammar first with only moving objects, up to three in number, then only later move into more abstract verbs that are difficult to represent with moving objects, or with cause-and-effect substituted for motion. Notice that grammar nicely falls into only about five basic patterns if only the common patterns of motion are used.

Consider Pattern #6, Pattern #7, and Pattern #8. From a general standpoint, these few sentence patterns fall into a small, logical set of possible motions of objects:

Pattern #2: Subject does not move, has an attribute and value: 1 object +

0 motions

Pattern #3: Subject does not move, is part of a superset: 1 object + 0 motions

Pattern #6: Subject moves itself: 1 object + 1 motion

Pattern #7: Subject moves and makes contact with Direct Object: 2 objects + 1 motion

Pattern #8: Subject transfers Direct Object to Indirect Object: 3 objects + 1 motion

Pattern #1 can be chosen to be ignored, since it deals only with an attribute of space or time.

Pattern #4 can be chosen to be ignored, since it deals with a propositional attitude, which is indirect.

Pattern #5 can be chosen to be ignored, since it deals with a linking verb over time: complicated.

Pattern #9 can be chosen to be ignored, since it deals with a propositional attitude, which is indirect.

Pattern #10 can be chosen to be ignored, since it deals with a propositional attitude, which is indirect.

The summary of these observations is shown in Figure 239.

The above heuristics were a lot of generalization for the sole purpose of simplifying human grammar, but this practice could be justified if it sufficiently improves the understanding of grammar, and/or if the simpler patterns are more frequently encountered. The names of these most basic visual sentence patterns are given by the author in Figure 240.

An interesting and little-known fact is that Tumbug's Transfer pattern does not exist in Spanish in the same way that it exists in English. For example, the English sentence "The students gave their professor their homework" would need to be grammatically rearranged to a Spanish grammatical pattern with an extra preposition, such as "The students gave their homework to their professor" ("Los estudiantes le dieron sus tareas a su profesor"). This fact illustrates that slight variations in the number of sentence pattern types occur between major languages, but that Tumbug representations remains unchanged between such syntactical rearrangements.

Three of the most basic Tumbug patterns (E, C, T) match three of the English verb valency patterns listed by D.J. Allerton (Allerton 2006, pp. 170-171), which demonstrates that Tumbug's natural grammatical patterns

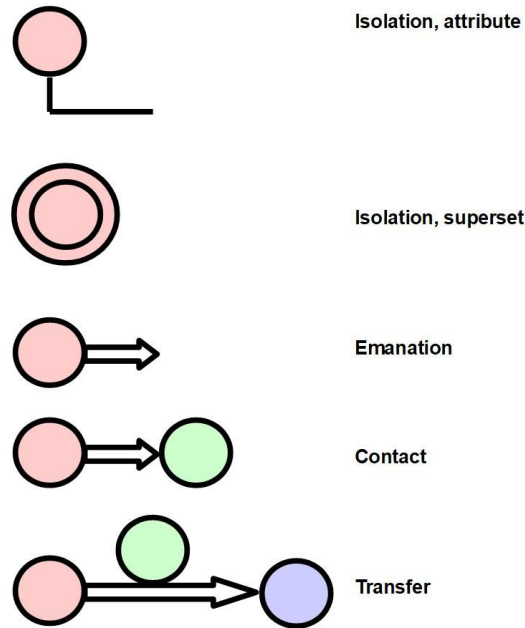


Figure 239: These five sentence patterns are the simplest, most basic, and by far the most common in English. These five patterns are the only patterns discussed in depth in this document.

<u>Tumbug pattern name:</u>	<u>parts of speech involved:</u>	<u>Motion involved?</u>
Isolation - Isolation, attribute - Isolation, superset	Subject	No motion
Emanation	Subject	Motion
Contact	Subject + Direct Object	Motion
Transfer	Subject + Direct Object + Indirect Object	Motion

Figure 240: A summary of Tumbug patterns, with regard to Parts of Speech and motion.

<u>Tumbug</u> <u>pattern</u> <u>name:</u>	<u>Allerton's</u> <u>pattern</u> <u>name:</u>
Isolation	--
Emanation	Mono-valent
Contact	Bi-valent
Transfer	Tri-valent
--	Tetra-valent A

Figure 241: Three of Allerton's valency categories exactly match three of the four basic sentence patterns of Tumbug. Allerton's tetra-valent category of sentence patterns involves four objects, which is beyond the number of Tumbug patterns discussed in detail in this document.

have already been discovered earlier in a different context, therefore Tumbug is already mostly supported by existing grammar theory. These matches are shown in Figure 241.

Side conjecture: Human parsing of a sentence, whether that input is written or spoken, may involve in part the operation of matching the grammatical pattern of the input to a small set of known, allowable grammatical patterns. This matching would be a very fast operation because of the small number of possible grammatical patterns involved, each of which has only small structures involved.

Various problems with the Kolln criteria for classifying English sentences often become evident, and are sometimes noticed by English teachers, therefore further detailed comparison between Tumbug patterns versus Kolln patterns is probably not worthwhile.

14.2.2 Allerton sentence patterns

Since Allerton mentioned a valency category that is not part of Kolln's patterns or Tumbug's five mentioned patterns, namely the tetra-valent category, this section briefly considers that category and its relationship to Tumbug. The tetra-valent categories are shown as text in Figure 242.

Allerton lists five examples of the tetra-valent category. Of those five examples, only one involves a fourth true object. Incidentally, Tumbug calls

Structure	Example of Structure
TETRA-VALENT	
S + v + O + IO + PÖ	<i>The child paid a large sum to me for it.</i>
	<i>The child paid me a large sum for it.</i>
S + v + O + AE + AE	<i>The child moved it from here to there.</i>
S + v + O + L + AE	<i>The child put the message over well.</i>
S + v + O + L + IO	<i>The child typed out the article for me.</i>
S + v + O + L + PÖ	<i>The child played me off against them.</i>

S = Subject
 v = Verb
 O = Object
 IO = Indirect Object
 PÖ = Prepositional Objoids
 AE = Adverbial Elaborator
 L = Limiter

Figure 242: Allerton's tetra-valent categories with examples and abbreviation key, all described with text. (Source: Allerton 2006, p. 171.)

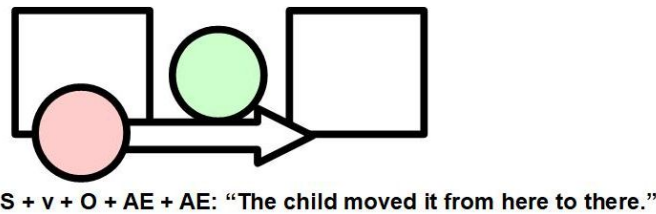
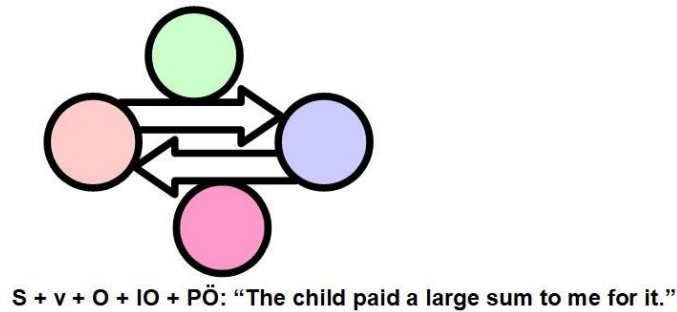


Figure 243: The first two tetra-valent examples of Allerton, all described with Tumbug images.

this sentence pattern "Swap" since it generalizes to any type of trade due to two people + two exchanged objects. The other Allerton examples show either two locations, which are not true objects, or phrasal verbs, each of which can be reduced to one verb. The first two examples in this category are shown in Figure 243.

The tetra-valent (= Swap) patterns suggest that there does not exist any upper limit to the number of sentence patterns that may exist. This is correct, and this is partly why Tumbug was developed: Tumbug is *model-based*, which means that any number of objects and actions can be combined in any way, which results in a system that can be arbitrarily complicated but will still be describable by a visual model. Some examples of higher valency are:

- swap (4-valent): 2 people + 2 objects
- escrow (4-valent): 3 parties + 1 property
- nuclear fusion (4-valent): 2 incoming atoms + 2 outgoing atoms

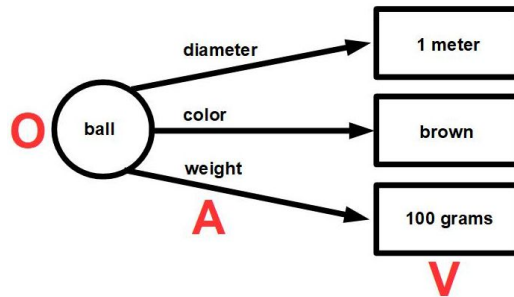


Figure 244: The traditional way to diagram an OAV triple.

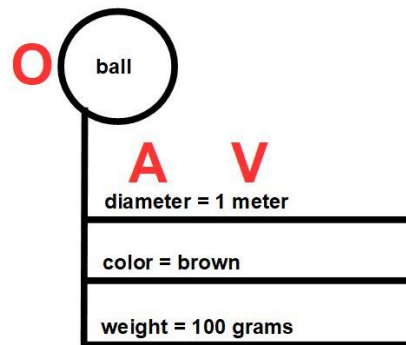


Figure 245: The Tumbug way to diagram an OAV triple.

- attacked pinned piece in chess (4-valent): 4 pieces
- discovered check whose uncovering piece wins a piece in chess (4-valent): 4 pieces
- swap of two properties through one escrow (5-valent): 2 properties + 3 parties

14.3 Object-attribute-value (= OAV) triples

Object-attribute-value (OAV) representation is relatively well-known in computer science, and is usually called an "OAV triple," or sometimes "OAV triplet." The traditional OAV way of diagramming OAV triples is shown in Figure 244, and the equivalent Tumbug way is shown in Figure 245.

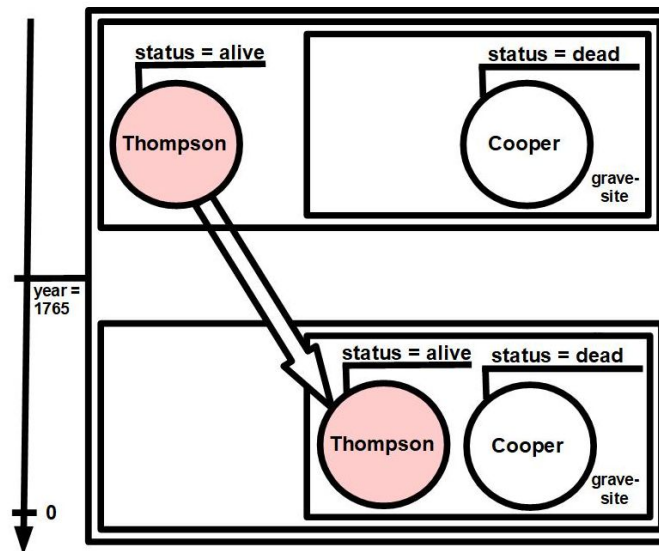


Figure 246: [78] Tumbug for "In 1765, Thompson went to Cooper's grave." This is an SOAV diagram because objects (O, e.g. "Thompson"), attributes (A, e.g., "status"), and values (V, e.g., "alive") are integrated into a larger system (S), where the larger system is the entire Tumbug diagram shown above for this English sentence.

Tumbug uses OAV terminology and its meaning verbatim, although Tumbug sometimes extends this terminology and the associated diagrams. For example, Tumbug sometimes adds a novel extension to OAV called "system-object-attribute-value" (SOAV), shown in Figure 246, and Tumbug sometimes splits the OAV concept into the "subject-object" (SO) concept, an example of which is shown in Figure 247.

14.4 Vector graphics editors

There exists a basic similarity between Tumbug and vector graphics editors in that both systems are typically used to represent real-world systems with diagrams. Both involve images in motion: Tumbug has arrows in its diagrams that imply motion, and vector graphics editors allow users to actually drag such diagrams across the computer screen in real time via a mouse button. Both systems manipulate images with respect to their shapes (circles,

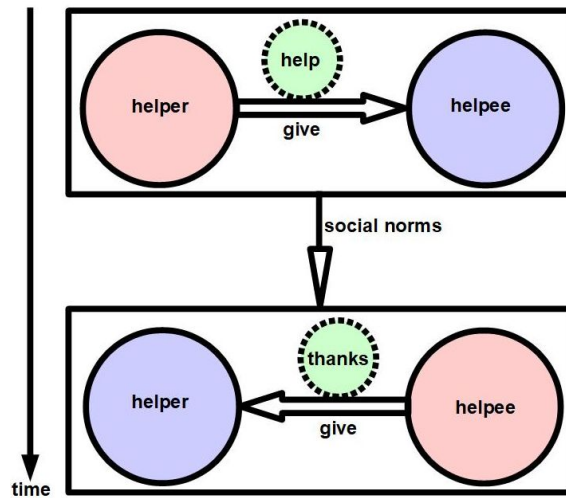


Figure 247: [3] Tumbug for "Per social norms, if a helper helps a helpee, then the helpee will thank the helper." This is an SO diagram because no attributes (A) or values (V) are shown in this system (S) that contains only objects (O) without Attribute Lines.

squares, rectangles, lines, arrows, etc.), motions (translation, rotation, scaling, reflection), shape distortions (shearing, etc.), and their attributes (color, texture, border, etc.). All the aforementioned listed motions are called affine transformations, which in 2D using homogeneous coordinates can be done via a single multiplication of a 3x3 transformation matrix with a 3x1 vector of the object's coordinates. Figure 248 shows an example of one vector graphics editor, OpenOffice Draw, where the user in the process of rotating an icon.

It should not be surprising to realize that the operations done by vector graphics editors are the same operations that humans see performed in the real world on physical objects, since moved diagrams are usually used to represent some motion aspect of the real world in some useful manner to humans. This is summarized in Figure 249, assuming that the occurrence is happening with respect to a given viewer.

The sole visual operations of a vector graphics editor that tend not to correspond to anything in the real world are the abilities to create and destroy objects instantly. (Spontaneous creation of virtual particles of the real world

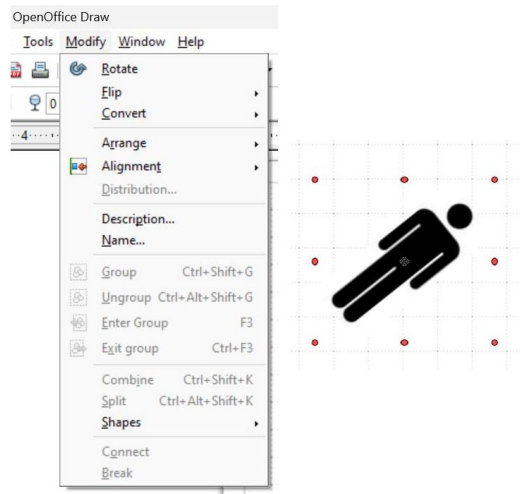


Figure 248: An OpenOffice Draw menu after an image has been selected. Note the affine transformations of Rotate and Flip (= reflection) at the start of the Modify menu.

is a possibility in theoretical physics, however.) A vector graphics editor might have an "Insert | Picture" menu option to implement the creation operation of an object (see Figure 250, and the deletion operation might be done by selecting the object and hitting the "delete" key on the keyboard. It might be said that the human brain uses an unknown piece of software that is a combination of vector graphics editor + simulator + video recorder.

Creation of objects out of nothing can be represented easily by Tumbug, as well, since when the history of the object is viewed with help from a timeline, suddenly the object is question ceases to exist before a certain point in its history. If a Tumbug diagrammer desires to be more explicit, the "Does Not Exist" value could be placed at the specific time and specific place that an object appeared, as shown in Figure 251. Conclusion: Tumbug can represent any primary operation that can be performed by any typical vector graphics editor.

The same statements are true for object deletion. See Figure 252.

<u>affine trans-formation</u>	<u>example of a real world occurrence wrt a viewer:</u>	<u>Projective trans-formation?</u>	<u>Rigid trans-formation?</u>	<u>use of a 3x3 transformation matrix</u>
translation	sideways motion	yes	yes	$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
rotation	rotation or rolling	yes	yes	$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
scaling	forward/backward motion	yes	no	$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
reflection	flipping or reflection	yes	yes *	$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
shearing: shearing X	physical collapse at an angle wooden box walls leaning sideways	yes	no	$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \lambda_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
shearing Y	wooden box walls squeezed at angle from sides			$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \lambda_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

* = depends on the definition

wrt = with respect to

Figure 249: A summary of affine transformations and how they relate to the real world.

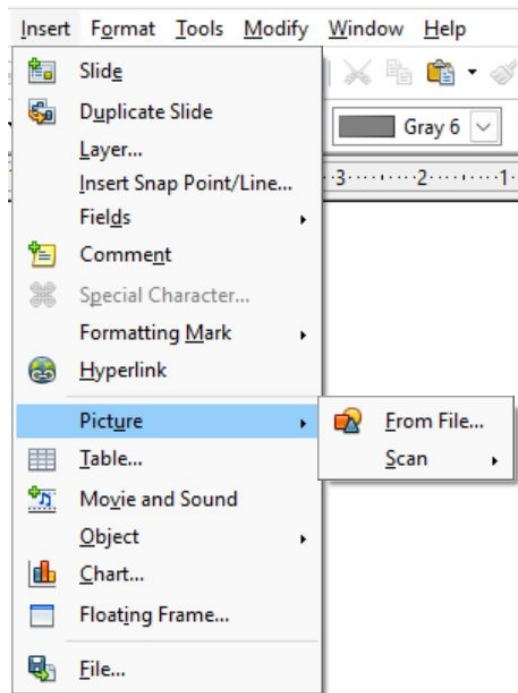


Figure 250: OpenOffice Draw implements the object creation operation by "Insert | Picture."

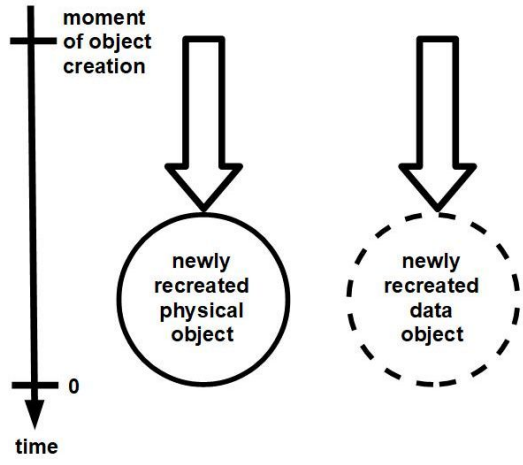


Figure 251: Creation of objects out of nothing can be represented in Tumbug by showing no earlier object history.

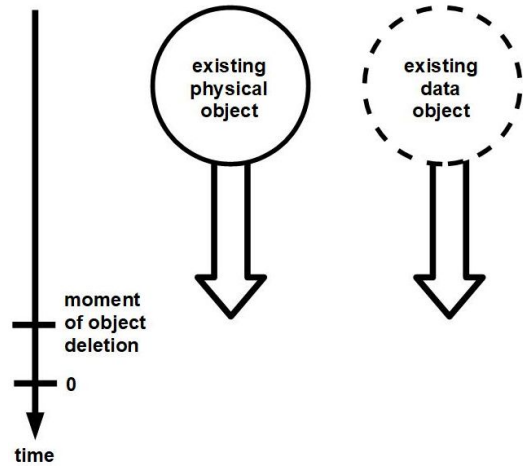


Figure 252: Deletion of objects can be represented in Tumbug by showing no later object history.

14.5 Semantic networks

Semantic networks are a fairly well-known KRM in computer science, one that uses nodes and links, and therefore bears a superficial resemblance to Tumbug. A well-known article by Woods (1975) discusses a large number of shortcomings of semantic networks. Tumbug addresses most or all these shortcomings. Full coverage of a comparison and semantic networks would be too lengthy for this document, however, so only a sampling of Woods' criticisms are given here, with each criticism followed by the Tumbug solution.

14.5.1 Kinship relations

With regard to data bases and semantic networks, the sentence "Harry is John's uncle" is problematic because:

1. The term "uncle" is partially ambiguous because it could refer to either the mother's side of the family or the father's side of a family.
2. Even if the ambiguity is removed, the uncle kinship relation involves comparing nodes that are not directly connected in a kinship tree.
3. Even if the applicable nodes are explicitly connected with an "uncle" link, it is too impractical to store all possible uncle relationships explicitly, due to combinatorial explosion of all possible matches.

See Figure 253 for a standard kinship tree to see the nodes and links that are involved in this problem. If a specific ego (= the person under consideration when describing that person's kinship relationships) is chosen, then the ego and the two choices of uncle are marked with a 0D Marker as in Figure 254. Tumbug solves these issues by the following solutions, respectively:

1. Either use an "OR" to disjunct the mother's side of the family with the father's side of a family, or combine both sides of the family into a more generic group called "parents' siblings" and use the male members of that group.
2. Use a 0D Marker to flag the ego and uncle group, then use a relocatable template that highlights any two nodes, based on the ego at one end

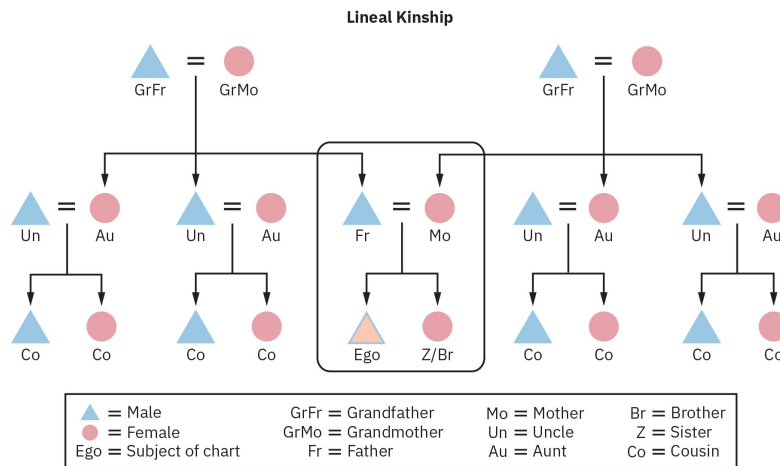


Figure 253: A standard (lineal) kinship diagram. (Source: “Eskimo Kinship Chart” by Fred the Oyster/Wikimedia Commons, CC0.)

and the uncle distance-and-position at the other end as in Figure 255 and Figure 256. (Again, one of the main goals of Tumbug is to represent everything visually, as much as practical.)

3. Store a specific ego-uncle relationship only when that specific relationship has been needed to be considered by the system when answering a query.

Further development anticipated: The uncle kinship relationship template shown is rigid and specific to the tree layout shown, not flexible with respect to arbitrary tree layout or unpredictable counts or unpredictable branches. The conversion of such a rigid to a flexible template was not attempted.

14.5.2 Specific objects versus generic objects

Woods wrote (Woods 1975, p. 22): “For example, if I create a node and establish two links from it, one labeled SUPERC and pointing to the “concept” TELEPHONE and another labeled MOD and pointing to the “concept” BLACK, what do I mean this node to represent? Do I intend it to stand for the “concept” of a black telephone, or perhaps I mean it to assert a relationship between the concepts of telephone and blackness—i.e., that telephones are black (all telephones?, some telephones?). When one devises a

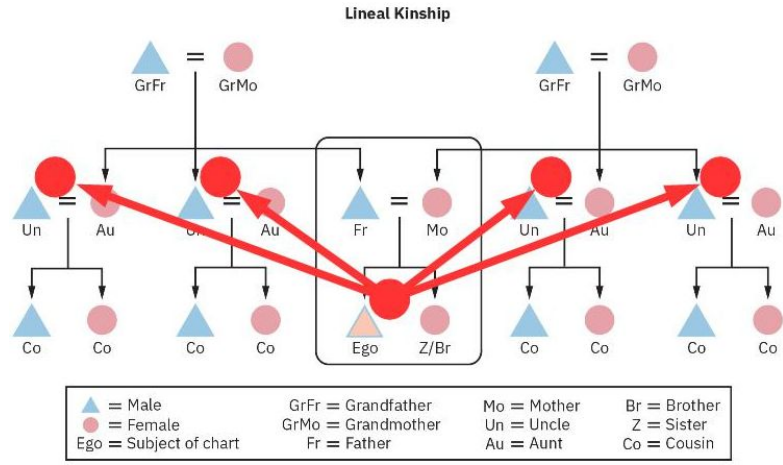


Figure 254: A kinship diagram that flags the possible uncles of the shown ego.

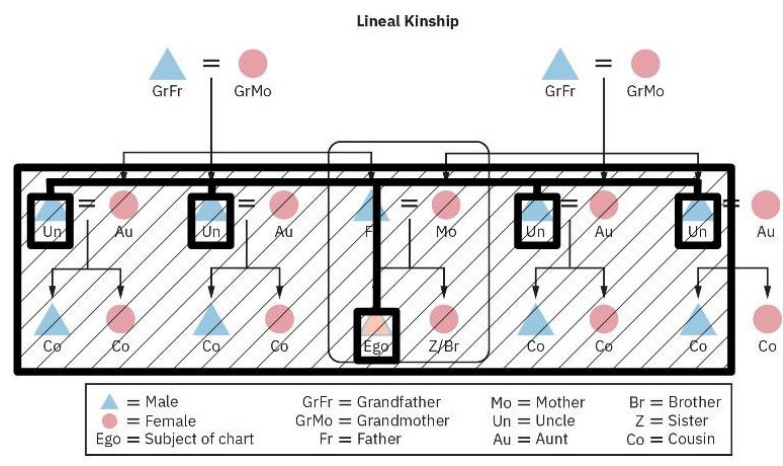


Figure 255: The ego-uncle relationship can be thought of a spatial template that can be moved to start at any ego node.

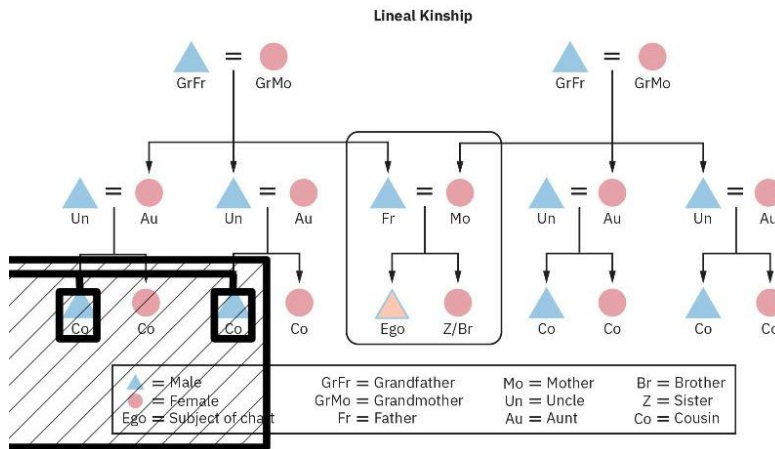


Figure 256: The ego-uncle relationship template moved to a different ego node (off the left-hand edge of the chart).

semantic network notation, it is necessary not only to specify the types of nodes and links that can be used and the rules for their possible combinations (the syntax of the network notation) but also to specify the import of the various types of links and structures—what is meant by them (the semantics of the network notation).”

First, presumably SUPERC = superclass, which would be clearer in modern times due to the modern popularity of object-oriented programming. Woods’ described semantic network diagram is shown in Figure 257.

The three interpretations that Wood mentioned are shown in Figure 258.

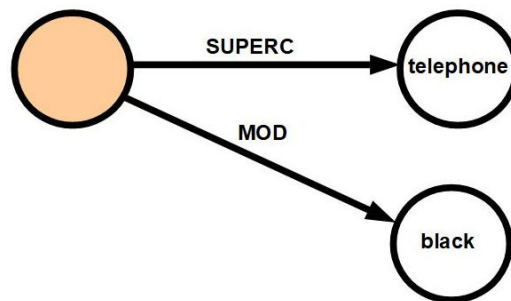


Figure 257: The SUPERC semantic network example described by Woods.

Tumbug represents a superclass using the Venn diagram convention of drawing a region around the region in question, where the larger region represents the superclass. Tumbug represents each of these three interpretations differently, by virtue of making attribute-value pairs distinctly different from Object Circle labels and Location Box labels, and by virtue of making objects and classes distinctly different via Object Circles and Aggregation Boxes, and by virtue of allowing Aggregation Boxes to span arbitrary regions, where each Aggregation Box may have its own unique label. Woods is correct about the drawbacks of semantic networks, but clearly Tumbug has much more representative power than semantic networks. Even object-oriented programming clearly has more representative power than semantic networks. Semantic networks are clearly becoming old-fashioned due to their inherent limitations.

14.5.3 Intensions versus extensions

Woods wrote (1975, pp. 22-23): "Basically a predicate such as the English word "red" has associated with it two possible conceptual things which could be related to its meaning in the intuitive sense. One of these is the set of all red things—this is called the extension of the predicate. The other concept is an abstract entity which in some sense characterizes what it means to be red, it is the notion of redness which may or may not be true of a given object; this is called the intension of the predicate."

This confusion was already addressed in this document by the insight that a concept can arbitrarily change its Part of Speech merely by moving the node that holds that concept to a different part of the OAVC diagram of Figure 330. Therefore "red" as an adjective differs from "red" as a noun in a diagrammatically simple way in Tumbug. Therefore "extension" is "red" as an adjective, "intension" is "red" as a noun. Such Parts of Speech differences were shown in Figure 259, and are shown again here in Figure 259 with only words for colors.

15 Applications of Tumbug

15.1 Natural language translation

Below is the translation of a single English sentence to two different foreign languages, where each foreign language is from a different branch of the Indo-European languages: French, from the Romance languages, and German,

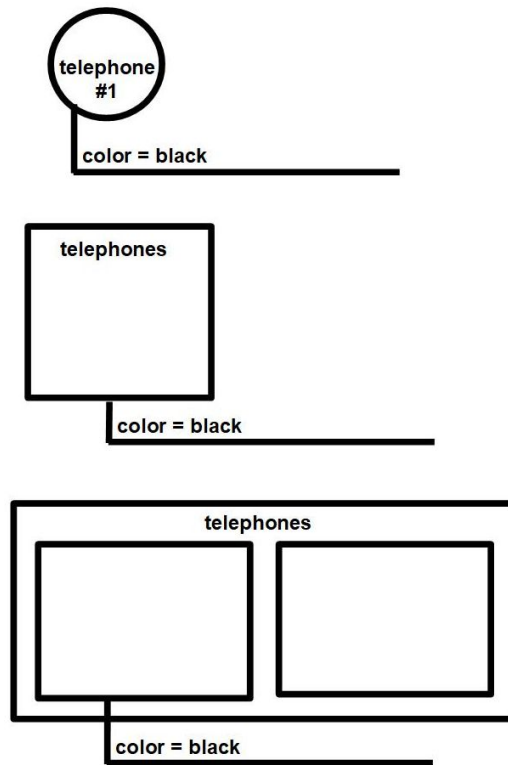


Figure 258: Woods' three interpretations of the SUPER example, as represented in Tumbug. Top: "The particular telephone in question (= telephone #1) is black." Middle: "All telephones are black." Bottom: "Some telephones are black."

<u>NOUN</u>	<u>ADJECTIVE</u>	<u>VERB</u>	<u>ADVERB</u>
----ALL PARTS-OF-SPEECH FORMS EXIST (4/4)----			
blackness	black	to blacken	blackly
blueness	blue	to blue	bluely
greenness	green	to green	greenly
redness	red	to redden	redly
whiteness	white	to whiten	whitely
yellowness	yellow	to yellow	yellowly
----NO ADVERB FORM EXISTS (3/4)----			
purpleness	purple	to purple	-
----NO VERB FORM EXISTS (3/4)----			
----NO VERB FORM OR ADVERB FORM EXISTS (2/4)----			
orange	orange	-	-

Figure 259: The concept of color, like most concepts, can appear as different Parts of Speech.

(1E) I threw the ball. [English]
 (1F) J'ai lancé la balle. [French]
 (1G) Ich habe den Ball geworfen. [German]

Figure 260: An English sentence translated into French and German.

from the Germanic languages. Three steps are shown in the translation to Tumbug: (1) translation, (2) analysis, (3) Tumbug representation, then a general discussion of the process and its nuances.

15.1.1 Conversion from a natural language to Tumbug

1. Translation

See Figure 260.

2. Analysis

See Figure 261.

3. Tumbug representation

(2E) I threw the ball. [English]
 subject = I
 direct object = ball
 gender of direct object = N/A
 article = the
 definiteness = definite
 verb as infinitive = throw
 time tense = simple past
 auxiliary verb, as infinitive = N/A
 semantic context = N/A

(2F) J'ai lancé la balle. [French]
 subject = je
 direct object = balle
 gender of direct object = feminine
 article = la (feminine)
 definiteness = definite
 verb as infinitive = lancer
 time tense = present perfect
 auxiliary verb, as infinitive = avoir
 semantic context = to throw with aim, a detached object

(2G) Ich habe den Ball geworfen. [German]
 subject = ich
 direct object = Ball
 gender of direct object = masculine
 article = den (feminine, accusative)
 definiteness = definite
 verb as infinitive = werfen
 time tense = present perfect
 auxiliary verb, as infinitive = haben
 semantic context = to throw with aim, a detached object

Figure 261: Analysis of the sentence in the three translations.

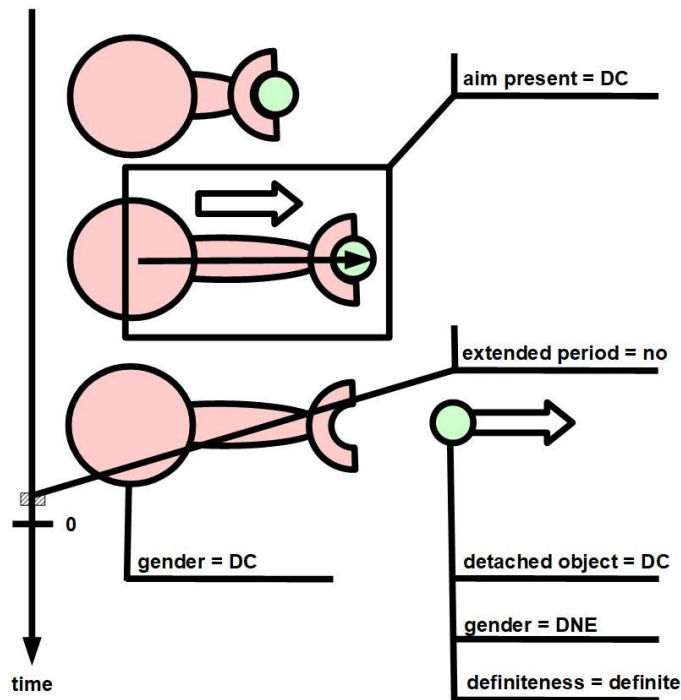


Figure 262: Tumbug for "I threw the ball" in English.

The following abbreviations appear in the diagrams:

DC = Don't Care
 DNE = Does Not Exist

For the English sentence, see Figure 262.
 For the French translation, see Figure 263.
 For the German translation, see Figure 264.

15.1.2 Discussion of language translation

1. Small mismatches

Evidence that Tumbug is nearly a universal representation between these three languages is that the iconic Tumbug structures for the same sentence are absolutely identical, as seen in Figure 265.

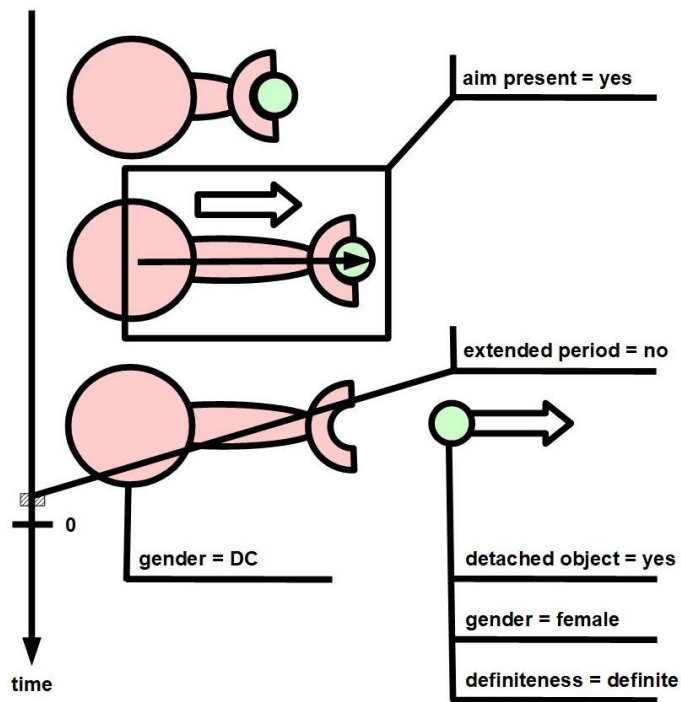


Figure 263: Tumbug for "I threw the ball" in French.

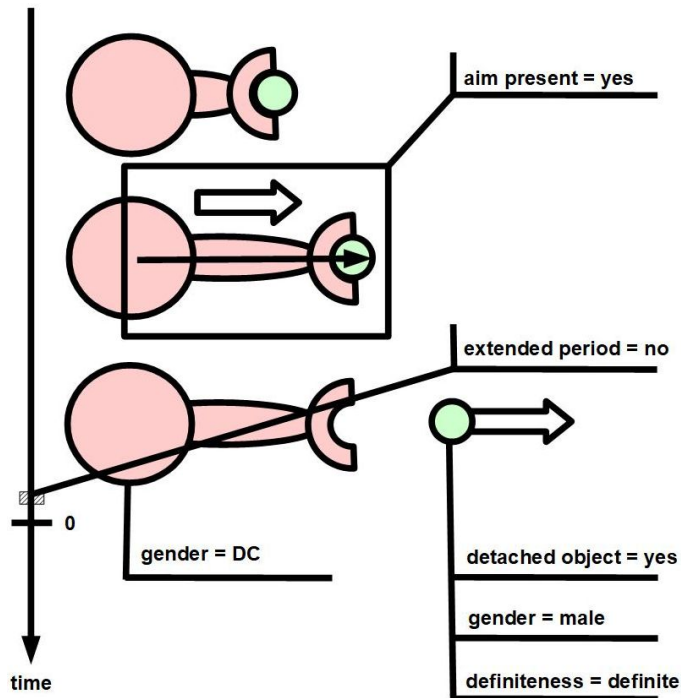


Figure 264: Tumbug for "I threw the ball" in German.

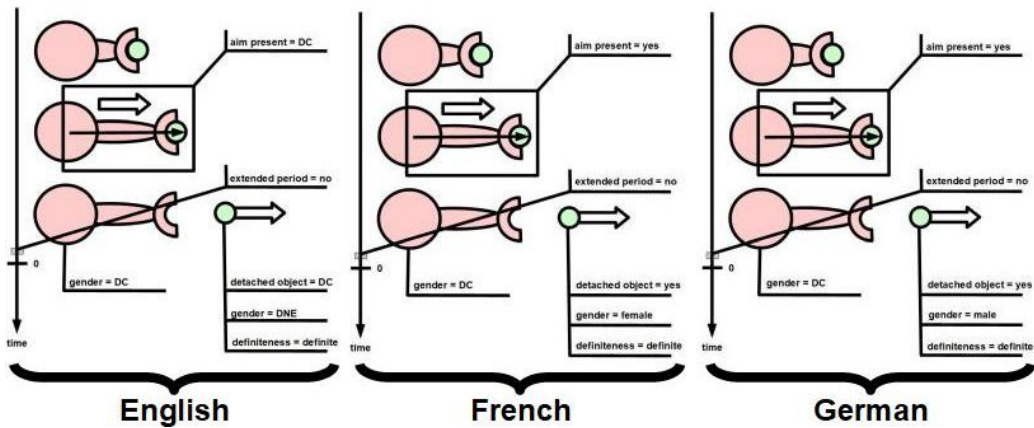


Figure 265: Comparison of Tumbug representations of the same sentence across three languages.

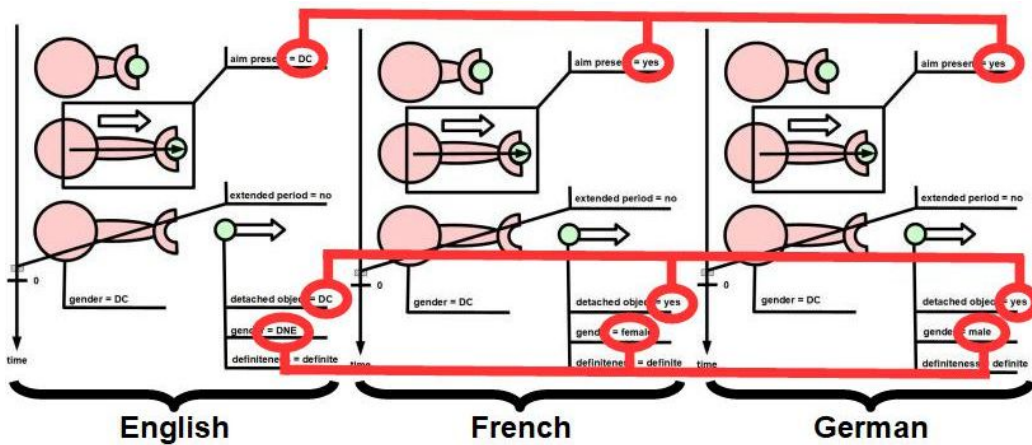


Figure 266: Emphasized differences between Tumbug representations of the same sentence across three languages.

There are a few subtle mismatches, however, all in the attribute values, and all due to some languages being more specific than others. These mismatches of attribute values are marked in Figure 266.

Pairs of minor mismatches across languages in Tumbug of this example are marked with arcs.

The mismatches are the following:

1. Regarding aim: French and German care about the exact meaning of "to throw" since each of those languages would use a different verb if the meaning were slightly different. For example, French would use "jeter" if no aim or detached object were involved, and German would use "abgeben" (or "ausgeben") for the same reason. English makes no such distinction, so English shows "DC" (= don't care) for the "aim present" attribute.
2. Regarding attachment: French and German care whether the object being thrown is an attached object or not, such as "to throw a switch" versus "to throw a ball." A different verb would be used, depending on the attachment status.
3. Regarding gender: French and German have different grammatical genders for their respective words for "ball." English has no grammatical

genders so it shows "DNE" (= Does Not Exist). In this case, grammatical gender is a human-attributed quality that does not exist in inanimate objects, so in this example, object gender is irrelevant. If a human were in the place of direct object or indirect object, however, then gender could be relevant, possibly because of outfit, marriage status, voice range, or any other influences of biological gender.

Other subtle mismatches could occur easily with different sentences or with different languages. For example "you" may have different levels of formality, such as in German and Dutch, some languages do not have grammatical genders, such as in Japanese, and some languages do not have articles, such as in Japanese and Latin.

All such mismatches are due to differing levels of details across the languages, however, none whatsoever with the structure and relationships between the top-level entities or actions after the exact word meaning has been determined. Of course a sentence could be completely ambiguous even in its own language, such as "The pot is in the pen," but that is a deficiency on the part of a speaker or the language, not a deficiency in Tumbug. Tumbug does have some potential weaknesses, but these are subtle and very unlikely to be encountered. These weaknesses are discussed later in this document.

2. Corrections of the small mismatches

There exist only two obvious ways to correct the aforementioned minor mismatches in the translation example: (1) require the writer/speaker to be more specific, as specific as the most specific languages that might be encountered in later translation, (2) use context to determine the most likely meaning. The first option is usually not within the reader's control, so usually context must be used.

Consider how context of a sentence can be implied by following sentences:

1. Textual passage: "I saw the player trying to steal third base and he wasn't too far away. I threw the ball."
Suggested context: baseball, where a ball is a critical piece of equipment that must be thrown accurately
Implied aim present: TRUE
Implied detached object: TRUE
2. Textual passage: "I was going through the trash, pocketing anything

of value and tossing aside anything else. First I found an earring, so I pocketed it. Then I found a disgustingly moldy ball of cheese. I threw the ball.”

Suggested context: trash, where the ball described is something the holder would toss aside quickly, to anywhere

Implied aim present: FALSE

Implied detached object: TRUE

The contextual information in this case clarifies the two main ambiguities from before: whether there was an intended target, and whether the thrown item was attached. The ambiguity of which gender is involved depends only on language-specific grammatical rules, and not the meaning of the word after the proper word has been clarified. The result of this context determination from the English (or other language) sentence is the first step for translation, and results in essentially a word vector of TRUE-FALSE values. The second step of translation matches this word vector against a database of (typically) foreign words, as shown in Figure 267.

The third step would be to calculate which context-word combinations had the closest match. For example, in the context of baseball, a context that was determined by analysis of the the sentence just described whose source language was English, each context vector of the source language would be matched against each context vector of the target language. For two contexts from English and two candidate words from French the result would be (2x2 =) four vectors, each cell of which contains an indication as to whether there was a match or mismatch. For example, TRUE matched with FALSE = mismatch, and FALSE matched with DON'T CARE = match. Figure 268 shows the final results. One simple metric for determining degree of match would be to simply add the number of matches. In the context of baseball (C1), the highest matching candidate French word was "lancer," which had two matches versus only one match with "jeter." Figure 268 shows this entire matching process as described.

15.2 Mathematics: Arithmetic

Arithmetic in Tumbug uses a Causation Arrow to represent the operator (e.g., "+" or "-") or function (e.g., 3x, x2), and uses Data C Object Circles to represent the numbers. Numbers are not real-world objects so numbers cannot use Physical C Object Circles. This is in contrast to the way the OOP

determined context from the English sentence



example:

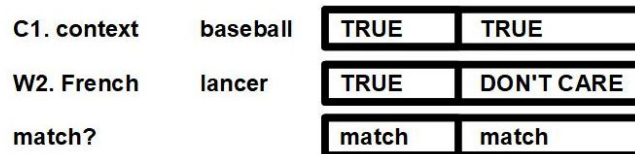


Figure 267: The one determined context (here C1) from the source language is matched against each word in a data base of candidate words in the target language, with one such candidate word (here W2) used in the example. Note that a "DON'T CARE" value will match either a "TRUE" or "FALSE" value.

		aim present:	detachment present:	
C1. context	baseball	TRUE	TRUE	} context vec- tors
C2. context	trash	FALSE	TRUE	
W1. French	jeter	FALSE	DON'T CARE	} word data base vec- tors
W2. French	lancer	TRUE	DON'T CARE	

**results of matches with each C
vector against each W vector:**

combination:	mismatch	match	# of matches
C1 with W1	mismatch	match	1
C1 with W2	match	match	2
C2 with W1	match	match	2
C2 with W2	mismatch	match	1

Figure 268: Each given context (here C1 and C2) is matched against each candidate word in the target language (here W1 and W2), and the count of matches across each vector's elements is shown at the right of that vector. In the context of baseball (C1) being matched against French words, the French word with the highest count is assumed to be the French word that has the closest match. The closest matching French word is therefore assumed to be the word "lancer" (W2), which has two matches, as compared to the next highest matching word, "jeter" (W1), which has only one match. Therefore in the context of baseball, the French word with the closest match to the English word "to throw" is "lancer."

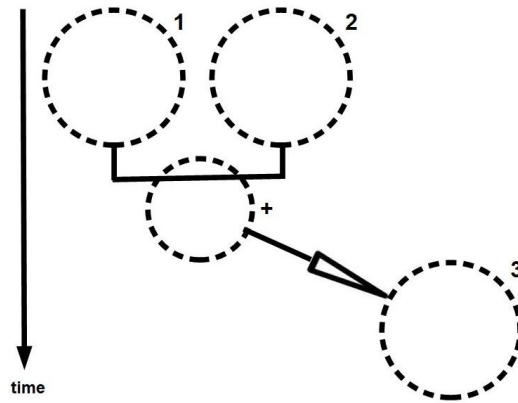


Figure 269: To add the numbers 1 and 2, Tumbug sends them as two pieces of data to a virtual operator called "+".

language Smalltalk handles numbers: in Smalltalk, numbers are objects, and operators are methods. For example, in Smalltalk to add two numbers, such as 1 and 2, Smalltalk would send the method "+ 2" to the number "1" (Winston 1998, p. 16).

In Tumbug, "1 + 2" would be rendered as in Figure 269, which translates to the verbal description "Messages 1 and 2 together are inputted to the virtual adder, which causes the number 3 to be outputted." A "virtual adder" could mean anything like a person's mind or an imaginary computer adder. If a real world computer or real-world calculator were involved then the circle around the Causation Arrow would be solid: a Physical C Object Circle. Numbers will always be virtual, so numbers are always represented by Data C Object Circles. Although computer memory can erase and overwrite numbers at will, human memory cannot: the concept of "1" and "2" are practically permanent concepts in human memory after being learned, as is their sum "3," therefore the diagram shows the "3" concept in a different location than directly below the "1" or "2" so that neither the "1" nor "2" are overwritten by Tumbug conventions. Whether the machine performing the adding is mechanical (e.g., a cash register) or electrical (e.g., a computer), any calculation it performs takes time, therefore a timeline (Time Arrow) should always be present in any Tumbug diagram that represents calculation.

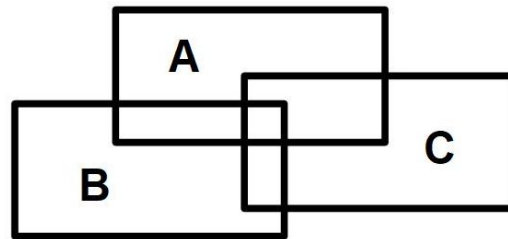


Figure 270: A Venn diagram in Tumbug uses C Aggregation Boxes instead of the familiar circles of Venn diagrams.

15.3 Logic: First Order Predicate Calculus

15.3.1 Syllogisms with Venn diagrams

Venn diagrams are not only closed geometrical figures, each of which may contain points in arbitrary locations within any given region, but can also be considered diagrams of physical regions that may contain objects within any given region, like an aerial view of a horse corral that may contain horses. Therefore a Venn diagram can be considered only a cleaned up and abstracted aerial photograph, which conveniently bridges the gap between reality and geometrical diagrams. Therefore there is no need for Tumbug to include a separate Building Block for a Venn diagram.

One type of formal logic is First Order Predicate Calculus (FOPC). FOPC often uses syllogisms represented as Venn diagrams, which are used to represent sets. As noted earlier in this document, sets are very conveniently represented in Tumbug with any type of Aggregation Boxes, since order (or 2D position) is not important in a set. Therefore three intersecting sets in Tumbug would be represented as three intersecting C Aggregation Boxes, as shown in Figure 270.

There exist only 24 valid types of syllogisms, and each type has been given a person's name. In this document the Socrates example has the Barbara form, the reptiles example has the Celarent form, and the rabbits example has the Darii form.

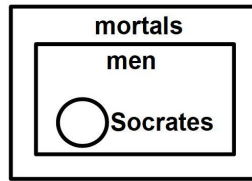


Figure 271: One way to represent the Socrates syllogism in Tumbug.

15.3.2 Barbara form (AAA-1)

The Socrates syllogism example in this section is the well-known syllogism:

All men are mortal.
 Socrates is a man.
 Therefore, Socrates is mortal.

In this syllogism the two sets involved—assuming that the information is regarded as sets whenever possible—happen to be nested and non-intersecting, as shown in Figure 272. In the literature, usually even Socrates is represented as a third set, for some reason, but technically Socrates is an element of a set, not a set *per se*, so is represented an object here.

The way a person would likely construct a mental picture such as in the aforementioned syllogism would be incrementally as each of the three sentences is read: a single mental picture is used across all three sentences, and after each sentence is read a new piece is added to the composite picture as shown in Figure 272. Note that the final composite picture does not use the "mortals" concept as a set, but rather as an attribute-value pair.

The clue that a set is involved is the verbal form "all X," such as in "all men" or "all birds" or "all even numbers." In logical calculus or math the term "for all" would be the analogous clue. The English verb "is" (or "are" or "am") when used in descriptions of the form "X is a Y" (or "X are a Y" or "I am a Y") means "X is in set Y," as in "Socrates is man" or "Tweety is a bird" or "6 is an even number."

Using these rules the aforementioned sequence of three steps results mechanically, with little or no thought, while building up the associated image on the side: (1) "All men are mortal" = "The set of men has attribute mortal" \Rightarrow create set "men" and attach attribute-value "mortality = mortal."

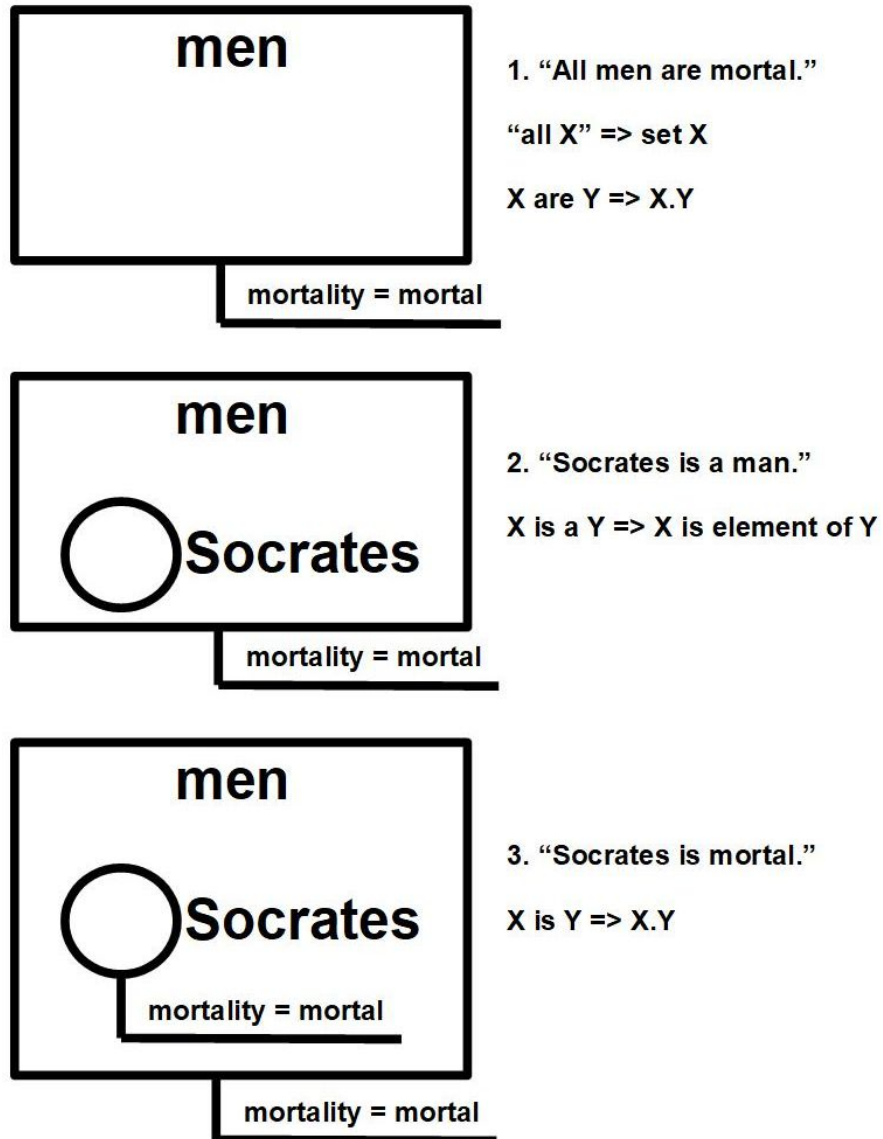


Figure 272: The three diagram creation steps that correspond to the three sentences of the syllogism.

(2) "Socrates is a man" = "The object Socrates is in the set of men" \Rightarrow create object "Socrates" and place it inside set "men." (3) "Socrates is mortal" = "The object Socrates has attribute mortal" \Rightarrow realize that since set "men" has attribute-value "mortality = mortal" then every object in that set must have that same attribute-value, by convention.

The last (3rd) step as represented in Figure 273 may seem somewhat trivial and redundant. However, if this syllogism had been represented with two sets, one set for "mortals" as is usually done, the insight of the third step could be made visible as an arrow connecting Socrates and the set "mortals." Such an insight more closely describes of people would typically think of such a new association, often as a "that's true, though I never quite thought of it that way before" reaction. This type of spatial fusion evidently happens frequently when mentally piecing together maps to plan a route. For example, if a person knows the layout of the interior of a building well, and knows the relative placement of the surrounding buildings but has always entered and exited through the same entrance of the first building, then if challenged with finding a new exit that is closer to the destination building, would likely mentally need to fuse the two separate mental maps in order to decide which exit would be closer to the destination from a global viewpoint—a shortcut that could be always be used thereafter. This same "insight arrow," called a Relationship Marker in Tumbug, is explicitly shown in all the syllogisms in this document. Note that extra connections such as these are likely responsible for increasing a person's "understanding" of a topic (Atkins 2019).

Note that the order of the three sentences of the Socrates syllogism is not critical. For example, if the 1st and 2nd steps were swapped, the conclusion (and final diagram) would be the same, but would have been built up in a different order, as shown in the Figure 274.

Socrates is a man.
All men are mortal.
Therefore, Socrates is mortal.

Insight: The described process of the brain building up composite images that represent meaning across multiple sentences may provide major clues about the brain automatically providing context for disambiguation of pronouns in subsequent sentences. In contrast, if a sentence containing an ambiguous pronoun was a sentence that was removed from its surrounding

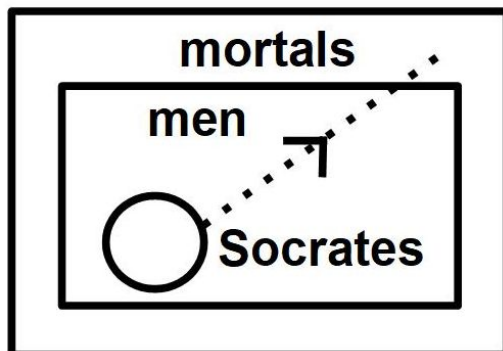


Figure 273: The insight that Socrates is mortal as represented by a Relationship Marker that directly connects the two concepts.

sentences then the sentence would likely lose critical information that could disambiguate that pronoun if the sentence were given to a computer to analyze.

15.3.3 Celarent form (EAE-1)

The reptile syllogism example of this section is of the Celarent form, and is shown in Figure 275.

No reptile has fur.
 All snakes are reptiles.
 Therefore, no snake has fur.

The clue that a set is involved is the verbal form "no X," such as in "no reptile" or "no man" or "no negative number." Implied is that there exists a set called "X" and no element in that set can have the properties described afterward. In logical calculus or math the term "there does not exist" would be the analogous clue.

A complication arises from this example: How to represent "no" (or "not") with respect to an allowed range in Venn diagrams. Two ways are shown in Figure 276 and Figure 277. Representation #1: The X-ed line between the two sets means that an element from one set cannot exist in the set on the other side of the X-ed line.

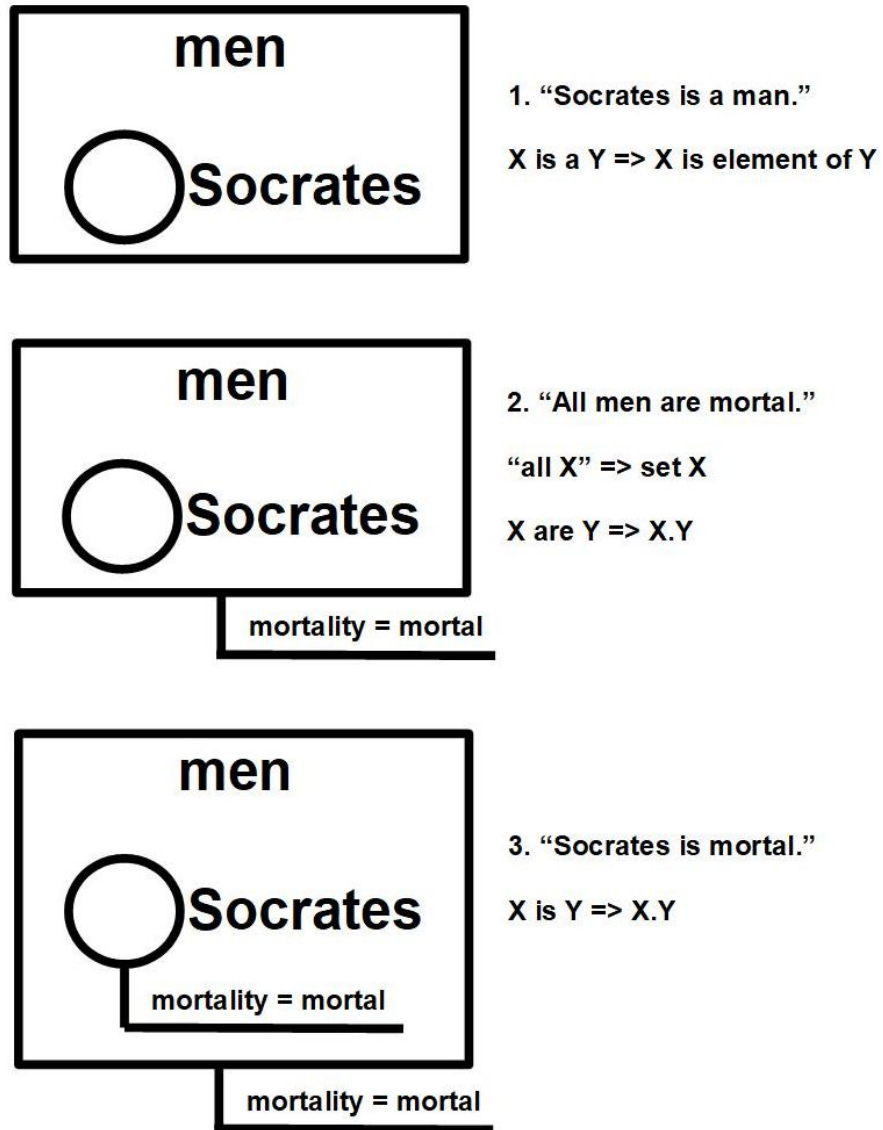


Figure 274: The three diagram creation steps that correspond to the rearranged three sentences of the syllogism.

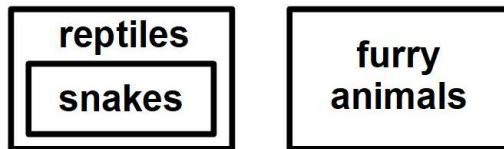


Figure 275: One way to represent the reptile syllogism in Tumbug.

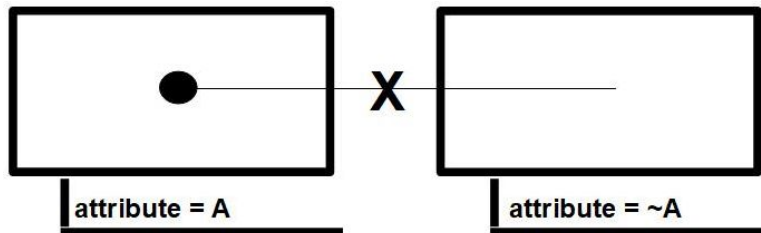


Figure 276: An element of the left set with attribute A cannot appear in the right set.

Representation #2: The black dot represents an element in the set on the left, and the arrows emanating from it suggest the range in which it is allowed to appear in the Venn diagram. The dot can appear in any hatched area, i.e., the dot can appear in any area except the white (unhatched) region. The arrows are shown for clarity here, but are not needed in practice since the shading already spans every region where the element may go. See Figure 277 and Figure 278.

For Tumbug the hatched region representation would probably be clearest, since Tumbug has many lines in the diagram already, and using an X-ed line does not make it clear where the element in the left set can go, only where it cannot go.

Using the same conventions as in the Barbara example, the following steps would be involved to create the appropriate Venn diagram, as shown in Figure 279. (1) "No reptile has fur" = "The set of reptiles and the set of furry animals cannot intersect" \Rightarrow create set "reptiles" and set "furry animals" so that they are non-intersecting. (Alternative: Attach attribute-value "skin covering = NOT fur" to the reptiles set, and attach attribute-value "skin covering = fur" to the furry animals set.) (2) "All snakes are reptiles" = "The set of snakes is a subset of the set of reptiles" \Rightarrow create set "snakes"

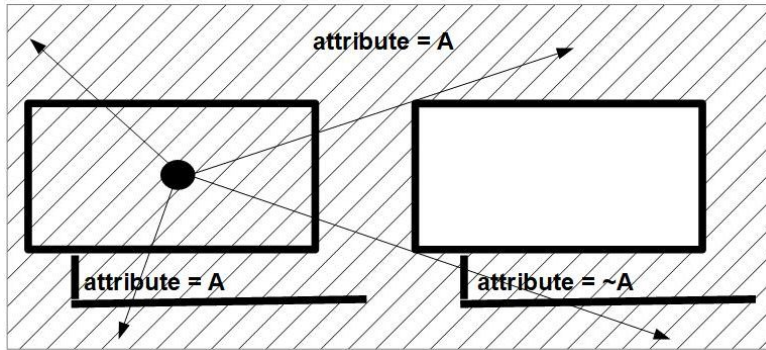


Figure 277: An element of the left set with attribute A can appear in any set or region except the white ($\sim A$) region.

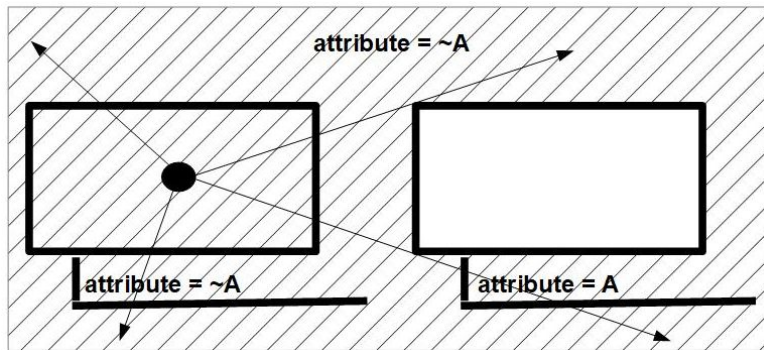


Figure 278: An element of the left set with attribute $\sim A$ can appear in any set or region except the white (A) region. This is the diagram that applies to the reptiles example, where $A = \text{fur}$, $\sim A = \text{NOT fur}$.

and place it inside set "reptiles." (3) "No snake has fur" = "No element in the snakes set can exist in the furry animals set." Realize that since any snake must exist only in the "snakes" set, which is mutually exclusive with the "furry animals" set, this is already implied, by the shading convention.

15.3.4 Darii form (AII-1)

The rabbits syllogism example of this section is of the Darii form. Figure 280 shows this syllogism in square Venn diagram form.

All rabbits have fur.
Some pets are rabbits.
Therefore, some pets have fur.

The clue that a set is involved is the verbal form "all X," here "all rabbits."

Using the same conventions as in the Barbara example, the following steps would be involved to create the appropriate Venn diagram, as shown in Figure 281. (1) "All rabbits have fur" = "The set of rabbits is a subset of the set of furry animals" \Rightarrow create set "rabbits" and set "furry animals" so that "rabbits" is a subset of "furry animals." (Alternative: Attach attribute-value "skin covering = fur" to the rabbits set.) (2) "Some pets are rabbits" = "Some of the set of pets is inside the set of rabbits, but not necessarily inside the set of furry animals" \Rightarrow create set "pets" and make it overlap with the set "furry animals" but in a way that it does not overlap into the purely "furry animals" region. (3) "Some pets have fur" = "Some part of the pets set is inside the furry animals set." Realize that part of the "pets" set exists in the "furry animals" set.

15.4 Computer language representation

Algorithms can be represented in Tumbug rather easily, especially from flowcharts, but also from the code itself.

Sequential steps of an algorithm are shown in Tumbug as occurring sequentially in time, along a Timeline Arrow. The rectangles along the timeline are not flowchart boxes, but rather C Aggregation Boxes. The contents of each C Aggregation Box on the left is the Tumbug representation of the corresponding statement s_i in the code to the right, so each C Aggregation

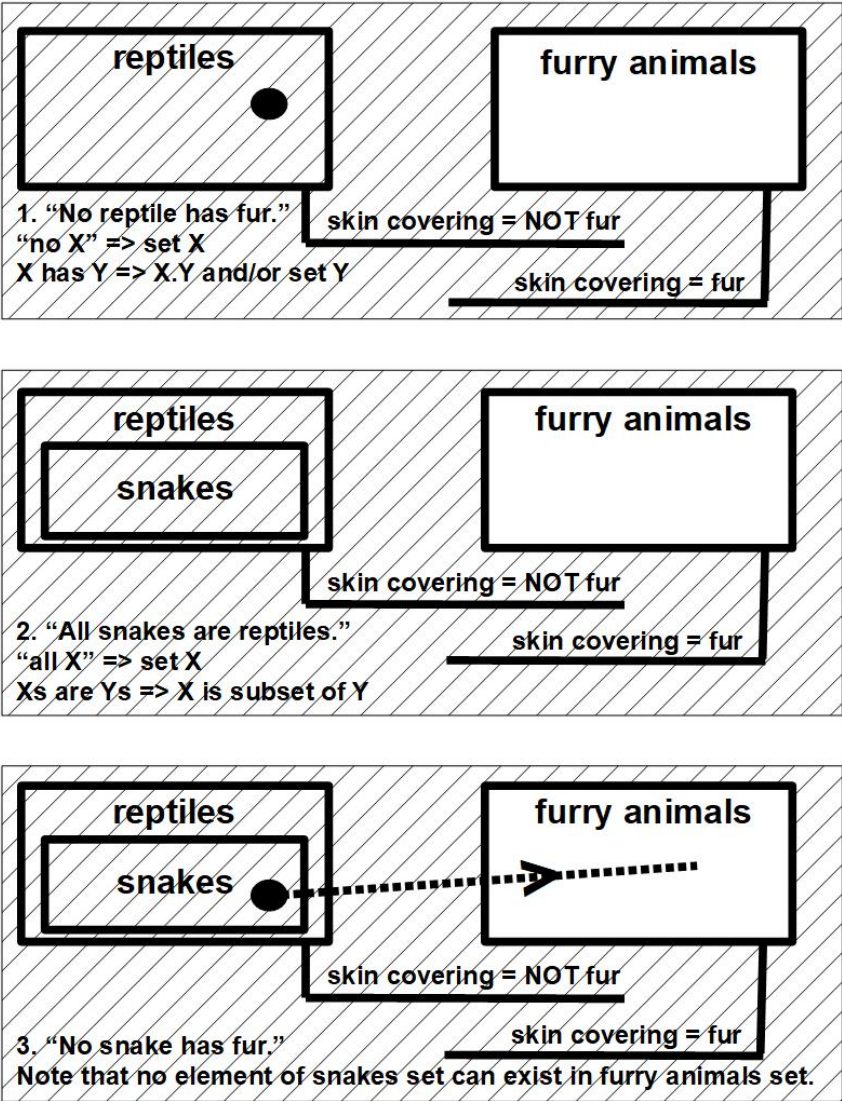


Figure 279: The three diagram creation steps that correspond to the three sentences of the syllogism.

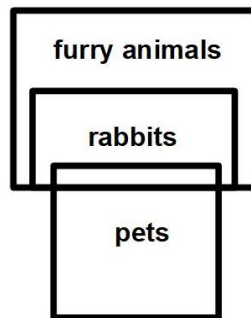


Figure 280: One way to represent the rabbits syllogism in Tumbug.

Box will typically hold connected Tumbug icons such as C Object Circles, Motion Arrows, attribute-value pairs, etc. Each red dot shown in the code samples here indicates the statement within the program that is executing, and there is a corresponding red dot beside the corresponding part of the Tumbug representation.

Caveat: Tumbug's ordinary level of representation deals with human life and human-sized objects, so using Tumbug for computer code would likely be extravagantly wasteful of space. The same is true of mathematical operations. Tumbug is similar to a very high order programming language, especially an object-oriented simulation language, so Tumbug would be most appropriate for important human-level events that lend themselves to visual simulation.

15.4.1 Sequential code

Sequential code as a flowchart in Tumbug would look like Figure 282. Figure 283 of a Tumbug representation of sequential flowchart run being traced suggests why Tumbug is more applicable to machine understanding of a situation: At each step the entire piece of code is recopied, with the only change being where the current program statement is, which is indicated by the red dot. This results in the ability to see not only the current focus (i.e., the current program statement) but also its context in all the code around it (i.e., the program).

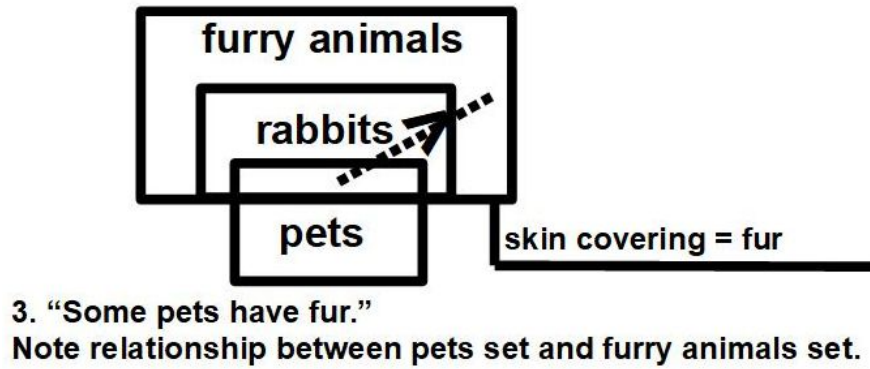
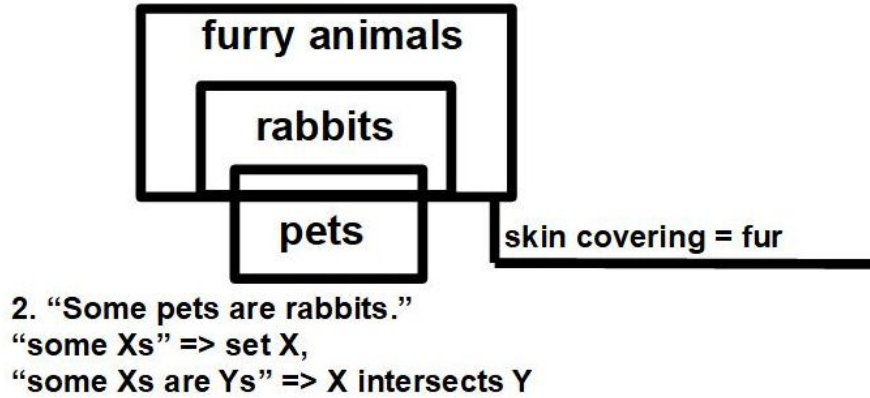
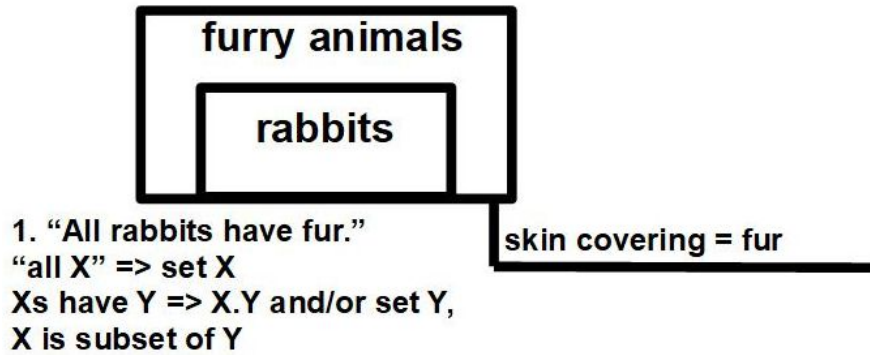


Figure 281: The three diagram creation steps that correspond to the three sentences of the syllogism.

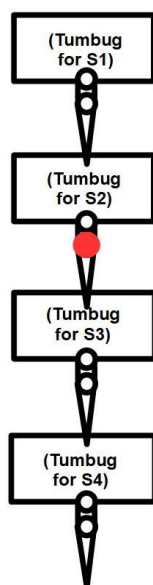


Figure 282: Sequential code in Tumbug is represented by one path composed largely of Pathway Tubes.

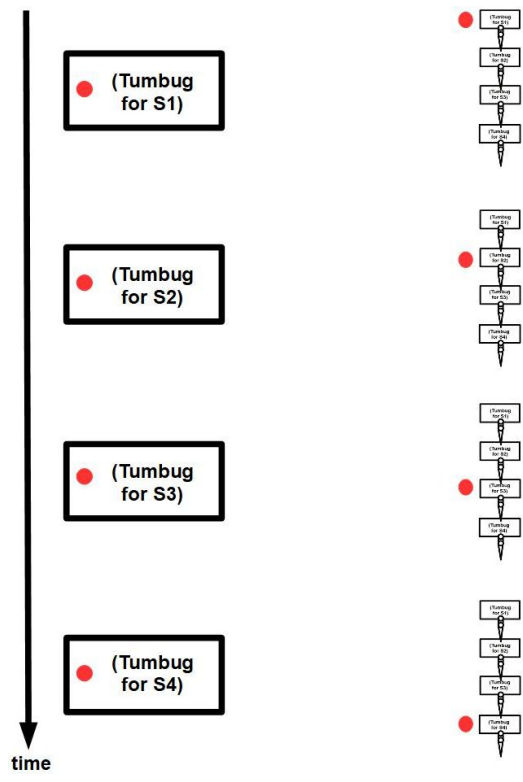


Figure 283: Trace of sequential code in Tumbug, with flowchart copied at each step on right: (S1 S2 S3 S4).

15.4.2 Looped code

A loop in the code would cause the collection of steps inside the loop to occur repeatedly, with each collection occurring sequentially, and the steps within each loop occurring sequentially, as well. This is shown in Figure 284 and Figure 285.

15.4.3 Branching code

A conditional statement (such as IF-THEN, IF-THEN-ELSE, or CASE) causes only one thread of execution to occur, chosen appropriately from the multiple alternatives given in the program. Tumbug uses a Split Time Arrow to represent this splitting of possibilities. The result looks very similar to a flowchart, as shown in Figure 286 and Figure 287.

15.5 Natural language representation

This section discusses the more difficult grammatical topics within English that might cause translation from English into Tumbug to be difficult. Almost all these topics are about verbs. The first section covers several of the simpler topics in a single section; each subsequent section covers one complicated topic per section.

15.5.1 Categories of verbs

1. Active verbs versus stative verbs

Active verbs are also called "dynamic verbs." Active verbs are verbs that describe obvious actions (e.g., "to throw," "to run," "to hit," "to jump," "to sing"), whereas stative verbs describe state of being (e.g., "to know," "to earn," "to love," "to want," "to own").

Sometimes a third category is included, called "event verbs." This is said to include sentences such as "Four people died in the crash" and "It's raining again," but these examples fit well as stative verbs since "to die" is a change of state from living to dead, and "to rain" is a change of state of weather to wet precipitation.

The differences between these categories is clearly represented in Tumbug because actions must either involve motion, whereupon a Motion Arrow and C Object Circle is used, or be caused or entered via a state diagram,

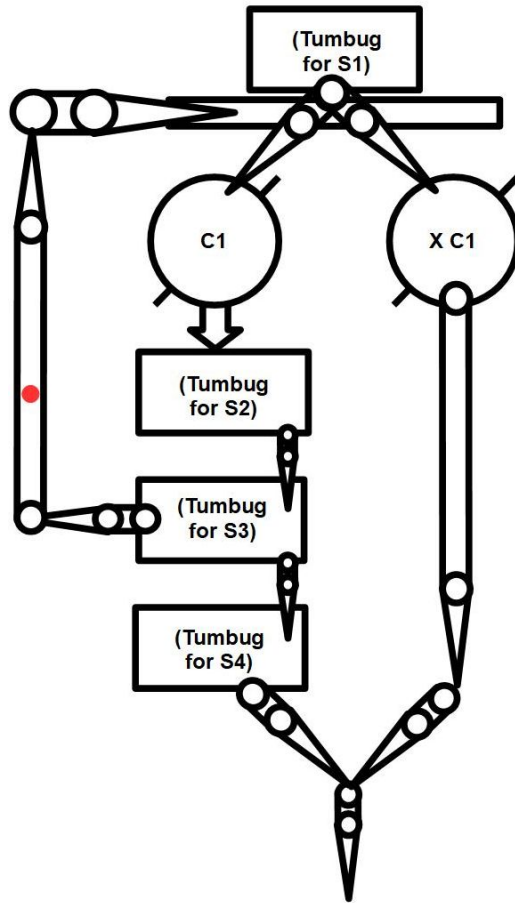


Figure 284: An IF-THEN in Tumbbug is represented by two paths, the choice of which depends on one state.

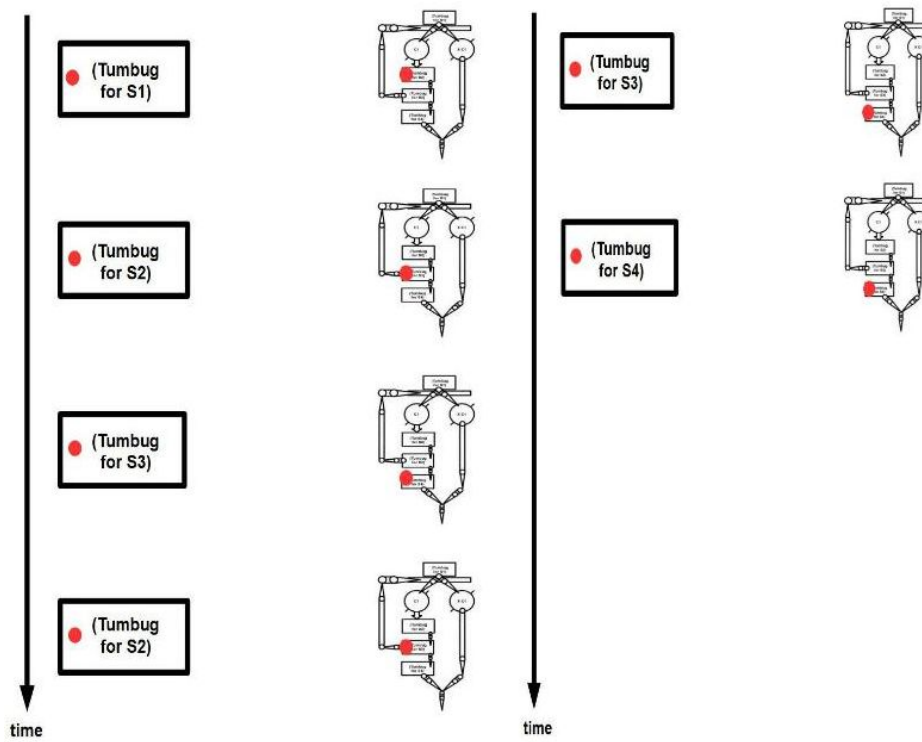


Figure 285: Trace of looped code in Tumbug, with flowchart copied at each step on right: (S1 S2 S3 S2 S3 S4).

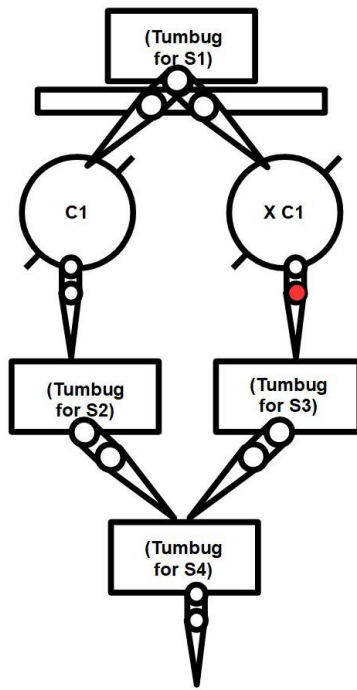


Figure 286: An IF-THEN-ELSE in Tumbug is represented by two paths, the choice of which depends of one state.

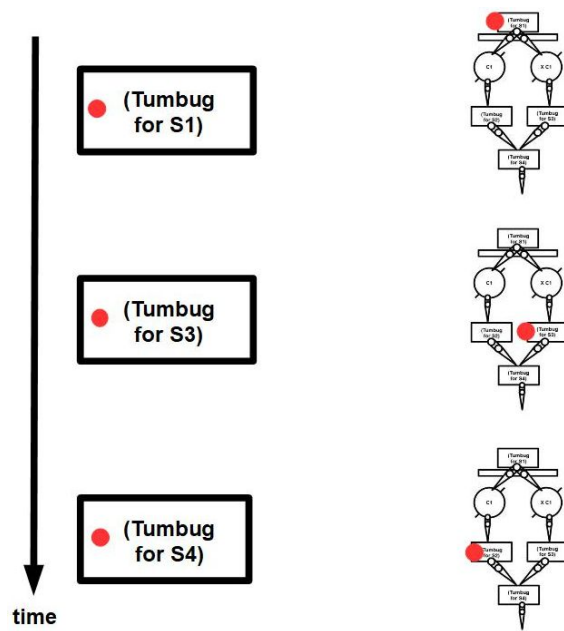


Figure 287: Trace of branching code in Tumbug, with flowchart copied at each step on right: (S1 S3 S4).

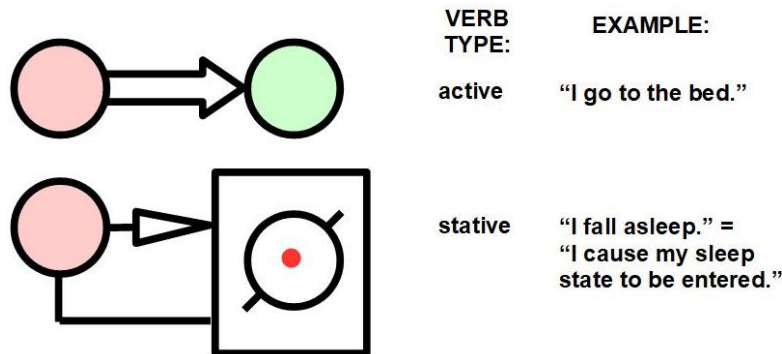


Figure 288: In Tumbug, active verbs use very different arrows than stative verbs.

whereupon a Causation Arrow and State Circle is used. If color coding is used then the difference in diagrams is even more obvious since State Circles have no color whereas C Object Circles do. See Figure 288.

2. Transitive verbs versus intransitive verbs

A transitive verb is one that requires a recipient object for the sentence to make sense (e.g., "I pushed the cart," "Please bring coffee," "Juan threw the ball"), in contrast to the sentence making without a recipient object (e.g., "They jumped," "The dog ran," "She arrived"). Some verbs can be either, depending on context (e.g., "Her husband shaved" versus "The barber shaved the customer"). The difference is extremely obvious in Tumbug: if only the subject's C Object Circle is present then the verb is transitive, but if the direct object's C Object Circle is present as well as the subject's C Object Circle then the verb is intransitive. See Figure 289.

3. Linking verbs

Linking verbs do not describe actions or causations of states, but rather describe an object with either a predicate adjective or a predicate noun. In Tumbug, a predicate adjective is represented by attribute-value pair that hangs off an Object Circle, and a predicate noun (also called a "predicate nominative") is represented as a superset that is represented as a C Aggregation Box that surrounds the predicate noun. The main example of a linking verb in English is "to be," which in Spanish has the two forms "ser" and "estar." Any verb that describes sensory input (e.g., "to feel <adjective>," "to smell <adjective>," "to taste <adjective>") or any verb that describes

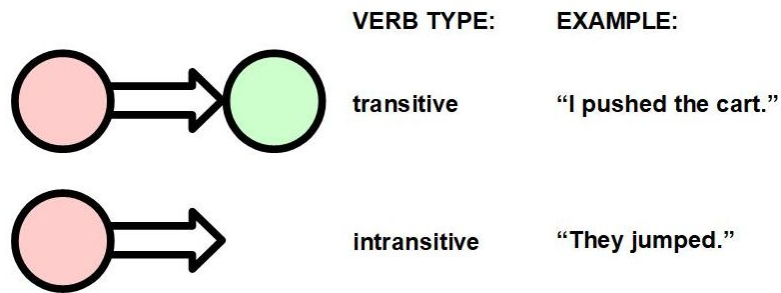


Figure 289: In Tumbug, transitive verbs use two Object Circles whereas intransitive verbs use one Object Circle.

impressions (e.g., "to seem <adjective>," "to look <adjective>," "to become <adjective>") is a linking verb. Tumbug inherently does not consider linking verbs to be verbs in the normal sense since Tumbug does not represent linking verbs with any type of arrow, only with Attribute Lines. See Figure 290.

4. Helping verbs

"Helping verbs" are also called "auxiliary verbs." A helping verb is an additional verb used with the main verb to convey some type of additional information. The types of information conveyed can be:

- time tense
 - HELPING VERB EXAMPLES: to be, to have, to do
 - SENTENCE EXAMPLES:
 - * I am going to the zoo.
 - helping verb = "to be"
 - conveyed info = present continuous tense
 - * I have waited a long time for this.
 - helping verb = "to be"
 - conveyed info = present continuous tense
 - * I did not want to go home.
 - helping verb = "to do"
 - conveyed info = simple past tense
- modality

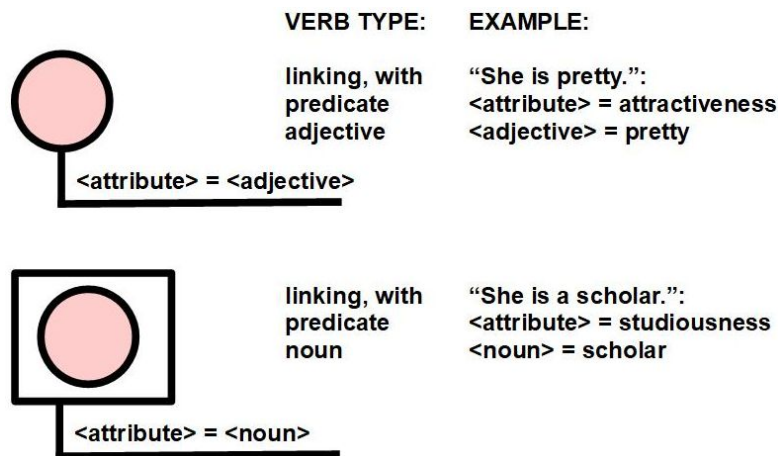


Figure 290: There exist two types of linking verbs, and neither involves an action or a causation.

– HELPING VERB EXAMPLES: can, could, may, might, will, would

– SENTENCE EXAMPLES:

- * It might rain.
 - helping verb = "might"
 - conveyed info = possibility
- * Jorgen can skate backwards.
 - helping verb = "can"
 - conveyed info = (general) ability
- * I may delete this later.
 - helping verb = "may"
 - conveyed info = possibility

- emphasis

– HELPING VERB EXAMPLES: to do

– SENTENCE EXAMPLES:

- * I do like the book!
 - helping verb = "to do"
 - conveyed info = emphasis

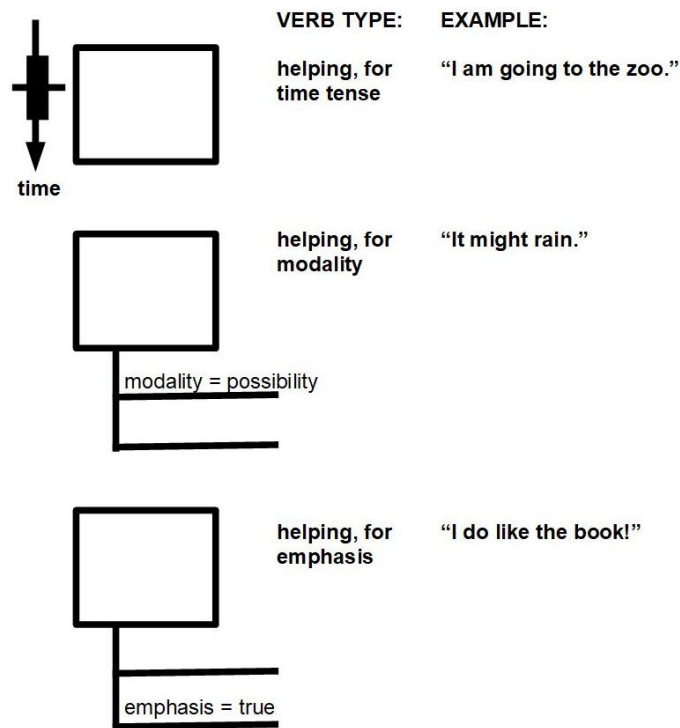


Figure 291: "Helping verbs" is a diverse class of verbs with different uses and representations in Tumbug.

From a Tumbug perspective, "helping verb" is an ambiguous concept based on syntax, not grammar, therefore most helping verb categories can be ignored in this study. See Figure 291. However, time tenses and modal verbs are deeper grammatical topics that are discussed in detail elsewhere in this document, each in its own dedicated section.

5. Regular verbs versus irregular verbs

The issue of whether a verb is regular or irregular is a syntactical issue, not a semantic issue. Since Tumbug deals with semantics, Tumbug *per se* does not consider whether a verb is regular or irregular, which means for example that there will be no written distinction between the concepts of "see" and "saw": the only difference is implied by the time shown at that event on the timeline (Time Arrow) as shown in Figure 292. The closest that Tumbug might approach this distinction is if: (1) software of some type were translating from Tumbug representation of a sentence in one language to a

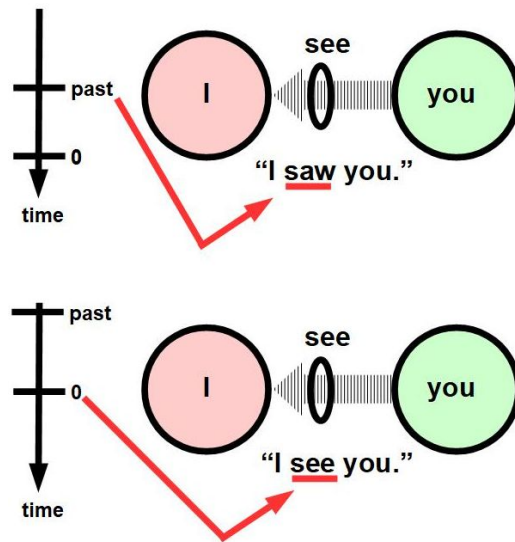


Figure 292: Tumbug ordinarily distinguishes between past and present via timeline, therefore Tumbug is not usually concerned with textual descriptions or their spelling differences since such text is redundant. Therefore the two diagrams are identical except for the positioning of times on the timeline.

different language; (2) Tumbug were being used to store conjugation rules.

6. Phrasal verbs

A phrasal verb is a verb that includes a preposition (e.g., for, with, to, up, across) and/or adverb to the verb, which together can create a new meaning that is completely different from the meaning of the original verb alone. Phrasal verbs are irrelevant in Tumbug, for the same reason that conjugations are irrelevant in Tumbug: these are spelling, phrasal, or syntactical differences that are independent of meaning. Some examples are listed in Figure 293.

A Tumbug diagram of the original verb will usually be very different than a Tumbug diagram of the same verb made into a phrasal verb.

7. Infinitives

As noted elsewhere, conjugations of verbs by time, number, and person are essentially meaningless in Tumbug since Tumbug fundamentally uses no text anyway; such conjugation information can only be inferred by the diagram or its attribute values, which also fundamentally use no text. Since the

to ask for
original verb = to ask
meaning of original verb = to pose a question
meaning of new phrase = to request

to put up with
original verb = to put
meaning of original verb = to place
meaning of new phrase = to tolerate

to talk down to
original verb = to talk
meaning of original verb = to speak
meaning of new phrase = to patronize

to lock up
original verb = to lock
meaning of original verb = to close a lock
meaning of new phrase = to secure

to cut across
original verb = to cut
meaning of original verb = to divide
meaning of new phrase = to traverse

Figure 293: Examples of phrasal verbs. Tumbug is concerned only with unique concepts, not with new meanings that have been attributed to groups of existing words by humans, therefore phrasal verbs have little inherent meaning to Tumbug.

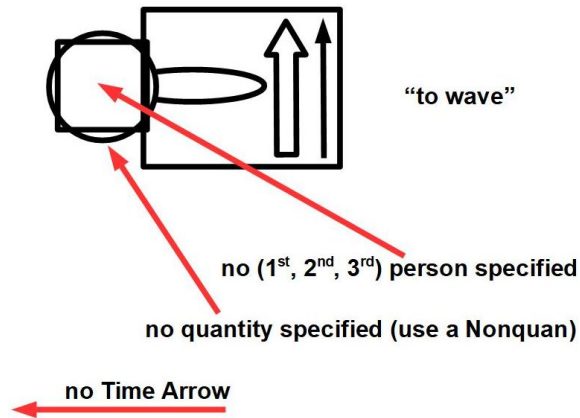


Figure 294: In Tumbug the infinitive form of a verb will show no tense (i.e., no Time Arrow), no quantity (i.e., a Nonquan is shown rather than an Object Circle or Location Box), and no person (i.e., no relative location of the agent is implied by the diagram). Here the arrows suggest an appendage moving upwards, which suggests the infinitive "to wave."

infinitive form of a verb is characterized by lack of time, number, and person, a generic Tumbug diagram that somehow depicts a given action alone is how an infinitive would appear in Tumbug, as shown in Figure 294.

8. Reflexive verbs

One definition of a "reflexive verb" is a verb whose direct object is the same as its subject. For example, a verb used with a reflexive pronoun in a sentence of the form "I ⟨verb⟩ myself" or "You ⟨verb⟩ yourself" are necessarily reflexive verbs, such as the sentence "I wash myself" as shown in Figure 295. Verbs that describe actions that are ordinarily done to oneself, such as "to wash" or "to shave," do not require a reflexive pronoun. In Tumbug, reflexive verbs are immediately visually recognizable because the action immediately circles back to the subject. This is a benefit since no confusion about the meaning of the sentence can occur, unlike in written English.

15.5.2 Active tense versus passive tense

Marvin Minsky mentioned active tense versus active tense (Minsky 1974, p. 33) via the three example sentences in Figure 296, Figure 297, Figure 298.

Tumbug works as beautifully on passive tense as it does on active tense.

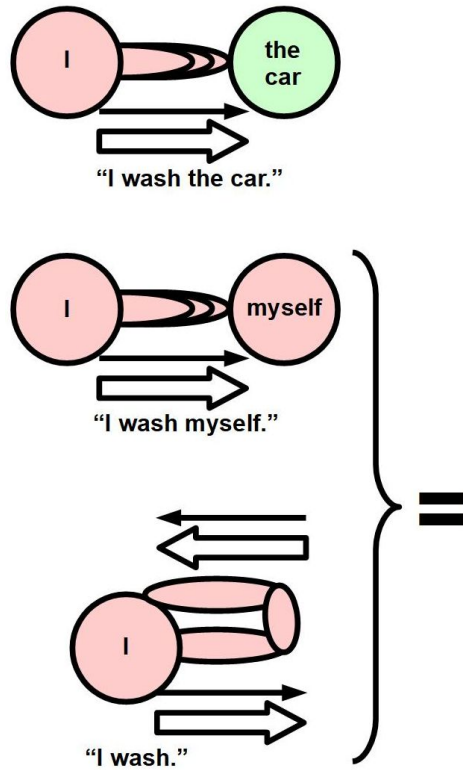


Figure 295: Top: "To wash" as a transitive verb. Bottom: "To wash oneself" or "to wash" as a reflexive verb.

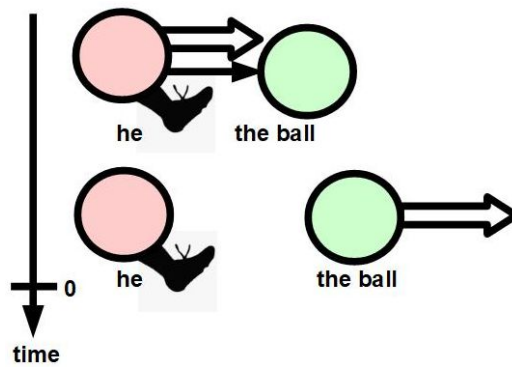


Figure 296: Tumbug for "He kicked the ball." (active tense)

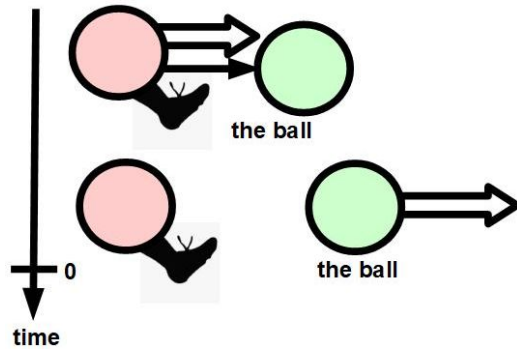


Figure 297: Tumbug for "The ball was kicked." (passive tense)

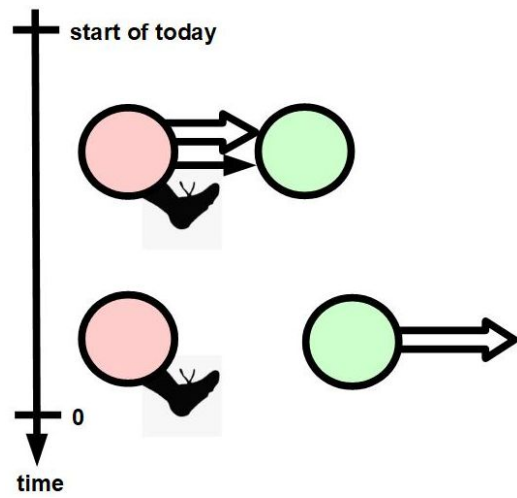


Figure 298: Tumbug for "There was some kicking today." (passive tense)

If one of the objects is not identified, a completely unlabeled Object Circle can be used, or a question mark "?" can either be placed inside an Object Circle, a lone question mark can be used, or an icon can be labeled with a lone question mark can be used. The representation method that is most consistent with the spirit of Tumbug is that of showing a completely unlabeled Object Circle for a sentence's Subject because some Subject must have caused the action: a missing Subject would imply that the Subject either did not exist or that action arose out of a void, or both, which is logically impossible. In addition, the placing of an unlabeled Object Circle in the diagram makes it less likely that such passive tense wording can be used to misdirect the reader, which is often the intent of passive tense. With this Tumbug convention any unaccounted-for object must automatically be brought into the diagram with an obvious presence that is difficult to ignore. For an example of this justification, in the sentence "The ball was kicked," the Merriam-Webster definition of "to kick" is "to strike out with the foot or feet," which means that a visual representation of "to kick" absolutely must contain some reference to a foot, which in turn implies the existence of some entity that had that foot as part of its body, an entity that was capable of striking out with that foot. This undefined entity is exactly the meaning of an unlabeled Physical Object Circle, therefore such a Physical Object Circle with a foot should be used, which has the automatic side benefit of making any implied unknowns blatantly obvious instead of being hidden via language.

15.5.3 Grammatical time aspects

The basic time ranges in which events are described in natural languages are, of course: (1) past, (2) present, and (3) future. Most natural languages add an additional complication to timelines called "aspect," however. The most common aspects are: (1) simple, (2) progressive, (3) perfect, (4) present progressive. "Aspect" describes whether a given event happened at a specific point in time, versus whether a given event happened over a period of time. Each of these cases is listed in Figure 299 with its overall description and associated Tumbug diagram. Note that these particular meanings of types of aspect are not always fixed—natural language is imperfect—but for any specific interpretation Tumbug will render them faithfully. For example, although "simple" aspect usually means at a single point in time, it can also refer to a cluster of points in time.

In this section the time spans that are of interest are emphasized using

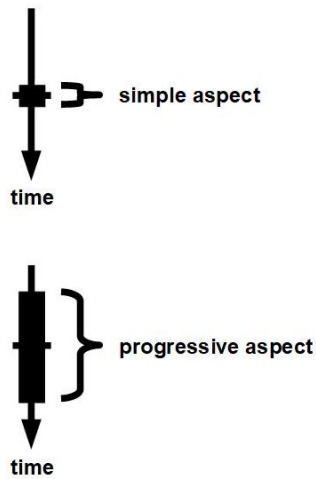


Figure 299: Aspect is largely only a matter of how long the interval of time is, over which an action happens.

a thickened Time Arrow in those regions. This practice is not a fixed part of Tumbug: braces or shaded regions can accomplish the same task of indicating any specific times spans of interest.

1. Simple aspect

The "simple" aspect means a single point in time, or sometimes a cluster of such points, shown here by the thick line that ranges over narrow period of time. See Figure 300, Figure 301, Figure 302.

2. Progressive aspect

The "progressive" aspect means the described action occurred over a

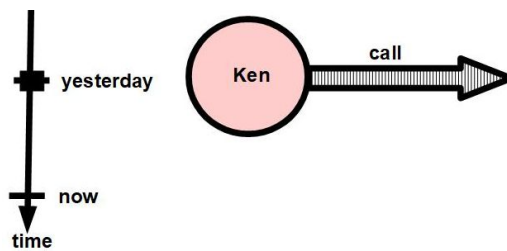


Figure 300: (in the past:) "Ken called (via an information line) yesterday."

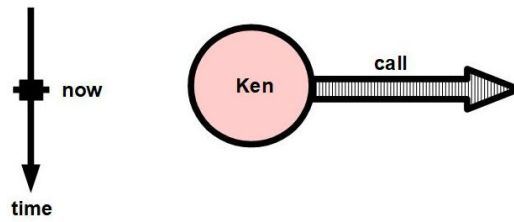


Figure 301: (in the present:) "Ken is calling (via an information line) now."

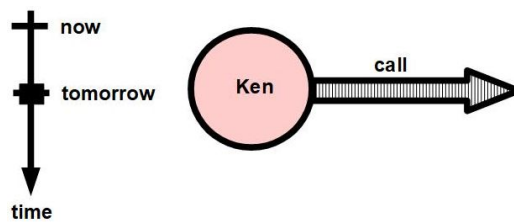


Figure 302: (in the future:) "Ken will call (via an information line) tomorrow."

range of time. This situation is shown here by the thick line that ranges over an extended period of time longer than a single moment. See Figure 303, Figure 304, Figure 305.

3. Perfect aspect

"Perfect" in grammatical context means "complete," meaning the described action has stopped. The "perfect" aspect means an ongoing action in the past was completed by a certain reference time.

See Figure 306, Figure 307, Figure 308.

Note that in the present perfect the reference time coincides with "now." This differs from the other two perfect tenses (past perfect and future perfect), where the reference time is always assumed to be after the time of the described action, and with both the described action and reference time on the same side of "now."

4. Perfect-progressive aspect

"Perfect" in grammatical context means "complete," meaning the described action has stopped. The "progressive" aspect means the described action occurred over a range of time. The "perfect-progressive" aspect means an action over a length of time was completed. See Figure 309, Figure 310,

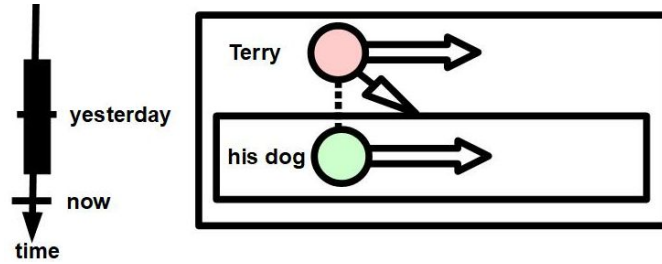


Figure 303: (in the past:) "Terry was walking his dog yesterday."

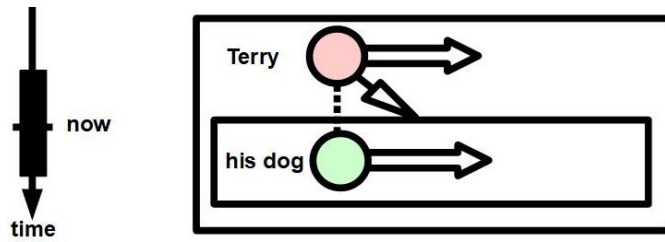


Figure 304: (in the present:) "Terry is walking his dog now."

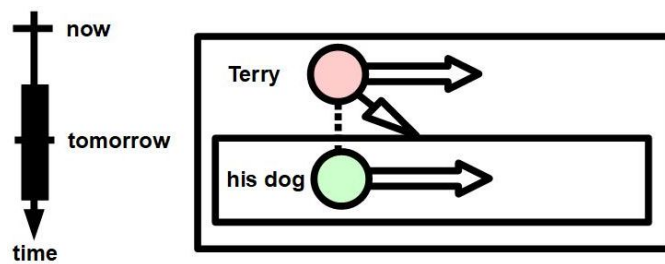


Figure 305: (in the future:) "Terry will be walking his dog tomorrow."

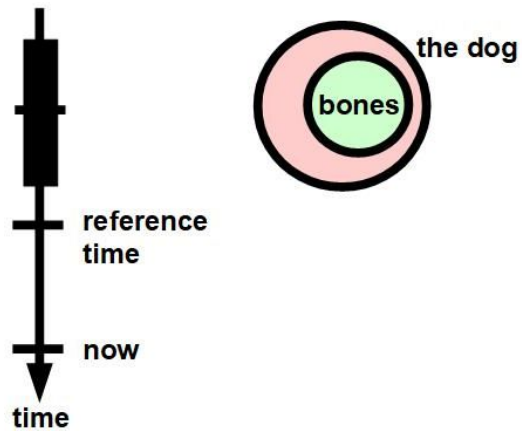


Figure 306: (perfect in the past / past perfect:) "The dog had eaten bones (by the time we got home)."

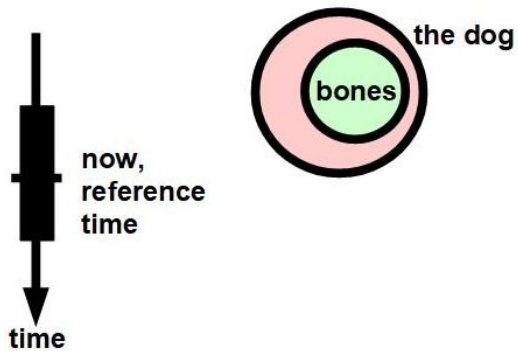


Figure 307: (perfect in the present / present perfect:) "The dog has eaten bones (by now)."

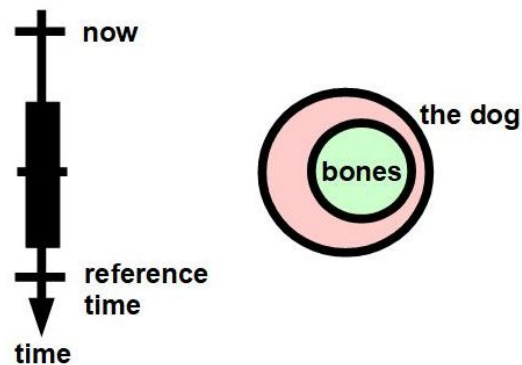


Figure 308: (perfect in the future / future perfect:) "The dog will have eaten bones (by the time we get home).

Figure 311.

Note that English is ambiguous in this aspect since this aspect does not necessarily imply that the described action will happen after the reference time. Some English instructors say the described action is "expected to continue," others say the described action "may continue." English obviously has some flaws for accurate scientific usage. The figures show that the thick line of event duration continues (via arrow head), which implies the "expected to continue" interpretation was intended. Figure 309, Figure 310, and Figure 311 are identical to the preceding figures except the event duration stops, which implies the "may continue" interpretation was intended.

The difference between perfect aspect and progressive-perfect aspect is summarized in Figure 312, Figure 313, and Figure 314.

5. Discussion of grammar

The preceding sections demonstrate that Tumbug can accurately represent all the time aspects of English grammar, so well in fact that certain ambiguities in English time aspects become very clear.

Note that Tumbug representations can be made much more thoroughly visual than they were in the aforementioned aspect diagrams. For example, instead of writing "yesterday," "now," or "tomorrow" on the timelines, any type of visual representation could be used instead, such as a clock, a calendar, numbers, or lengths of a cylinder that represents some multiple of time. Similarly, icons of humans can be used instead of C Object Circles. Also, details can always be added, such as a human holding a leash that

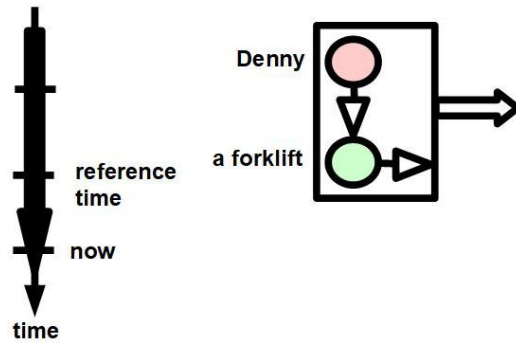


Figure 309: (in the past / past perfect progressive:) "Denny had been driving a forklift (and might have continued to do so)."

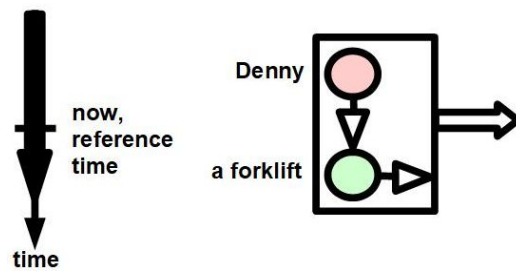


Figure 310: (in the present / present perfect progressive:) "Denny has been driving a forklift (and might still be doing so)."

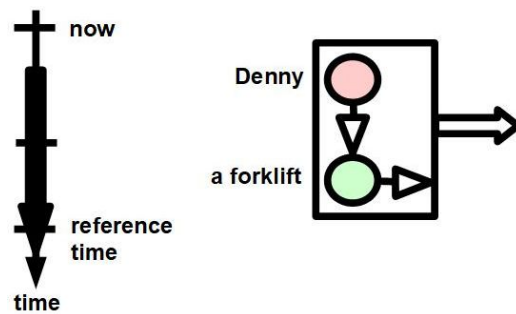


Figure 311: (in the future / future perfect progressive:) "Denny will have been driving a forklift (and might continue to do so)."

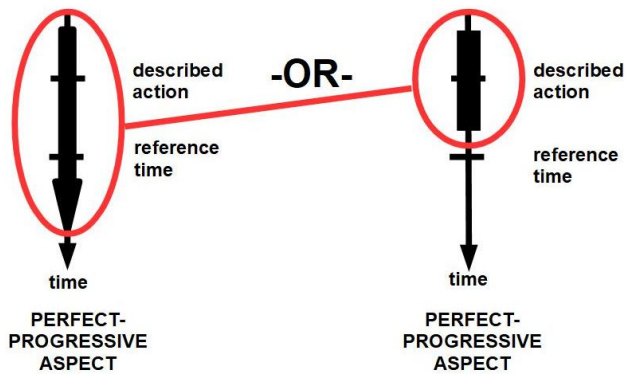


Figure 312: In unfortunate English language ambiguity, perfect-progressive aspect can mean either that the described action continued or that it stopped.

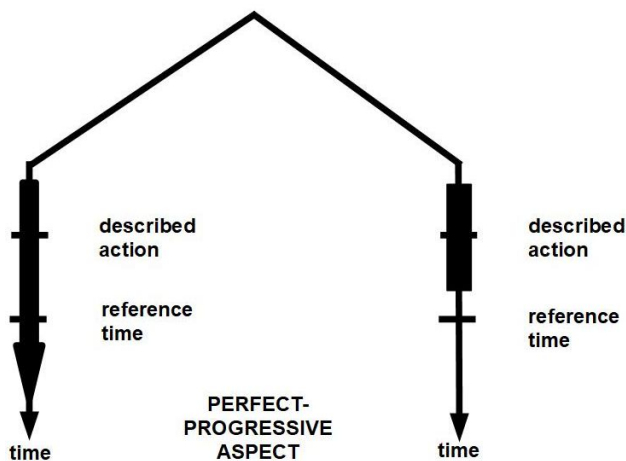


Figure 313: A Split Time Arrow can be used to include both possibilities of "perfect-progressive aspect."

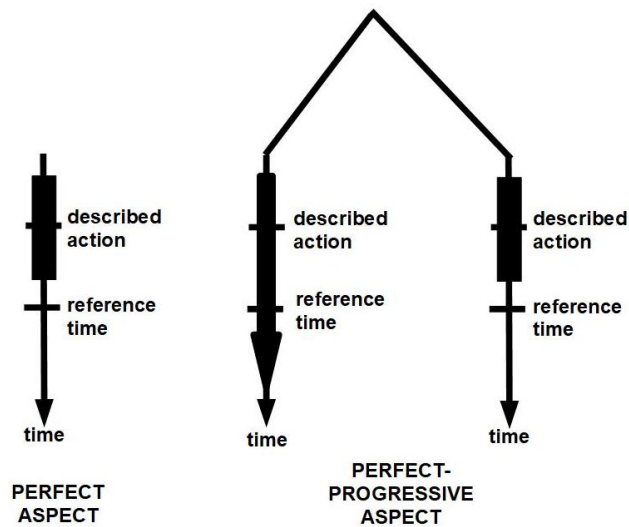


Figure 314: A comparison of the "perfect aspect" diagram with the Split Time Arrow diagram of "perfect-progressive aspect."

is around the dog's neck, which would presumably represent an additional force (represented by a Force Arrow) that constrains the dog's movement. Tumbug is ultimately intended to be entirely visual, and it has very nearly that capability already if the human diagrammer has the time to fill in the visual details.

15.5.4 Modal verbs

1. Lists

1.1. Simple lists

The following list is likely a complete list of the modal verbs of American English and British English, in alphabetical order:

1. be able to
2. can
3. could
4. had best (uncommon, spoken only)

5. had better (uncommon)
6. have got to (uncommon)
7. have to
8. may
9. might
10. must
11. needn't (uncommon)
12. ought to
13. shall
14. should
15. will
16. would

These modal verbs precede regular verbs, and convey the following modal concepts that modify the context of the regular verbs that follow them:

1. ability
2. advice
3. conditional - decomposes
4. formal directive
5. future
6. habit in past tense
7. habit, present tense
8. intention

9. likelihood/certainty
10. not necessary
11. obligation/correctness/rightness
12. offer/invitation/persuasion
13. permission
14. (polite) request
15. prediction
16. suggestion
17. willpower/intent (after "I" or "we" in formal British)

Unfortunately the modal verbs listed above are highly overloaded, meaning that one modal verb can have multiple meanings from the above list. The above list is a conglomeration of the words used to describe of concept meanings given by different authors on the Internet. Below is the nearly the same list of modal verbs, with most of their possible conveyed concepts added, each concept taken from the above list.

1.2. Modal verb \Rightarrow modal concept

1. be able to
 - (a) Mike is **able to** solve complicated math equations. (ability)
 - (b) Will you **be able to** walk the dog this afternoon? ((polite) request)
2. can
 - (a) She **can** swim underwater. (ability)
 - (b) You **can't** learn a new language if you don't practice. (conditional - decomposes)
 - (c) I **can** be with you at 9 a.m. (likelihood/certainty)
 - (d) **Can** I give you a hand? (offer/invitation/persuasion)

- (e) Yes, you **can** have an ice-cream. (permission)
- (f) **Can** you change this for a different color, please? ((polite) request)

3. could

- (a) He **could** run the mile in 4 minutes when he was younger. (ability)
- (b) If I **could** be President for one day, I would change the world. (conditional - decomposes)
- (c) I **could** make friends easily when I was younger. (habit in past tense)
- (d) Mary **could** have been better than her sister at ballet. (likelihood/certainty)
- (e) I **could** go for you? (offer/invitation/persuasion)
- (f) **Could** you pass me the salt please? ((polite) request)
- (g) **Could** I leave early today, please? (permission)

4. had best

- (a) You'd **best** leave it till Monday. (advice, informal)

5. had better

- (a) I **had better** not buy that coat. (advice, formal)

6. have got to

- (a) Drivers **have got to** get a license to drive a car in the US. (necessity)
- (b) I **have got to** be at work by 8:30 AM. (obligation/correctness/rightness)

7. have to

- (a) We **have to** wear a uniform at work. (obligation/correctness/rightness)

8. may

- (a) She **may** come later. (likelihood/certainty)
- (b) You **may** go now. (permission)

(c) **May** I speak to Mr. Jones please? ((polite) request)

9. might

(a) The train **might** be delayed. (likelihood/certainty)

(b) **Might** I have a glass of water please? ((polite) request)

(c) She **might** like a jewelry box for her birthday. (suggestion)

10. must

(a) It's snowing, so it **must** be very cold outside. (likelihood/certainty)

(b) Children **must** do their homework. (obligation/correctness/rightness)

11. needn't

(a) You **needn't** do your homework. (not necessary)

12. ought to

(a) You **ought to** have your car serviced before the winter. (advice)

13. shall

(a) Pupils **shall** not use the main entrance. (formal directive)

(b) This time next week I **shall** be in Scotland. (future)

(c) **Shall** we tell him? (offer/invitation/persuasion)

(d) We **shall** not be moved! (willpower/intent)

14. should

(a) You **should** stop smoking. (advice)

(b) He **should** be here in 5 minutes. (future)

(c) The government **should** reduce the sales tax. (ideal/preferred)

(d) **Should** I turn the heating on? (offer/invitation/persuasion)

(e) **Should** we invite Sarah and David? (possibility)

15. will

(a) This year Christmas **will** fall on a Monday. (future)

- (b) John **will** always be late! (habit, present tense)
- (c) She **will** come tomorrow. (intention)
- (d) You **will** need to show your boarding pass at the gate. (obligation/correctness/rightness)
- (e) Short hair **will** suit you. (prediction)
- (f) **Will** you take the dog for a walk please? ((polite) request)

16. would

- (a) Tom **would** do something like that, wouldn't he? (habit, present tense)
- (b) When I lived in Italy, we **would** often eat in the restaurant next to my flat. (habit in past tense)

In particular, requests such as "Could you pass me the salt please?" are at the polite end of a spectrum of imperatives, the other end of which would be a stark imperative such as "Pass me the salt." Prediction also involves a spectrum of likelihood between certainty and non-certainty, such as "The total will be 10" versus "Short hair will suit you." Note that the "permission" concept of some modal verbs is analogous to having a pass, whereas the "formal directive" concept (in the negative) is the opposite, and analogous to not having a pass, therefore these two concepts are related.

There are some interesting or important things to know about these modal verbs: (1) Consensus among linguists is that Future Tense always implies possibility, at least if the agents are people. (2) "To be able" seems at first impression like a synonym for "can" but a little-known English grammar rule renders this first impression untrue since the generality of the event being discussed dictates which of those two modal verbs must be used. However, some people maintain that either "to be able" or "can" can be used for a (polite) request, even though politeness usage is rarely mentioned in conjunction with "to be able." The table in Figure 315 includes these extra meanings for "to be able," and includes the implication of possibility with each use of Future Tense, though with these extra meanings put in parentheses.

In summary, the following concept meanings were excluded for the following reasons:

- Conditional: irrelevant since "if-then" has no effect on the concept included
- Desire: often seems implied, but not listed by anybody as implied, as in fact is not logically implied

Modal verbs may be redundant, especially in tense, due to prior coverage by Tumbug. For example, one meaning of "shall" is mere future tense: In British English "I shall" is exactly equivalent to "I will" in American English, yet Tumbug already is capable of showing future tense with a timeline so Tumbug does not need a modal verb to express future tense another way.

In this way the original list of modal concepts is reduced to the following shorter list of modal concepts:

1. Ability
2. Advice
3. Formal Directive
4. Formality (Formal, Informal)
5. Habit (Past Tense, Present Tense)
6. Ideal
7. Intention
8. Likelihood
9. Obligation
10. Offer
11. Permission
12. Possibility
13. Prediction
14. (polite) Request
15. Suggestion

16. Tense

17. Willpower

2. Table of modal verbs versus meanings

Under the assumption that the above list is definitive, a vector that contained all of these attributes would therefore capture all the nuances of every modal verb in English by inclusion or exclusion of selected modal concepts within the vector's components. For example, if the vector's components were defined as...

[Ability, Advice, Formal Directive, Formality, Habit, Intention, Likelihood, Obligation, Offer, Permission, Request, Possibility, Prediction, Suggestion, Will]

...then the modal verbs of American English and British English would be defined as in Figure 315.

3. Tumbug implementation

All of the above lists and summary table of modal verbs converted the ambiguities of modal verbs into discrete cases arranged as mathematical vectors (= each of the lists) and a matrix (= the table). Tumbug eventually must be able to visually represent any given modal verb, however, which is discussed next.

The table in Figure 315 can be regarded as a crossbar switch, where each row is a specific modal value, and each column is a modal concept. As each specific modal verb-with-meaning (row) is activated, one or more associated modal concepts (columns) become activated. If each row is implemented as an artificial neuron and each column is implemented as an artificial neuron, then the firing of one row causes the firing of one more columns as dictated by the connections in crossbar switch. The Figure 316 shows the specific modal verb-with-meaning "can (permission)" neuron activating the modal concepts "Per" (= Permission) and "Req" (= Request) neurons. Note that a device other than an artificial neuron could be used instead, such as a light-emitting diode (LED) as a signaling device.

Similar Figure 317 shows the specific modal verb-with-meaning "able to (ability)" neuron activating the modal concept "Abi" (= Ability) neuron and the modal concept "Req" (= Request) neuron.

When only the column header and its choices are considered, this combination of slots appears as a pegboard-like pattern, shown in Figure 318. Activation of a neuron within one of the vertically protruding banks of neurons could be implemented in various ways, such as: (1) using the vertical,

	[Abi	Adv	Dir	For	Hab	Ide	Int	Lik	Obl	Off	Per	Pos	Pre	Req	Sug	Ten	Wil]
be able to (ability)	[1	0	0	0	0	0	0	0	0	0	0	0	0	(1)	0	0	0]
be able to (request)	[0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0]
can (ability)	[1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0]
can (likelihood)	[0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0]
can (offer)	[0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0]
can (permission)	[0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0]
can (request)	[0	0	0	0	0	0	0	0	0	0	0	0	0	(1)	0	0	0]
could (ability)	[1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0]
could (habit)	[0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0]
could (likelihood)	[0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0]
could (offer)	[0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0]
could (request)	[0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0]
could (permission)	[0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0]
had best (advice, inf.)	[0	1	0	I	0	0	0	0	0	0	0	0	0	0	0	0	0]
had better (advice, for.)	[0	1	0	F	0	0	0	0	0	0	0	0	0	0	0	0	0]
have got to (necessity)	[0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0]
have got to (obligation)	[0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0]
have to (obligation)	[0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0]
may (likelihood)	[0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0]
may (permission)	[0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0]
may (request)	[0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0]
might (likelihood)	[0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0]
might (request)	[0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0]
might (suggestion)	[0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0]
must (likelihood)	[0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0]
must (obligation)	[0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0]
needn't (not necessary)	[0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0]
ought to (advice)	[0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0]
shall (directive)	[0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0]
shall (future)	[0	0	0	0	0	0	0	0	0	0	0	(1)	0	0	0	>	0]
shall (offer)	[0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0]
shall (willpower)	[0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1]
should (advice)	[0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0]
should (future)	[0	0	0	0	0	0	0	0	0	0	0	(1)	0	0	0	>	0]
should (ideal)	[0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0]
should (obligation)	[0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0]
should (offer)	[0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0]
should (possibility)	[0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0]
will (future)	[0	0	0	0	0	0	0	0	0	0	0	(1)	0	0	0	>	0]
will (intention)	[0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1]
will (request)	[0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0]
will (obligation)	[0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0]
will (prediction)	[0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0]
will (habit, present)	[0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	=	0]
would (habit, past)	[0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	<	0]
would (habit, present)	[0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	=	0]

Abi = Ability
 Adv = Advice
 Dir = Formal Directive
 For = Formality (spectrum points: I = informal, F = formal)
 Hab = Habit
 Ide = Ideal
 Int = Intention
 Lik = Likelihood
 Obl = Obligation
 Off = Offer
 Per = Permission
 Pos = Possibility
 Pre = Prediction
 Req = Request
 Sug = Suggestion
 Ten = Tense (spectrum points: < = past; = = present; > = future)
 Wil = Willpower

Figure 315: All the modal verbs of English and their meanings. A cell with parenthesized contents means this extra implication should exist.

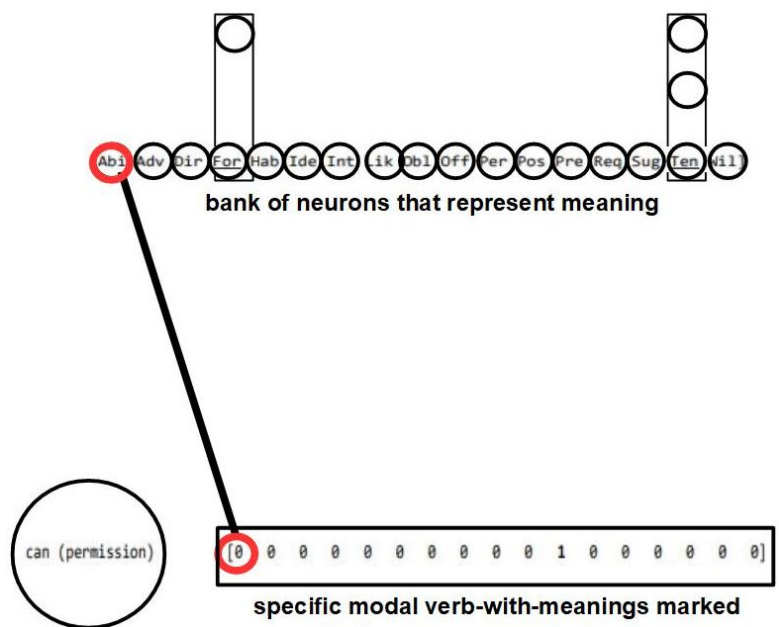


Figure 316: Tumbug implementation of modal verb-with-meaning "can (permission)."

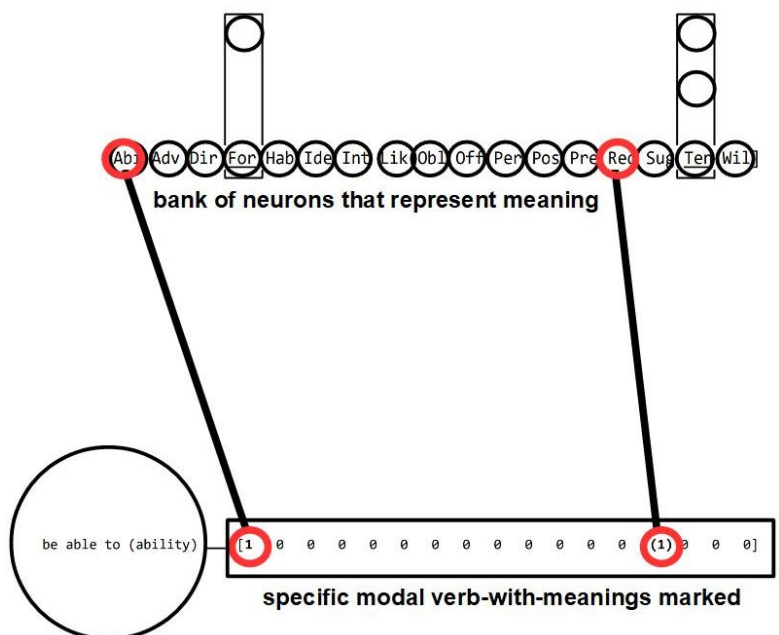


Figure 317: Tumbug implementation of modal verb-with-meaning "be able to (ability)."

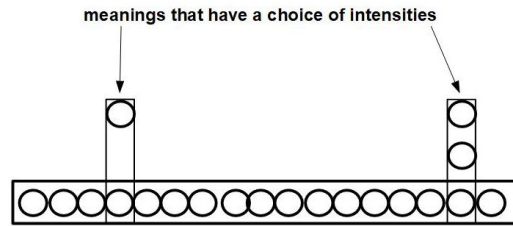


Figure 318: The skeletal template for any modal verb, which will represent a single modal verb when appropriate cells are filled in.

perpendicular bank of neurons as shown in the figure; (2) omitting the vertical bank of neurons, and using a neuron firing at a different intensity in the horizontal bank of neurons, such as firing weakly to represent Informal versus firing strongly to represent Formal.

When this shape is enhanced with activated neurons in unique locations, each activated location shown here in white, a uniquely colored variation of the pegboard pattern results. Each of these unique visual patterns can be considered a visual representation of a different modal verb.

Although such a visual pattern might seem to bear no resemblance to any image a human might associate with a given modal verb, this does not matter. The reason it does not matter is that the brain routinely uses maps that do not intuitively correspond to any map in the real world. For example, a spatial map exists across the somatosensory cortex that represents its person's body, roughly in the same shape as the body in one region (see Figure 319), and does (or at least could) contain a representation every part of the body that contains sensors, whether internal or external. Via this map, stimulation of a given set of neurons is inherently understood by the brain to represent signals from that corresponding part of the body, not understood as signals from neurons themselves. In fact, the phenomenon of "phantom limbs" has been hypothesized by Dr. Ronald Melzack to be caused by a "neuromatrix" that is an intermediate biological neural network that sometimes errs in the mapped locations that it forwards to the brain (Melzack and Wall 1965). There is no reason to believe that collections of neurons cannot represent nonexistent or abstract concepts such as modal verb meanings in the same way. Swirly arrays bridge the gap between smooth, irregular, biological structures, and regular, grid-like, computer data structures.

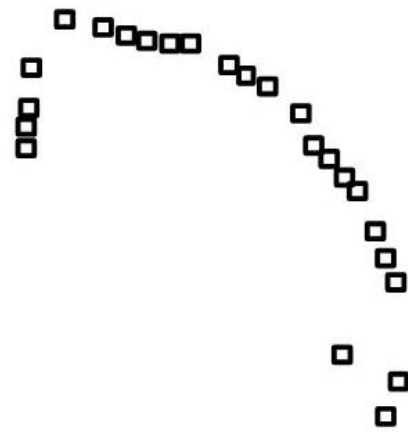
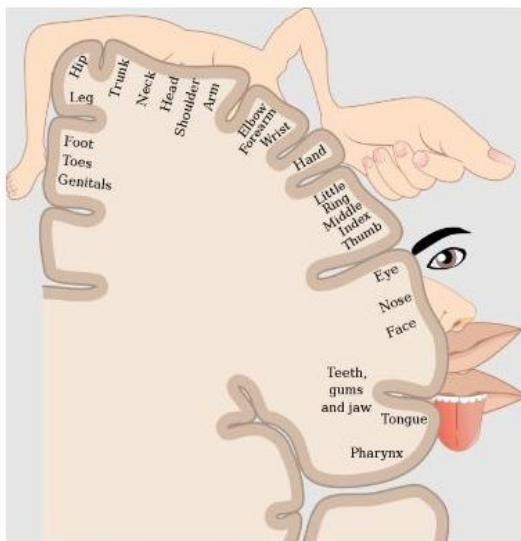


Figure 319: Left: A "body map" of a human as represented by a human's somatosensory cortex, with labeled regions. (Source: Course Hero, Inc.) Right: A Swirly Array whose cells spatially correspond to each labeled region on the left.

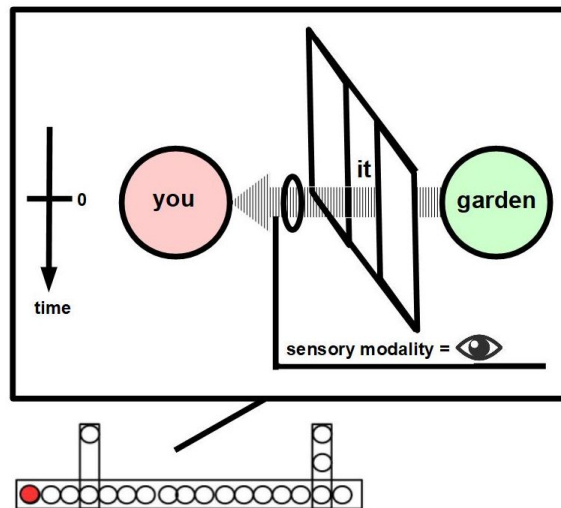


Figure 320: [31] Tumbug for "You can see the garden through it."

4. WS150 example: #31 (garden)

"[31] There is a gap in the wall. You can see the garden through it. You can see the garden through what? POSSIBLE ANSWERS: {the gap, the wall}"

In Figure 320, the entire diagram for "can (ability)" is used in place of text, which renders that modal verb-with-meaning a pure image. Similarly, an eye icon is used in place of the word "vision" after "sensory modality =".

Figure 321 shows which of the modal concepts use multiple choices for their attribute values, and shows additional attributes that could optionally be used in a choice-like or spectrum-like manner.

15.5.5 Propositional attitudes

1. Lists

A propositional attitude is a mental state held by an agent about a proposition. The presence of a propositional attitude in a sentence is signaled by a verb of attitude followed immediately by a clause that begins with "that," a clause that the verb governs. Some examples from WS150 are: "[41] I'm sure **that** my map will show this building..." has the verb "to be

<u>modal concept</u>	<u>Spectrum used?</u>	<u>Variable used in spectrum:</u>	<u>New spectrum possible?</u>	<u>New variable possible in new spectrum:</u>
Ability	no		yes	duration
Advice	no		yes	importance
Formal Directive	no		yes	officialness
<u>Formality</u>	yes	formality		
Habit	no		yes	tense
Ideal	no		yes	idealness
Intention	no		yes	conviction strength
Likelihood	no		yes	degree of likelihood
Obligation	no		yes	degree of necessity
Offer	no			
Permission	no			
Possibility	no			
Prediction	no		yes	degree of likelihood
(polite) Request	no		yes	
Suggestion	no			
<u>Tense</u>	yes	relative time		
Willpower	no		yes	willpower strength

Figure 321: All modal concepts currently covered by Tumbug, with emphasis on those that may involve a spectrum of values.

sure that” that signals belief. ”[133] Dr. Adams informed Kate **that** she had cancer...” has the verb ”to inform that” that signals knowledge. ”[109] Bill thinks **that** calling attention to himself was rude to Bert...” has the verb ”to think that...” that signals belief.

The propositional attitudes listed by Davis are (Davis 1990, ch. 8):

1. belief/believe/believes
2. knowledge
3. knowing whether and what
4. perception/perceive/perceives

Stuart J. Russell and Peter Norvig list some of these and more (Russell and Norvig 2010, p. 450):

1. belief/believe/believes
2. intends
3. informs
4. knows
5. want/wants

Various online sources list these:

1. assert
2. belief/believe/believes
3. command
4. consider
5. deny
6. desire
7. doubt

8. fear
9. hope
10. imagine
11. intend
12. judge
13. know
14. perception/perceive/perceives
15. want/wants
16. wish

2. Tumbug implementation

Iconic representation of propositional attitudes is difficult because of their artificial nature and because of their quantity. Unlike emotions and modal verbs, each of which constitute a relatively small quantity of basic categories that are often grouped by similarity, there exist at least dozens of propositional attitudes of extremely varied meanings. As a result, a slightly different way of representing propositional attitudes should be used for Tumbug. This document does not recommend any specific solution in the form of a master icon that shows all propositional attitudes (similar to the Robinson Icon), though in general it is recommended that the methods of organization used earlier in this document for emotions and modal verbs be used, namely (1) generalization of meaning, (2) alphabetization, and (3) dedication of one neuron per concept. For example, the following categories appear promising, and cover all the aforementioned listed propositional attitudes:

- emotional motivation: fear, hope
- general motivation: desire, intend, want, wish
- cognitive: believe, consider, deny, doubt, imagine, judge, know, perceive
- communication: assert, inform

- grammatical: command

Pro tem, in this document propositional attitudes are represented as text, namely as Label Strings, placed at the top of a C Aggregation Box to avoid the need to label a typically short, tilted line stemming from the agent. Tumbug uses a C Aggregation Box to hold a proposition, and the C Aggregation Box is given an attribute-value pair in text that contains the propositional attitude, such as "belief = true" or "desirability = false." Note that propositional attitudes need not necessarily have discrete true or false values. For example, "believe" and "desire" can span a wide range of applicability strength, and "imagine" and "perceive" are not binary at all.

Note that proposition attitudes are typically based on a multitude of incoming data from multiple senses, therefore the specific sensory modalities involved are hidden as an irrelevant detail, and similarly Motion Arrows representing the flow of objects (such as soup, for taste) or data (such as light, for vision) are omitted when propositional attitudes are used. Instead, single Attribute Line connects a C Aggregation Box that summarizes the impression, perception, belief, knowledge, etc. that the subject has about what is happening inside the C Aggregation Box.

Also note that a propositional attitude always involves an observer who is interpreting the situation depicted in the C Aggregation Box. This is analogous to a cathected emotion, as in the Robinson Icon, meaning that the observer is investing energy in an object, idea, or person, which in the case of propositional attitudes is a situation. Therefore a propositional attitude situation diagrammed in Tumbug must always contain: (1) the observer (Object Circle, though often unlabeled), (2) the situation (C Aggregation Box), (3) the attitude (label over the C Aggregation Box), (4) the link (Attribute Line) between the observer and the situation.

Figure 322 and Figure 323 demonstrate "belief" and "desire," two of the most common propositional attitudes in the English language. Memory is shown as a Data C Object Circle since memory is a virtual phenomenon. (If desired, memory could be shown housed inside of a skull that is represented by a Physical C Object Circle.) Memory is shown attached to the human icon with an Attribute Line, since memory is an attribute of humans. (If desired, a zoom ability could be used to position this memory inside the head of the human icon, where the skull is physically located, but the resulting diagram would be difficult to understand at a glance because of the very small sub-diagram.) Propositions are also virtual phenomena, so those are also shown

as Data C Object Circles. Two causations are shown via Causation Arrows, one caused by a person's memory automatically storing a new proposition, another caused by a person's mental models automatically assigning a belief level to the newly stored proposition. Note that nearly the same situation occurs in the human memory ("memory with mental models and tastes"), regardless of which of these propositional attitudes is involved.

3. WS150 example: #87 (dishwasher)

Figure 324 shows Tumbug representation of a part of WS150 question #87, which involves the propositional attitude "want." Note that the result resembles the method used to depict a personal fantasy that is used in Hollywood films, where the person doing the visualization is visualizing themselves at an approximate time period while doing what they want to do. Often films use a wavy fluctuation of the fantasy enactment to clue the viewer that the scene is fantasy, often done as a smaller frame within the main frame, whereas in Tumbug the fantasy scenario is contained in an Aggregation Box.

15.6 Tumbug as software

Tumbug was intended to be a diagrammatic KRM that can be drawn in 2D, but Tumbug would greatly benefit if Tumbug were implemented as a software tool. Some desirable effects of software implementation of Tumbug would be:

- Correlation Boxes could keep attribute values in sync, automatically, in the background.
- Motion Arrows could be eliminated because each intended moving object would actually move.
- Timelines could be eliminated because flow of time would already be present and obvious.
- Every intermediate position or state between two endpoints in time could be explicitly seen.
- Automatic storage of new experiences into memory could be implemented.
- Automatic assignment of propositional attitudes as attributes in memory could be implemented.

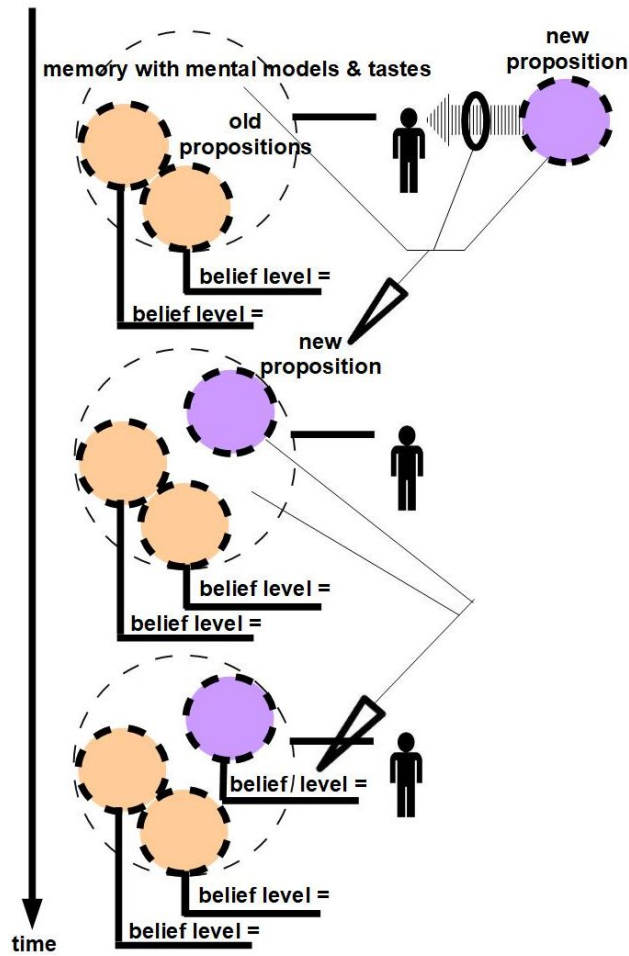


Figure 322: Belief: New proposition automatically enters memory, and is automatically assigned a belief level.

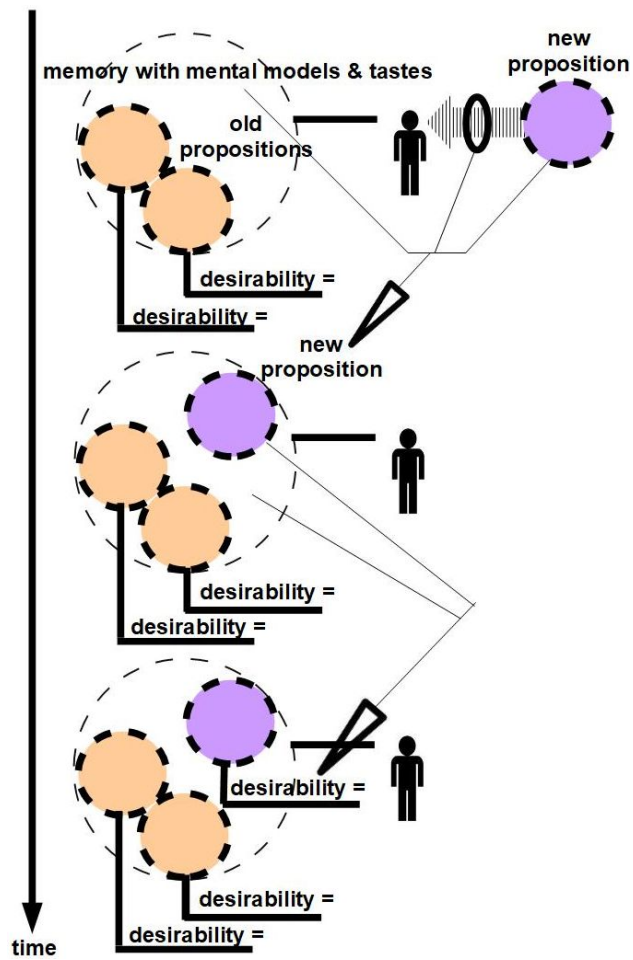


Figure 323: Desire: New proposition automatically enters memory, and is automatically assigned a desirability level.

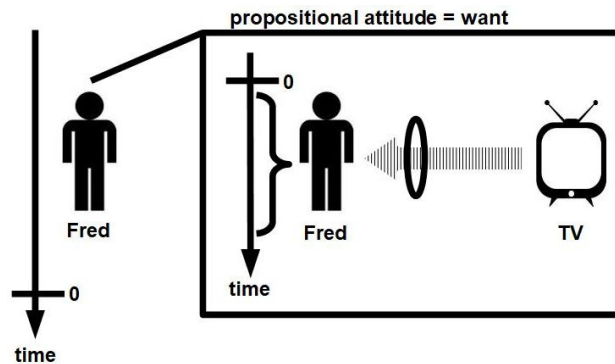


Figure 324: [87] "Fred wanted to watch TV."

- Any static zoomed diagrams could be omitted because zooming would be available on demand.
- Icon libraries could be extensive and readily available, which would enhance understandability.
- Libraries of scenarios of common actions (e.g., walking, talking, buying) could save writing time.
- 3D displays would be much more practical.
- Visual aids could be automatically placed on the diagram, such as to show alignment of objects, if desired.
- All diagrams created by a visual editor that are in a document could be easily updated in parallel, mostly automatically, to reflect a change in one diagram or in the text describing that diagram.

Tumbug is only a KRM and intended to be only a foundation for subsequent AGI architectures, but a software version of Tumbug that converted textual sentences into spatiotemporal icons might already have a few uses, such as the following:

- A written story could be immediately simulated, especially when writing a movie script.

- Contradictions, especially in legal cases, might be detected via conflicting attribute values.
- An aid to search engines, by allowing objects and actions to be combined to narrow the search, or to search for a specific situation using general terms that would ordinarily return too many matches.
- An aid to foreign language translation programs.
- Children learning to read could immediately relate words to visual objects and visual motions.

16 A few deep implications of Tumbug

16.1 SCOVA: an extension to OAV triples

It is remarkable that all of Tumbug's Building Blocks generalize to only five Basic Building Blocks (BBBs):

- (O) Object-like concepts. (objects and aggregations of objects)
- (A) Attribute-like concepts. (adjectives and adverbs)
- (V) Value-like concepts. (values, ranges of values, and wildcards)
- (C) Change-like concepts. (time, motion, causation, state changes, and functions).
- (S) System-like concepts. (systems composed of any mixed combination of the above types)

Since three of these concepts (O, A, V) are already addressed by standard OAV triples, this suggests that the OAV triple form of knowledge representation could be augmented if the other two concepts (C, S) of this document were added. With letters rearranged, this combination of concepts {O, A, V, C, S} can be spelled "SCOVA," which could be the name of a new KRM. The main difference between SCOVA and OOP would be that whereas change in OOP is often done by message passing between objects, change in SCOVA would be an additional component that would reduce

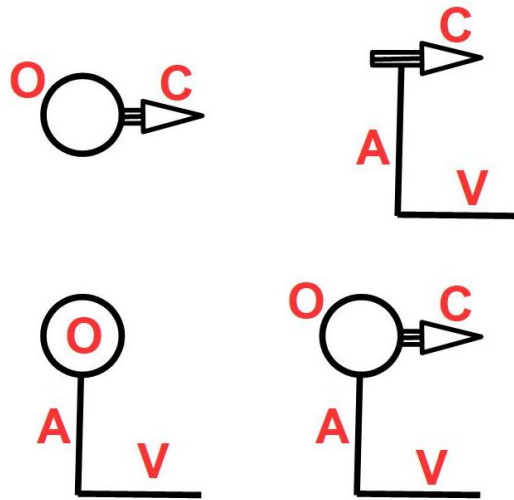


Figure 325: Some commonly mentioned, simple combinations of the SCOVA components. From left-to-right, starting on the top row, these are named in this document: OC, AVC, OAV, OAVC.

message passing and would eliminate functions ("methods") defined inside of OOP software objects.

Some of the simplest, most common, most useful, and most meaningful combinations of these give SCOVA components mentioned in this document are shown in Figure 325. The names used here attempt to keep the more familiar abbreviation "OAV" at the beginning of each longer abbreviation. "S" is not often used until Phase 2.

Side conjecture: Since the number of Basic Building Blocks of Tumbug, namely SCOVA, are five in number, and since the layers of the neocortex are six in number (I. molecular layer, II. external granular layer, III. external pyramidal layer, IV. internal granular layer, V. Internal pyramidal layer, and VI. multiform layer), it is conceivable that each layer of the neocortex is assigned one of the five components of SCOVA. If true, this would confirm the suspicion of Jeff Hawkins that the existence of those six layers is a major clue to intelligence (Hawkins 2004, p. 51, 69), since the implication would then be that the neocortex has somehow managed to implement what appear to be the five naturally arising building blocks of representation.

Side conjecture: Since the number of Basic Building Blocks of Tumbug,

namely SCOVA, are five in number, and since the number of main types of neocortical neurons are six in number (1. pyramidal cells, 2. fusiform cells, 3. stellate cells, 4. basket cells, 5. cells of Cajal-Retzius, and 6. cells of Martinotti), it is conceivable that each layer of the neocortex is assigned one of the five components of SCOVA.

16.2 Mathematical implications

The main mathematical implication found during this study is that mathematics is a disembodied form of Tumbug.

The Correlation Boxes of Tumbug make clear the relationship between Tumbug and mathematics: mathematics is concerned only about the numerical values of attributes of objects, whereas Tumbug puts those numbers into context and gives them meaning with respect to the entire diagrammed system. This is why a single number in mathematics has no relationship to the real world unless it is associated with a variable that is defined, especially if units of measurement are supplied (e.g., centimeters, joules, years, radians, mean, etc.). This is also why mathematics is considered abstract: it strips away all meaning and structure from the real world except for numerical values. This in turn suggests that mathematics is the wrong tool for implementing AGI, since AGI must simulate the real world with all of its structures, complexities, fuzziness, and vagueness. This, in fact, is said to be exactly the reason that microworld research (Partridge 1991, p. 95) of early AI failed, which included famous systems such as the SHRDLU program (which operated in the virtual blocks world), Shakey the robot (which operated inside a special, uncluttered building), FREDDY the stationary robot (which operated on a steerable platform), the SAM program (which operated in the virtual world of textual stories), and the FRUMP program (which operated in the virtual world of wire-service news reports).

As an example, Figure 132 is the earlier Tumbug representation of a bottle that poured water into a cup, and Figure 326 shows the Tumbug equivalent of the same system with everything stripped away except for the mathematical components.

Note that any sense of time is removed (i.e., the timeline is removed), any sense of the meaning of the variables is removed (i.e., the objects and their inner partitions are removed), and the physical contact between the objects is removed (i.e., the stream of water is removed). All the mathematical system knows is the values for the variables, how to transform one variable's

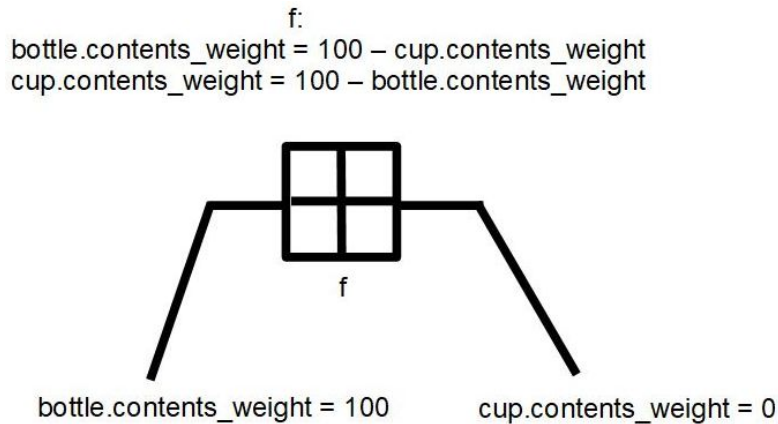


Figure 326: [24] A Tumbug diagram with a Correlation Box that has everything except the math stripped away.

value into the other variable's value (via the function f), and maybe the variable names (which in the computer are ultimately only numerical memory addresses, not even text, and certainly the computer does not understand how the text relates to anything in the real world). These observations lend credence to the various claims that mathematics is the wrong tool for describing the real world, and therefore the wrong tool for AGI, e.g., (Devlin 1997, p. 283), (Wolfram 2002, p. 821).

17 Insights from Tumbug

17.1 CD theory and Parts of Speech

The core of CD theory, described earlier in this document, is Roger Schank's insight that oftentimes several similar words refer to the same concept, but the words differ only because of the Part of Speech for which they are used. (In this document on this topic, the only Parts of Speech of interest are noun, adjective, verb, and adverb.) For example, "quick" is the word form when used as an adjective, but "quickly" is the word form when used as an adverb, "quickness" is the word form when used as a noun, and "quicken" is the word form when used as a verb. Obviously the same concept and same root word are referenced, viz. "quick," but the word becomes distorted

due only to English grammatical rules, a situation that is neither logical nor efficient when representing meaning in a diagrammatic form. (In speech, however, where a sentence diagram must be converted to a 1D temporal chain for audio communication, this is very efficient.) Some English words that function this way in at least three Parts of Speech are listed in the table in Figure 327.

It is understandable that a natural language would need to modify the root word to succinctly flag its use in the sentence, otherwise a grammatically correct sentence such as "Quick quickness quickly quickened" (see Figure 328 would be rendered as "Quick quick quick quick." However, in a diagram that represents meaning, this is inefficient practice since essentially the same word appears in quadruplicate. This suggests an interesting insight about language: objects, actions, and attributes can be described by the same concept. This means that a single concept, say represented by variable Q, can appear as a description of any object, action, or attribute, as illustrated in Figure 329.

The reverse situation generates another implication: The Part of Speech usage of concept Q depends only on the type of structure that Q is identifying. See Figure 330. In particular: (1) If Q is identifying a given C Object Circle then Q must be rendered as a noun in spoken or written language. (2) If Q is identifying a given Motion Arrow, then Q must be rendered as a verb in spoken or written language. (3) If Q is identifying a given Attribute Line then Q must be rendered as an adjective or adverb, depending only on whether the Attribute Line is attached to an C Object Circle or Motion Arrow, respectively. Otherwise, Q is one of the unlabeled core concepts of which Schank described as the foundation of CD theory.

There exists at least one caveat to the above observation, however: Since Motion Arrows are 2D rather than 1D, it must be specified whether the action described by a Motion Arrow is moving toward versus away from the state Q specified. In English the *de facto* convention is clear: Motion toward state Q is always implied, never in the reverse direction. For example, the verb "to embolden" means to become bold, not to stray from being bold. Similarly, the verb "to beautify" means to make something beautiful, not to cause something to be no longer beautiful.

<u>NOUN</u>	<u>ADJECTIVE</u>	<u>VERB</u>	<u>ADVERB</u>
-----ALL PARTS-OF-SPEECH FORMS EXIST (4/4)-----			
anger	angry	to anger	angrily
beauty	beautiful	to beautify	beautifully
boldness	bold	to embolden	boldly
care	careful	to care	carefully
clarity	clear	to clarify	clearly
coolness	cool	to cool	cooly
darkness	dark	to darken	darkly
enthusiasm	enthusiastic	to enthuse	enthusiastically
haste	hasty	to hasten	hastily
heat	heated	to heat	heatedly
impression	impressive	to impress	impressively
largeness	large	to enlarge	largely
length	lengthy	to lengthen	lengthily
lightness	light	to lighten	lightly
piercingness	piercing	to pierce	piercingly
power	powerful	to empower	powerfully
quickness	quick	to quicken	quickly
richness	rich	to enrich	richly
smoothness	smooth	to smooth	smoothly
softness	soft	to soften	softly
solidness	solid	to solidify	solidly
speed	speed	to speed up	speedily
strength	strong	to strengthen	strongly
success	successful	to succeed	successfully
-----NO ADVERB FORM EXISTS (3/4)-----			
shrinkage	shrunk	to shrink	-
trash	trashy	to trash	-
-----NO VERB FORM EXISTS (3/4)-----			
candidness	candid	-	candidly
coldness	cold	-	coldly
eagerness	eager	-	eagerly
foolishness	foolish	-	foolishly
greed	greedy	-	greedily
happiness	happy	-	happily
mercy	merciful	-	mercifully
neatness	neat	-	neatly
noise	noisy	-	noisily
obviousness	obvious	-	obviously
supremacy	supreme	-	supremely
tact	tactful	-	tactfully
truth	true	-	truly
wonder	wonderful	-	wonderfully

Figure 327: The rules of English often require a concept to change its concept name slightly, according to which Part of Speech it is adopting at any given moment.

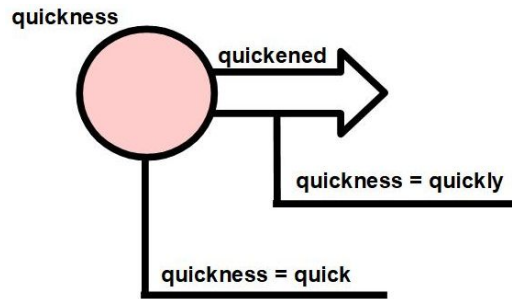


Figure 328: Tumbug for "Quick quickness quickly quickened," which is grammatically correct. Parts of Speech rules of English dictate that the concept's basic word must change form in many cases, however.

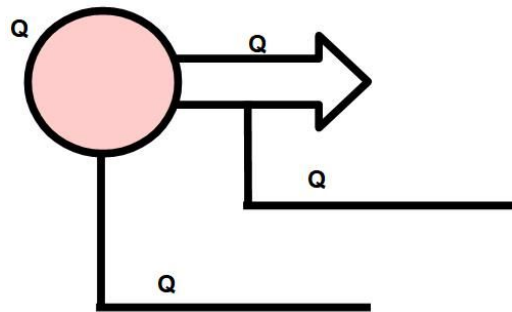


Figure 329: In a very general view, any single concept represented by variable "Q" can occupy any slot for any structure or attribute, regardless of Parts of Speech rules of English, and regardless of whether a word formally exists for every Part of Speech variation of that concept.

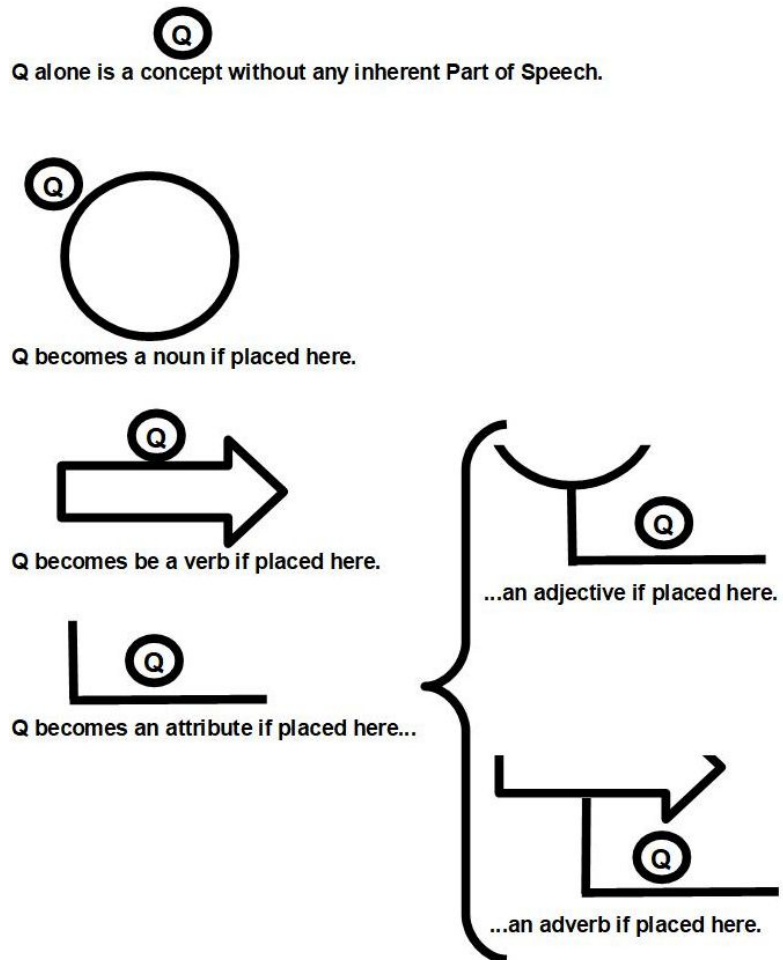


Figure 330: The Part of Speech associated with core concept Q is dependent only on the structure at which is appears.

18 Pros and cons of Tumbug

18.1 The main strengths of Tumbug

- Tumbug is likely the first purely visual representation ever used in computer science.
- Tumbug initiates a totally new, unexplored category of WS and CSR approaches.
- Tumbug clearly distinguishes between active tense and passive tense.
- Tumbug clearly distinguishes between simple aspect and progressive aspect.
- Tumbug clearly distinguishes between perfect and perfect-progressive aspect.
- Tumbug has better representational power than CD theory.
- Tumbug appears to implement the "universal language representation" needed in natural language processing (NLP).
- Tumbug closely approaches the ability of a visual simulator, therefore probably also closely approaches the way that humans think about events.

18.2 Possible weaknesses of Tumbug

1. It is not clear in a Tumbug diagram with a timeline which phases of the event should be documented, especially in how much detail. For example, an eating (INGEST) action for a human usually involves details such as chewing and swallowing, but these details are probably

not important enough to document. As another example, reaching the top of the stair for a human usually involves traversing each step along the way, which is likely an irrelevant detail for most purposes.

2. The functions currently allowed in the Correlation Boxes are not required to be invertible functions. For example, the allowed function $y = x^2$ over the real numbers is not invertible because then $x = \pm\sqrt{y}$. This allowance is currently assumed to be necessary since general intelligence would need to be able to access the total picture of how a function (or system) behaves, and would presumably be able to compensate for the many-to-one maps, or to decide on its own which values to accept from multiple possibilities. However, some drawbacks of allowing arbitrary functions are: (1) increased complexity of the mapping, (2) lack of clear-cut direction of correlation (viz., positive or negative), (3) possible inability of the system to make rapid, accurate guesses about the type of correlation involved between newly encountered variables because of the extra caution the system must exercise in order to remain general, (4) possible inability of the system to fill in missing functions by unsupervised examination of its own newly stored knowledge.
3. There can probably never be a canonical form of every sentence that is to be rendered in Tumbug. Even if the shortcut notations, such as for motion or neglect of a C Aggregation Box, are ignored, there still may not be a clear-cut standard form for some reason. There may be, but this is not clear. A convincing argument that canonical forms are difficult for any KRM to produce is given by William A. Woods (Woods 1975, pp. 16-21).
4. Several problematic situations occur when attempting to represent situations spatially. For example, how would one diagram "a small cluster of points in time" along a timeline without showing an exact number of points in the representation, and without showing their exact temporal order? Or how would one diagram a list of unsorted numbers in a generic way, without showing a specific number of numbers, or any of their values? How would one show in a generic way that a given set of jigsaw puzzle pieces needed to be placed together tightly but without forcing, and without showing the number of puzzle pieces or their shapes? The author has solved a number of such representational problems, therefore Tumbug can represent at least some such complications,

but that topic is out of scope for this document. More generally, such problems qualify as representation of mathematical concepts, which has been barely addressed in this study.

5. Some Tumbug icons are difficult to generalize easily from 1D to 2D. For example, a 1D sound broadcasting line is difficult to draw as fanning out into 2D space from a point. Similarly, a moving appendage such as an arm is difficult to describe with straight arrows, since most likely the appendage will swing on joints, which implies usage of arrows swinging through an angle. Most such problems could be resolved sufficiently with a software implementation of Tumbug without too much difficulty, but such software does not exist yet.
6. The differences and overlap of action verbs and stative verbs is not completely clear. For example, the verb "to eat" would probably be most accurately classified as an action verb since eating involves food moving to the mouth, the motions of chewing and swallowing, and the food moving to the stomach. However, from a wider perspective the detailed motions of eating are not very important. For example, some animals such as starfish eat by egesting their stomachs around their victims instead of using their mouths to hold and chew food, so overall the act of eating is to change the state of the eater and food so that the food becomes located inside the eater for nutrition purposes. Also, a phrase such as "the photocopier ate my document" is immediately understandable, despite not involving any living organism, mouth, or stomach. The author's prediction is that Marvin Minsky may have been correct: it is very possible the brain uses several types of KRMs, including motion diagrams, state diagrams, and verbatim images, and that the brain links these representations in a coordinated way through learning.

18.3 Programs writing programs

A KRM like Tumbug is probably the most promising approach to provide a future means for machines to write computer programs or self-modifying code because Tumbug's ability to store spatiotemporal images verbatim reduces information lost in the transition from the real world to the virtual world. Currently, the real world consists of spatiotemporal images that need

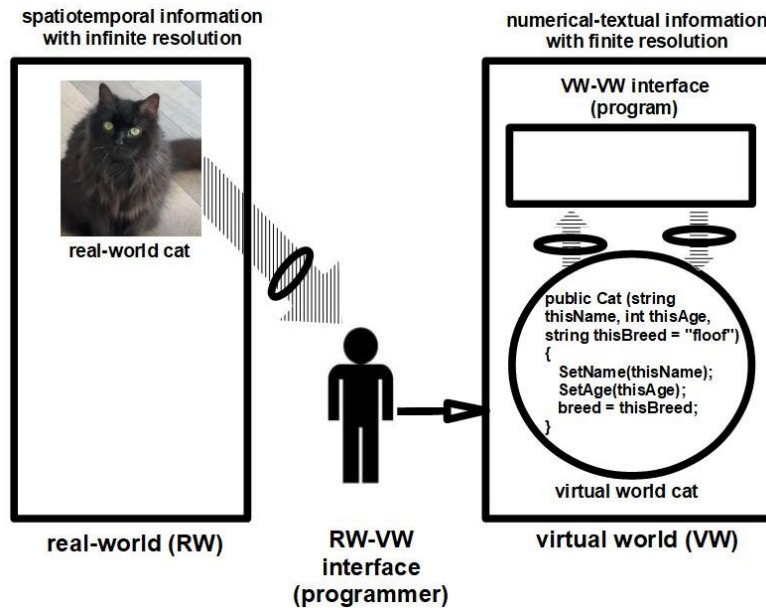


Figure 331: Programs struggle to write other programs partly due to their lost spatiotemporal information. As a result, programs need a programmer to cause a program to exist in the virtual world, based on the programmer's ability to understand the real world well enough to abstract it and to represent that abstraction.

to be converted to a numerical-textual format that the virtual world of digital computers can store and manipulate easily, which typically requires time to be discretized, images to be discretized, details to be omitted that initially seemed irrelevant to the programmer (but that may be deemed important later), shape information to be lost, and cause-effect information (that only a human would understand well) to be lost, as shown in Figure 331. Currently, computers cannot truly understand the real world, even when outfitted with video cameras, radar, and lidar. Even with multisensory input, computers cannot make commonsense conclusions about occluded objects or about interactions between physical objects. Computers also suffer additional problems from multisensory integration.

In contrast, a machine that could store information in the same spatiotemporal organization as the real world would automatically eliminate the programmer, at least as a data type translator, as shown in Figure 332.

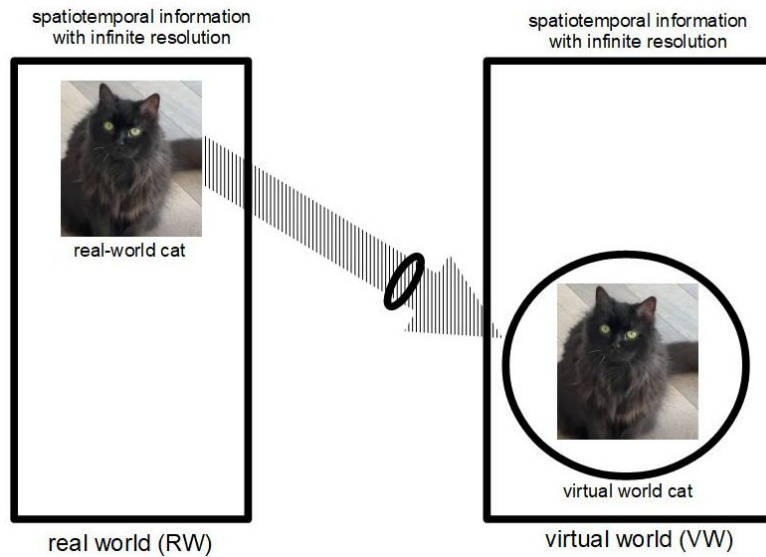


Figure 332: Programs whose representation matched that of the real world would not need a programmer.

That would free the programmer to focus on writing behavioral programs instead of also needing to convert between data types. The language of the real world obviously consists of spatiotemporal images across various sensory modalities, not abstractions like numbers, text, and algorithms. An immediate consequence of having such an internal representation that matched the representation of the real world is that presumably a program could then write other programs as shown in Figure 333, which could be considered the ultimate goal of computer programming.

19 Discussion of Tumbug

The two main categories of approaches to ANI are symbolic AI and sub-symbolic AI. Symbolic methods refer to text and are typically implemented with rule-based expert systems, logic programming, and semantic nets. Sub-symbolic methods refer to non-text and are usually implemented with neural networks. Tumbug is closer to a symbolic approach, but uses icons instead of symbols. Since Tumbug's icons have great freedom of placement, unlike

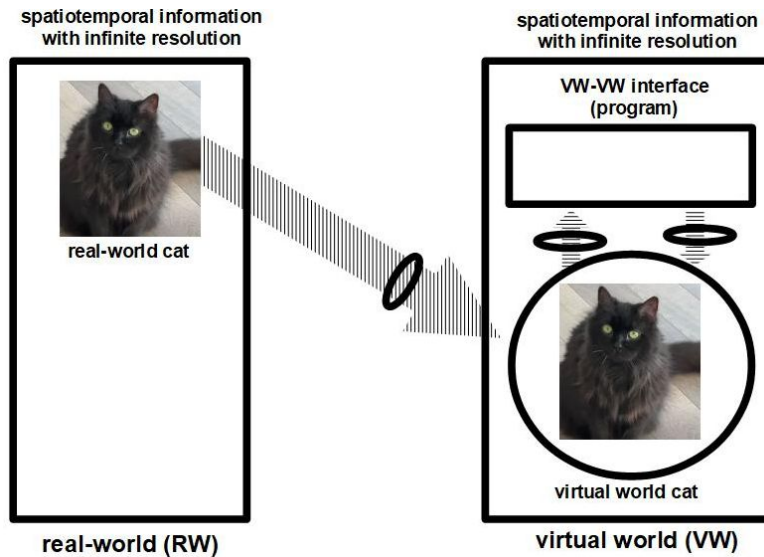


Figure 333: A virtual world that matched the real world could host a program to write other programs.

text, Tumbug is more flexible than text, and therefore more powerful than text for representation. Thus Tumbug represents a new category of approach that has not been pursued, which in itself seems promising.

Was the design of Tumbug intended to imply that the brain is literally moving around circular icons? No. The circular icons are merely the simplest pictorial concept of a generic "thing." In practice all that would be needed to implement a circular icon would be a fuzzy cluster of one to several neurons whose summarized attributes include coordinates of the thing they represent, combined with a possibly crude method of distinguishing one location (or state) from another. In this sense, Tumbug circular icons could be considered part of a fuzzy system, as in fuzzy logic, where the exact size, shape, boundaries, and curvature of the icons are irrelevant.

Tumbug can be considered a fuzzy system, but the fuzziness is created through simpler diagrams, not membership functions. In a sense, $\langle \text{percentage of fuzziness} \rangle = 1 - \langle \text{percentage of detail present in a diagram} \rangle$.

Is not the design of Tumbug's representation of modal verbs and emotions spatially arbitrary? Yes, but so are biological systems. The same patch of skin with pressure sensors can be placed anywhere on the surface of the

body, but in one position it could represent pressure on a foot whereas in another position it could represent pressure on the neck. In biological systems the only difference in meaning is based on that patch's location on the body map that is stored in the cortex. In a computer, labels substitute for locations in such a body map. For virtual concepts such as modal verbs, such a design suggests that humans can physically "feel" the difference between two different modal verb meanings using the same hardware that humans use to feel the difference between two physical body locations.

The late Marvin Minsky may or may not have agreed with the approach being used by Tumbug. Although Minsky strongly advocated emphasis on KRMs as the foundation of AGI, his recommended approach was to *combine existing* KRMs and to focus on the management system that would coordinate those KRMs (GBH Archives 1990), whereas the Tumbug approach has so far been to find a single, universal KRM that could represent every possible concept, and to eschew the unnatural, nonvisual KRMs that have been developed so far for use only in digital computers. In Pei Wang's terminology, Minsky's approach may have tended toward the hybrid approach whereas the Tumbug approach definitely takes the unified approach. However, Minsky's opinion still retains its merit: note that Tumbug has naturally gravitated toward appending specific computer science KRMs such as SCOVA + state diagram as in Figure 203, and state diagram + image as in Figure 138. If such lower level KRMs are what Minsky envisioned, then Tumbug is already tending toward Minsky's envisioned "missing link" of AGI.

Conjecture: Humans have a part of the brain that allows humans to feel virtual concepts in the same way that a part of the brain feels physical sensations. This conjecture fits Jeff Hawkins' mentioned observation that the same cortical structure is used throughout the brain, regardless of sensory modality. Tumbug suggests that this observation can be generalized to state that the same cortical structure might be used for virtual concepts as is used for sensory modality, i.e., that understanding a virtual concept is architecturally equivalent to understanding a sensory modality.

20 Summary

The main premise of this project is that if the proper KRM is used then the most difficult parts of AGI system design are automatically and greatly simplified. The proposed KRM for that purpose is a visual, iconic system called

Tumbug that was designed by the author. Tumbug is the research topic of this Phase 1, which is part of a larger research project that aims at producing full AGI. The full envisioned research project has five phases that are collectively called The Visualizer Project. Older but unpublished results from initial excursions into Phase 2 and Phase 3 algorithms appear to confirm the premise that Tumbug is a promising foundation for AGI. Tumbug consists of about 30 Building Blocks (icons) that generalize to only five Basic Building Blocks, which turn out to be essentially the three components of OAV triples, namely Object (O), Attribute (A), and Value (V), plus two additional components called Change (C) and System (S). Change is a general concept that includes concepts such as time, motion, force, and cause-and-effect, and System is any legally assembled combination of Tumbug's Building Blocks.

By using a single pictorial KRM that consists of only about 30 Building Blocks, Tumbug can represent an extremely wide span of topics in mathematics, logic, algorithms, physics, human motivations, and human activities. Tumbug is a radically different type of KRM in that Tumbug can be made fully visual without use of text or numbers, which sets Tumbug apart from all other AI approaches such as expert systems, neural networks (whose units require labeling with text), cellular automata (whose units require labeling with text), and any variations of these. In short, Tumbug can probably visually represent any topic or concept that the average person is likely to encounter. This unified representation method could allow various computer systems, whether based on natural language, data base records, numbers, computer code, knowledge graphs, or images, to be integrated without the need for conversion at their interfaces. Several examples taken from sentences of the Winograd Schema are shown. For additional practical applications, Tumbug seems to be particularly well-suited for natural language translation, and possibly also for universal computer code representation in the style of .NET.

Tumbug appears to be an improvement on the Conceptual Dependency KRM of Roger Shank in that Tumbug's Building Blocks are far more universal and general, such as objects, motions, and attributes rather than detailed and largely human-based actions such as ingest, expel, and grasp. Tumbug could be the basis of "The Language of Thought" that Jerry Fodor conjectured in 1975, a hypothesized language often called "Mentalese."

<u>re-</u> <u>search</u>	<u>KRM</u>	<u>algorithm</u>	<u>algorithmic</u>	<u>amount of</u> <u>training</u>	<u>milestone</u> <u>that would be</u>
<u>phase:</u>	<u>needed:</u>	<u>needed:</u>	<u>complexity:</u>	<u>time needed:</u>	<u>achieved:</u>
→ 1	Tumbug	none	none	none	KRM
2	Tumbug	KM	low	none	programmable non-SR CSR
3	Tumbug	SR	low	none	programmable CSR
4	Tumbug	learning	low?	very high	trainable CSR
5	Tumbug	visualizer	low?	none	AGI

AGI = artificial general intelligence
 CSR = commonsense reasoning
 KM = knowledge matching
 KRM = knowledge representation method
 SR = spatial reasoning

Figure 334: The research plan. This document completes Phase 1. The developed KRM from this phase (viz., Tumbug) will be used in all expected future phases of this research.

21 Future Research Plan

21.1 The Visualizer Project

A "visualizer" is defined here as a type of processor whose native KRM is imageoids, in contrast to a "computer," whose native KRM is numbers. Therefore computers and visualizers are different processing architectures that each have their own unique niche of problems that they can solve particularly competently.

This article completes and documents Phase 1 of a 5-phase research project called the Visualizer Project that aims to produce AGI in the form of a design for a visualizer. Tumbug is the foundation of all the phases that are expected to follow. The author believes that the design of a *bona fide* AGI system will be produced by the end of the fifth phase, though probably not yet coded at that time. Figure 334 shows all the anticipated phases of the Visualizer Project.

One inherent limitation of Phase 1 is it is difficult to state when a collection of Building Blocks is "complete" because there is no theoretical limit to the number of Tumbug Building Blocks, only a practical limit, usually based on the frequency of usage within a given domain, and also there has not been a need in Phase 1 to fully explore some recently introduced Building Blocks.

A few Tumbug Building Blocks were introduced in this document largely because they are expected to be needed sometime in future, especially for spatial reasoning (especially Correlation Boxes and 1D Markers), for reducing textual descriptions from Tumbug (especially Value Bars), for coding implementation (especially 0D Markers and Zoom Boxes), for a learning algorithm (especially Data Set Boxes), or for mathematical theorems (especially C-A Aggregation Boxes), but so far there has not been sufficient need to justify the time required to thoroughly explore those constructs.

It is expected that the need for several new Building Blocks will arise from the need to represent certain concepts from mathematics, especially from calculus (concepts such as "continuous" and "limit"), group theory, topology (concepts such as "connectedness"), probability (concepts such as "likelihood," "heuristics," "permutation," and "combination"), statistics (concepts such as "distribution"), algebra (concepts such as "invertible function"), and geometry (concepts such as "alignment," "area," "volume," and "dimension"), as well as from puzzle solving (concepts such as "ordered," "random," "path," and "to flush fit") and science/engineering topics in general (concepts such as "density," "cluster," "complexity," "independent," "coupled," and "representation").

21.2 Roadmap of the next phases

The Visualizer Project is not just a theoretical conjecture with no results and no motivating clues. The following subsections describe existing documentation, existing discoveries, and expectations of the next phases.

21.2.1 Phase 2 (non-spatial reasoning)

The basic algorithm for Phase 2 was already developed and documented in 2022 (Atkins 2022) with several complete examples diagrammed step-by-step during the CSR solution process. That article was not accepted for publication, however. In that year the author believed that the Tumbug-SOAV approach was already capable of solving 82% of WS150 problems, namely all of WS150's Non-spatial Reasoning Problems. Now that an extensive Tumbug introduction exists in the form of this Phase 1 document, presumably the older article describing the algorithm needs only to be rewritten with finer details, preferably also with more examples, without needing to include a description of Tumbug as well, in order to complete Phase 2.

21.2.2 Phase 3 (spatial reasoning)

A few Phase 3 results already exist, as well. The 2022 Tumbug article already mentioned the discovery that two of the nine Spatial Reasoning Non-Algorithmic problems from WS150 appear to be solvable by a more generalized version of Tumbug. If this generalization can be generalized even further, then the 82% solvability of WS150 problems by Tumbug would increase to at least 97%, probably to 100%. An article documenting the more general algorithm would likely use the same organization as a newly revised Phase 2 article.

21.2.3 Phase 4 (learning)

It is likely that the needed learning algorithm will ultimately be based on the Descriptive Boxes described in this document. A bridge needs to be made from the real world, currently representable in Tumbug by Verbatim Boxes, to the semi-computerized format of Descriptive Boxes, in order for Tumbug to work directly with real-world data without the need of a programmer. The author has already developed visual versions of several spatial prepositions such as "near," "above," "to," and "inside," so this phase definitely has the potential to move quickly, as well.

21.2.4 Phase 5 (visualizer)

A visualizer will need several additional components and techniques even if trainable CSR were produced. The author has already published one of these techniques and has publicly discussed a second technique that is believed to be needed. Additional components of a visualizer were already described in a research proposal written by the author in 2014, but that proposal was not awarded.

22 References

Allerton, D.J. 2006. "Verbs and their Satellites." In *The Handbook of English Linguistics*, edited by Bas Aarts and April McMahon, 146-179. Malden, MA: Blackwell Publishing.

Atkins, Mark. 2000. "S-96: A Semantic Net Implemented With Synchronized Neurons for Binding and Inferencing." Ph.D. dissertation, Florida Tech, Melbourne, Florida.

Atkins, Mark. 2019. "Two Approaches Toward Graphical Definitions of Knowledge and Wisdom." DaKM 2020 conference, Vienna, Austria.

Atkins, Mark A. 2022. "A promising visual approach to solution of 82% of Winograd Schema problems via Tumbug Visual Grammar." Unpublished.

Atkins, Mark A. 2023. *Important things to know before learning any foreign language*. Unpublished.

Baez, Albert V. 1967. *The New College Physics: A Spiral Approach*. San Francisco: W. H. Freeman and Company.

Bailey, James. 1996. *After Thought: The Computer Challenge to Human Intelligence*. New York, NY: BasicBooks.

Beckmann, Petr. 1971. *A History of Pi*. New York: The Golem Press.

Berthoz, Alain. 2000. *The Brain's Sense of Movement*. Cambridge, Massachusetts: Harvard University Press.

Brooks, Rodney A. 1992. "Intelligence without representation." In *Foundations of Artificial Intelligence*, edited by David Kirsh, 139-159. Cambridge, Massachusetts: The MIT Press.

Cercone, Nick, and Gordon McCalla. 1987. "What is Knowledge Representation?" In *The Knowledge Frontier: Essays in the Representation of Knowledge*, edited by Nick Cercone and Gordon McCalla, 1-43. New York: Springer-Verlag.

Chang, Le, and Doris Y. Tsao. 2017. "The code for facial identity in the primate brain." *Cell*. 2017 June 01; 169(6): 1013–1028.e14.

Chubb, Charles. 1995. "Motion Perception." In *Early Vision and Beyond*, edited by Thomas V. Pappas, 109-112. Cambridge, Massachusetts: The

MIT Press.

Coppin, Ben. 2004. *Artificial Intelligence Illuminated*. Sudbury, Massachusetts: Jones and Bartlett Publishers.

Davis, Ernest. 1990. *Representations of Commonsense Knowledge*. San Mateo, California: Morgan Kaufmann Publishers.

Davis, Ernest. 2018. Collection of Winograd Schemas.

Davis, Ernest, and Gary Marcus. 2015. "Commonsense reasoning and commonsense knowledge in artificial intelligence." *Communications of the ACM*, 58(9), 92-103. <https://doi.org/10.1145/2701413>.

Devlin, Keith. 1997. *Goodbye, Descartes: The End of Logic and the Search for a New Cosmology of the Mind*. New York: John Wiley & Sons.

Dreyfus, Herbert L. 1979. *What Computers Can't Do, Revised Edition: The Limits of Artificial Intelligence*. New York, N.Y.: Harper & Row, Publishers.

Eade, James. 1996. *Chess For Dummies*. Foster City, CA: IDG Books Worldwide.

Fischler, Martin A., and Oscar Firschein. 1987. *Intelligence: The Eye, the Brain, and the Computer*. Reading, Massachusetts: Addison-Wesley Publishing Company.

GBH Archives. 1990. "The Machine That Changed The World: Interview with Marvin Minsky."

Gleick, James. 1987. *Chaos: Making a New Science*. New York, New York: Viking Penguin.

Gosling, James, Bill Joy, Guy Steele, and Gilad Bracha. 1996. *The Java Language Specification, Second Edition*. Boston: Addison-Wesley.

Harris, Laurence R., and Michael R.M. Jenkin. 1997. "Computational and psychophysical mechanisms of visual coding." In *Computational and Psy-*

chophysical Mechanisms of Visual Coding, edited by Michael Jenkin and Laurence Harris, 1-19. New York, NY: Cambridge University Press.

Haugeland, John. 1985. *Artificial Intelligence: The Very Idea*. Cambridge, Massachusetts: The MIT Press.

Hawkins, Jeff. 2004. *On Intelligence*. New York: Times Books.

Hofstadter, Douglas R. 1979. *Gödel, Escher, Bach: an Eternal Golden Braid*. New York: Basic Books.

Hogan, James P. 1997. *Mind Matters: Exploring the World of Artificial Intelligence*. New York: The Ballantine Publishing Group.

Huang, Bin, Siao Tang, Guangyao Shen, Guohao Li, Xin Wang, and Wenwu Zhu. 2020. "Commonsense Learning: An Indispensable Path towards Human-centric Multimedia." HuMA'20, October 12, 2020, Seattle, WA, USA.

Jenkin, Heather L. 1997. "A historical review of the imagery debate." In *Computational and Psychophysical Mechanisms of Visual Coding*, edited by Michael Jenkin and Laurence Harris, 268-295. New York, NY: Cambridge University Press.

Kaku, Michio. 2011. *Physics of the Future: How Science Will Shape Human Destiny and Our Daily Lives By the Year 2100*. New York: Doubleday.

Khinchin, A. Ya. 1964. *Continued fractions*. Chicago: University of Chicago Press.

Kocijan, Vid, Ernest Davis, Thomas Lukasiewicz, Gary Marcus, and Leora Morgenstern. 2022. "The Defeat of the Winograd Schema Challenge." <https://arxiv.org/abs/2201.02387> (accessed September 3, 2022)

Kolln, Martha, and Robert Funk. 2006. *Understanding English Grammar, Seventh Edition*. New York: Pearson Education.

Kosslyn, Stephen M., William L. Thompson, and Giorgio Ganis. 2006. *The Case for Mental Imagery*. New York: Oxford University Press.

Kurzweil, Raymond. 1990. *The Age of Intelligent Machines*. Cambridge, Massachusetts: Massachusetts Institute of Technology.

Kurzweil, Ray. 1999. *The Age of Spiritual Machines: When Computers Exceed Human Intelligence*. New York, New York: Viking Penguin.

Lettvin, J. Y. H. R. Maturana, W. S. McCulloch, and W. H. Pitts. 1959. "What the Frog's Eye Tells the Frog's Brain." Proceedings of the IRE (Volume: 47, Issue: 11, November 1959).

Luger, George F., and William A. Stubblefield. 1998. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Third Edition*. Reading, MA: Addison Wesley Longman.

Lytinen, Steven L. 1992. "Conceptual Dependency and its Descendants." Computers & Mathematics with Applications, Vol. 23, No. 2-5, pp. 51-73. Great Britain: Pergamon Press.

MacLean, Paul D. 1990. *The Triune Brain in Evolution*. New York: Plenum Press.

Maor, Eli. 1994. *e: The Story of a Number*. Princeton, New Jersey: Princeton University Press.

Markman, Arthur B. 1999. *Knowledge Representation*. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Marr, David. 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Cambridge, Massachusetts: The MIT Press.

Melzack, Ronald, and Patrick D. Wall. 1965. "Pain Mechanisms: A New Theory: A gate control system modulates sensory input from the skin before it evokes pain perception and response." Science, Vol. 150, Issue 3699, 19 Nov 1965, pp. 971-979, DOI: 10.1126/science.150.3699.97.

Minsky, Marvin. 1974. A Framework for Representing Knowledge. Memo

No. 306.

Minsky, Marvin. 1986. *The Society of Mind*. New York: Simon and Schuster.

Newell, Allen. 1990. *Unified Theories of Cognition*. Cambridge, Massachusetts: Harvard University Press.

O'Neill, Gerard K. 1981. *2081: A Hopeful View of the Human Future*. New York: Simon and Schuster.

Orban, Guy A. 2008. "Higher Order Visual Processing in Macaque Extrastriate Cortex." *Physiological Reviews*. January 2008, pp. 59-89.

Partridge, Derek. 1991. *A New Guide to Artificial Intelligence*. Norwood, New Jersey: Ablex Publishing Corporation.

Pezzulo, Giovanni. 2007. "Anticipation and Future-Oriented Capabilities in Natural and Artificial Cognition." In *50 Years of Artificial Intelligence: Essays Dedicated to the 50th Anniversary of Artificial Intelligence*, edited by Max Lungarella, Fumiya Iida, Josh Bongard, and Rolf Pfeifer, 257-270. Berlin, Germany: Springer-Verlag.

Reichgelt, Han. 1991. *Knowledge Representation: An AI Perspective*. Norwood, New Jersey: Ablex Publishing Corporation.

Reisberg, Daniel, and Fridrike Heuer. 2005. "Visuospatial Images." In *The Cambridge Handbook of Visuospatial Thinking*, edited by Priti Shah and Akira Miyake, 35-80. New York, NY: Cambridge University Press.

Rieger, Chuck. 1975. "The Commonsense Algorithm as a Basis for Computer Models of Human Memory, Inference, Belief and Contextual Language Comprehension." *Theoretical Issues in Natural Language Processing*.

Robinson, David L. 2009. "Brain function, emotional experience and personality." *The Netherlands Journal of Psychology*. pp. 152-167.

Rorvig, Mordechai. 2021. "Supersized AI". *New Scientist*. October 9, 2021, pp. 37-40.

Russell, Stuart J., and Peter Norvig. 2010. *Artificial Intelligence: A Modern Approach, Third Edition*. Upper Saddle River, New Jersey: Prentice Hall.

Sabbatini, Renato M.E. 1998. "The mind, artificial intelligence and emotions: Interview with Marvin Minsky."

Schank, Roger C. 1972. "Conceptual Dependency: A Theory of Natural Language Understanding." *Cognitive Psychology* 3, 552-631.

Schank, Roger C. 1975. "The Primitive ACTs of Conceptual Dependency." In *Theoretical Issues in Natural Language Processing*.

Schank, Roger C. 1976. "The Role of Memory in Language Processing." In *The Structure of Human Memory*, edited by Charles N. Cofer, 162-189. San Francisco: W. H. Freeman and Company.

Shastri, Lokendra, and Venkat Ajjanagadde. 1990. "From simple associations to systemic reasoning: A connectionist representation of rules, variables and dynamic bindings." *Behavioral and Brain Sciences*, 16(3), January 1990, pp. 417-494.

Simpson, Patrick K. 1990. "Neural Networks for Sonar Signal Processing." In *Handbook of Neural Computing Applications*, edited by Alianna Maren, Craig Harston, and Robert M. Pap, 319-335. San Diego, California: Academic Press.

Sowa, John F. 2000. *Knowledge Representation*. Pacific Grove, CA: Brooks Cole Publishing Co.

Tveter, Donald R. 1998. *The Pattern Recognition Basis of Artificial Intelligence*. Los Alamitos, California: The Institute of Electrical and Electronics Engineers.

Winston, Patrick Henry. 1998. *On To Smalltalk*. Reading, Massachusetts: Addison-Wesley.

Wolfram, Stephen. 2002. *A New Kind of Science*. Champaign, IL: Wolfram

Media.

Woods, W.A. 1975. "What's in a link: Foundations for Semantic Networks."
Bolt Beranek and Newman.

Woods, William A. 1986. "Important Issues in Knowledge Representation."
Proceedings of the IEEE, 74(10), October 1986, pp. 1322-1334.

23 arXiv response

arXiv:submit/5311760 [cs.AI] 22 Dec 2023

Tumbug: A pictorial, universal knowledge representation method

Mark A. Atkins, Ph.D.
aginik@outlook.com

December 22, 2023

Abstract

Since the key to artificial general intelligence (AGI) is commonly believed to be commonsense reasoning (CSR) or, roughly equivalently, discovery of a knowledge representation method (KRM) that is particularly suitable for CSR, the author developed a custom KRM for CSR. This novel KRM called Tumbug was designed to be pictorial in nature because there exists increasing evidence that the human brain uses some pictorial type of KRM, and no well-known prior research in AGI has researched this KRM possibility. Tumbug is somewhat similar to Roger Schank's Conceptual Dependency (CD) theory, but Tumbug is pictorial and uses about 30 components based on fundamental concepts from the sciences and human life, in contrast to CD theory, which is textual and uses about 17 components (= 6 Primitive Conceptual Categories + 11 Primitive Acts) based mainly on human-oriented activities. All the Building Blocks of Tumbug were found to generalize to only five Basic Building Blocks that exactly correspond to the three components {O, A, V} of traditional Object-Attribute-Value representation plus two new components {C, S}, which are Change and System. Collectively this set of five components, called "SCOVA," seems to be a universal foundation for all knowledge representation.

Contents

1 Introduction	6
----------------	---

1

Figure 335: December 22, 2023: Submission to arXiv with date stamp.

arXiv submission submit/5311760



e-prints@arxiv.org

To: You

Dear arXiv user,

We have received your submission to arXiv. Your temporary submission identifier is: submit/5311760.

You may update your submission at: <https://arxiv.org/submit/5311760>

Your article is currently scheduled to be announced at Mon, 25 Dec 2023 01:00:00 GMT. Updates before Fri, 22 Dec 2023 19:00:00 GMT will not delay announcement.

All submissions are subject to arXiv moderation, which may result in a delay. Information about moderation is at: <https://arxiv.org/help/moderation>. General information about submission status is at: https://arxiv.org/help/submit_status.


Your submission status is displayed in your user dashboard: <https://arxiv.org/user/>. There may be a delay of an hour or more after the announcement time when the submission is not visible while it is being processed by our system. When the process is complete the public version of the work will appear.

After the announcement process you will be emailed a link to the paper. That email will include a paper password that you should share with co-authors to allow them to claim ownership.

The title and abstract will appear in the daily announcement mailings as displayed below, except that the submission identifier will be replaced by the official arXiv identifier.

Figure 336: December 21, 2023: arXiv e-mail stating that posting is scheduled for December 25, 2023.

← → ↻ arxiv.org/user/ 🔍 ☆ 📄 📱 👤 ⋮

 Cornell University We gratefully acknowledge support from the Simons Foundation, member institutions, and all contributors. [Donate](#)

arXiv > user Logged in as **GreenIdeas** | [Help](#) | [My Account](#) | [Logout](#)

Your arXiv.org account: GreenIdeas

E-mail: aginik@outlook.com **Affiliation:** Unaffiliated
Name: Mark Atkins **URL:**
Default Category: cs.AI **Country:** United States of America
Groups: physics and cs **Career Status:** Other

[Change User Information](#) | [Change Password](#) | [Change Email](#) | [Disable MathJax \(What is MathJax?\)](#)

Article Submissions 📄 Update ✖ Delete 📄 Unsubmit				
Identifier	Type	Title	Status	Expires
submit/5311757	New	Submission ID: 5311757	incomplete	2024-01-05
submit/5311760	New	Tumbug: A pictorial, universal knowledge representation method	on hold	

[START NEW SUBMISSION](#)

Figure 337: December 26, 2023: arXiv indicates that article is still on hold.