Coding Notes

Jeffrey Young

Feb 23, 2017

Purpose

The Purpose of encodings is to systematically and formally codify real world, in the wild, explanations so that observing larger patterns becomes possible. The end goal of this is:

1. Generate a database of encodings
2. Analyze database to find patterns of explanation
3. 1 and 2 become formative work for exploring DSL possibilities in XOP

Scope

The scope of the database is restricted to explanations of common Computer Science algorithms *from* Universities only. Restricting the scope in this manner provides two benefits:

1. All explanatory objects have a stated, intrinsic goal to communicate the mechanics, application, and implementation of similar things
2. There are numerous examples of different approaches to explain *the same* thing, and numerous examples of *like* approaches to explain different things

Data Collection

All Data is coded by human individuals with reference to this document. The location index represents the location of an encoding *relative* to the document. So a location of 1 means the literal first page of the document (typically a header slide or introduction).

First Typology Attempt

Some Terminology

In our terminology an explanation is called an "explanation artifact", our working model of the process of explaining is a non-linear sequence of "steps", where each "step" denotes some progress in the understanding of the explanation artifact on the part of the information receiver. More formally:

Explanatory Artifact The whole explanation, including all the step taken in the explanation, the language used in the explanation etc.

Step The steps that are taken, in an explanatory artifact, that guide the reader from non-understanding to understanding.

Syntax and Grammer

The syntax of an encoding is given by _ - _ - _ - _ $\subseteq$ Location $\times$ Level $\times$ Role $\times$ Notation where

l $\in \mathbb{N}$

g $\in$ Level ::= Problem | Algorithm | Implementation

r $\in$ Role ::= Background | Definition | Constraint | Example | Application | Variant | Analogy | Idea | Proof | Performance | Properties

n $\in$ Notation ::= English | Math | Diagram | List | Table | Sequence | Pseudocode | Code | Animation | Picture | Plot | Empty | n/n

Semantics

Location

a location l, specifies the location that the encoding is referring to, this could be a slide, a number line etc.

Level

a Level g, Specifies the level of abstraction for a given step, a level can by one of:

Problem $\triangleq$ the purpose of a given step is to elucidate the motivating problem of the algorithm, i.e the problem that the algorithm will solve

Algorithm $\triangleq$ the purpose of a given step is to explain the algorithm at hand

Implementation $\triangleq$ the purpose of a given step is to explain the implementation details of the algorithm, e.g. What data structures to use, What the form of the code that implements the algorithm should be

Role

a Role r, specifies how the Level is trying to be reached, denotes the answer to question such as "What is the step trying to convey?":

In general the meaning of each role is:

Background $\triangleq$ A given level is reached by a step that describes the history, creators, genealogy of the Algorithm

Definition $\triangleq$ A given level is reached by a step that explicitly provides a formal definition.

Constraint $\triangleq$ A given level is reached by a step that explicitly presents a limit or condition in which the level would cease to be valid, e.g. Dijkstra's only works on non-negative weighted graphs

Example $\triangleq$ A given level is reached by a step that provides an Example

Application $\triangleq$ A given level is reached by a step that explains what the algorithm is useful for

Variant $\triangleq$ A given level is reached by a step that describes things that are similar but slightly different than the algorithm. For example, describing Prim's algorithm and it's similarities to Dijkstra's or describing the similarities between a dog and a wolf

Analogy $\triangleq$ A given level is reached by a step that provides an Analogy to explain the algorithm at hand. For example a visual analogy for Dijkstra's could be: If you have a physical model of a graph, and you pick it up by one vertex, then the vertex with the shortest path to the "source" vertex will be the one farthest from the ground.

Idea $\triangleq$ A given level is reached by a step that adds an abstract idea to the explanation as a way to progress. For example, the statement "Well we have this, *what if we did* this?"

Proof $\triangleq$ A given level is reached by a formal proof

Performance $\triangleq$ A given level is reached by a step that explicitly describes the computational complexity of the level

Properties $\triangleq$ A given level is reached by a step that explicitly describes the properties of that level. For Example, AVL Trees are both *balanced* and *ordered*.

Consider the following matrix of Level Role combinations of Dijkstra's algorithm. Not all of the cells will be orthogonal to each other. In this case we would have:

Problem: How to traverse the shortest path in a non-negative weighted graph

Algorithm: Dijkstra's Algorithm

Implementation: You should use a Priority Queue that has efficient lookup, mutate operations.

- $\perp$ used to denote cells which may be nonsensical

Role	Problem	Algorithm	Implementation
Definition	Mathematical definition of Problem	Mathematical Definition of Algorithm	$\perp$
Example	Display of a non-negative weighted graph	Showing the algorithms execution on the map	Showing requisite data structures etc.
Application	Real World Example of the problem	$\perp$	Triage System in a Hospital
Background	History of the Problem	History, Author, etc.	History of Priority Queues
Variant	Perhaps a teleporter exists, now what is shortest path	Description of Bellman-Ford	Description of slightly different Priority Queues
Analogy	$\perp$	Exposition of Prim's algorithm	$\perp$
Performance	$\perp$	Complexity	Complexity of requisite data structs
Idea	$\perp$	$\perp$	$\perp$
Constraint	Depiction of the Constraints of the Problem	Depiction of domain where Algorithm lacks validity	Requirements of internal Data Structs
Proof	$\perp$	Explicit Proof of Algorithm correctness	Explicit Proof of some requisite part of the algorithm

Notation

a Notation n, specifies the form of the role, and can be one of:

English $\triangleq$ Human language to give explanations/statements.

Diagram $\triangleq$ Diagram in the manner of data structures, such as graph, list.

List $\triangleq$ List of similar items

OrderedList $\triangleq$ Step by step items

Math $\triangleq$ Formulas/math style symbols.

Pseudocode $\triangleq$ Algorithm presented as pseudocode

Code $\triangleq$ executable code to show the algorithm explicitly

Table $\triangleq$ Explanatory information displayed in a table

Animation $\triangleq$ a gif or animation of any type is used.

Picture $\triangleq$ A photo/screenshot or picture is used.

Sequence $\triangleq$ A conjunction of steps meant to show progress in a serial manner

Plot $\triangleq$ A mathematically generated plot that adheres to some coordinate system

for example a definition might be described in English, followed by the same definition described by geometry. Notations can be combined for a single location like so:

$$\frac{n \in \text{Notation} \quad m \in \text{Notation}}{n/m \in \text{Notation}} \tag{1}$$

Typology with following Bellack et al.

Disclaimer

Make sure you are comfortable with the terminology of Bellack et al's typology. I will be referring to it throughout this document and assume the reader is familiar. If you don't know what I mean by Substantive-logical Meanings then go read the Theory Primer document.

Strategy

Merging our typology with Bellack et al's is tricky but doable. In general we will be adding some categories to the "Types of Pedagogical Moves", adding one category to "Substantive-Logical Meanings" and changing the "Speaker" category to a "slide number or page number". Lastly, there is an important design decision that I am making here in which *for each algorithm, we will now be forced to define substantive meanings.* Furthermore, I am going to completely remove the last 3 indices in Bellack's typology. These mostly deal with in class instructional phrases which are completely useless for our purposes.

Syntax, Grammar, Terms

Terms

We will follow with Bellack as close as possible as long as it suits our needs, so a pedagogical move has the form:

```
Move = 1. Line Number or Slide Number
       / 2. Type of Move
       / 3. Substantive Meaning
       / 4. Substantive-Logical Meanings
       / 5. Number of Lines in (3) or (4)
```

Semantics of Moves

1. **Line Number or Slide Number** $\triangleq$ Indicates the source of the statement:
2. **Type of Pedagogical Move** $\triangleq$ reference to function of move, there are 2 types each with sub-types:
   (a) Initiatory Moves
      i. *Structuring* (STR) $\triangleq$ sets context for subsequent behavior by launching or halting-excluding interaction.
      ii. *Soliciting* (SOL) $\triangleq$ directly elicits a verbal, physical, or mental response; coded in terms of response expected.

   (a) Detailing Moves
      i. *Responding* (RES) $\triangleq$ fulfills expectation of solicitation; bears reciprocal relation only to solicitation.
      ii. *Expositing* (EXP) $\triangleq$ a move that explicitly provides or introduces further or new information
   (b) *Not Codable* (NOC) $\triangleq$ serves as the $\perp$ in their coding scheme.
3. **Substantive Meaning** $\triangleq$ reference to a subject matter topic
4. **Substantive-Logical Meaning** $\triangleq$ reference to cognitive process involved in dealing with subject matter under study. 3 Main Types each with subtypes:
   (a) Analytic Process $\triangleq$ Use of language or established rules of logic
      i. *Definining-Denotative* (DED) $\triangleq$ object referent of term
      ii. *Defining-Connotative* (DEC) $\triangleq$ defining characteristics of class or term

iii. *Defining-Definitive* (DEF) $\triangleq$ to give the defining characteristic of a *class and* to give a specific example of an item with that class

iv. *Interpreting* (INT) $\triangleq$ verbal equivalent of a statement, slogan, aphorism, or proverb

v. *Defining-Operational* (DEO) $\triangleq$ to give a definition of some thing as a series of operations or steps affecting a state or machine. This is typically used to describe the explicit presentation of programming code or pseudo-code.

(b) Empirical Process $\triangleq$ sense experience as criterion of truth

i. *Fact-Stating* (FAC) $\triangleq$ what is, was, or will be without explanation or evaluation.

ii. *Explaining* (XPL) $\triangleq$ relation between objects, events, principles, conditional inference, cause-effect, explicit comparison-contrast, statement of principles, theories or laws

(c) Evaluative Process $\triangleq$ set of criteria or value system as basis for verification

i. *Opining* (OPN) $\triangleq$ personal values for statement of policy, judgment or evaluation of event, idea, state of affairs, direct and indirect evaluation included

ii. *Justifying* (JUS) $\triangleq$ reasons or argument for or against opinion or judgment

(d) Visual Process $\triangleq$ a visual representation is provided and discussed

(e) *Logical Process Not Clear* (NCL) $\triangleq$ this serves as $\perp$ for Substantive-Logical Meanings

5. **Number of Lines in 3 and 4 above**

Syntax

We will follow Bellack's syntax as well:

- The / Operator: The moves constituents are syntactically conjoined, *in order*, into strings with the "/" operator like so:

$$\frac{n \in \text{Move Constituents} \quad m \in \text{Move Constituents}}{n/m \in \text{Partial Move}} \tag{2}$$

- An Example coded pedagogical move is:

```
2/STR/MOT/IMX/2
```

The interpretation is as follows:

```
 2 / STR / MOT / IMX /  2
(1)/ (2) / (3) / (4) / (5)
```

This translates to: On slide 2 (1), the presentation makes a *structuring* (2) move in which it *explains* (4) the *motivation* (3) for something for *two* (5) slides

Here is Bellack et al's example for reference:

```
T/STR/IMX/XPL/4/PRC/FAC/2
```

The interpretation is as follows:

```
 T / STR / IMX / XPL /  4  / PRC / FAC / 2
(1)/ (2) / (3) / (4) / (5) / (6) / (7) /(8)
```

This translates to: A *teacher* (1) makes a *structuring* (2) move in which they *explain* (4) something about *imports and exports* (3) for *four* (5) lines of transcript and also states *facts* (7) about class *procedures* (6) for *two* (8) lines of the transcript.

Here is a useful way to think of this (if you haven't read the Theory primer and skipped my warning!):

```
Place of Move / Turn in Language Game / Subject of the Move / How the Move is talking about it
      (1)     /        (2)           /            (3)       /            (4)
```

Differences

There are only a few slight differences between this typology and Bellacks:

1. I've added Types of Pedagogical Moves that are useful for powerpoint slides or lecture notes and removed the reacting type because one cannot react in the same sense to lecture notes or powerpoint slides

2. I've removed the last three categories, see strategy for why

3. I've added *Visual Moves* and *Visual Processes* to the type of moves and the substantive-logical meanings respectively. More discussion on this in number (2) in Caveats and Design Decisions below

4. I've altered the Speaker category to be the location of the move in the document.

Caveats and Design Decisions

For the most part Bellack's system is useful to our needs. There are some important aspects to consider though:

1. We have risen a level of abstraction: In our original Typology we could count the number of proofs or the number of examples. In this revised typology we *can only* state the types and frequencies of pedagogical moves. This means that if you want to know how many pictures there are in a document then this system will fail you. Rather we would be able to say that the document 1) uses visual aids and 2) has a teaching cycle like STR EXP VIS in that the document cycles a Structuring move, then an Expository move, and then has a Visual move to wrap up.

2. Where should Visual moves information be? In the revised typology I've added Visual moves to the "Types of Pedagogical Moves" I did this because this category is based on Wittgenstein's concept of a language game. So in Bellack's typology it made sense that they would have a Soliciting move, a Responding move, and a Reacting move because they were analyzing lectures in the classroom. In our typology it makes no sense because we are analyzing a different type of communication viz. power points and lecture notes. So we need to add a few. But that leads to the curious case of Visual Moves. There are both slides that are of a visual type and slides that discuss something using visual notations e.g. pictures or cartoons, so it seems that Visual information is *both a Type of Move, and a Substantive-Logical Meaning*. Recall that *Substantive-Logical Meanings* is the category that denotes the cognitive process involved in dealing with the subject being discussed so we are saying that for power points or maybe lecture notes there are cognitive visual processes occurring.

3. The last major design decision I made here regards the 3 category, namely *Substantive Meaning*. In Bellack's typology they use this category to be specific about what exactly is being discussed in relation to the overall topic. So for them, the overall topic could be international trade, and then the specifics would be trade tariffs or trade groups. For us the overall topic would be "Dijkstra's Algorithm" or "AVL Trees" and then the specifics could be "Motivation for the algorithm", "Applications of the algorithm", "implementation details of the algorithm". Or we could be even more specific and say the specifics are "Priority Queues" for Dijkstra's and "Tree Rotation" or "Tree Balancing" for AVL Trees. This sort of system is powerful, and more adaptable than our previous notation in that it *does not* seek to provide global, transcendent categories for *every* algorithm. Rather it allows us to *define* the specific attributes for each algorithm and add that information to the typology. This may be or may not be desirable and we should have a lengthy discussion about it at some point because I'm not sure exactly what the implications are or are not.

Substantive Codings Per Algorithm

AVL Trees

Substantive meanings for AVL Trees can be one of:

1. *Motivation* (MOT) $\triangleq$ refers to discussions of the motivations for the thing being discussed
2. *Problems* (PRB) $\triangleq$ refers to explicit problems with the thing being discussed
3. *Tree* (TRE) $\triangleq$ refers to general points about an AVL tree. When a move's substantive meaning is unclear or seems to fit many meanings this term is used.
4. *Traversal* (TRV) $\triangleq$ refers to discussion of how to traverse a tree
5. *Manipulation* (MAN) $\triangleq$ refers to discussion of how to insert, delete, rotate or in general manipulate a tree.
6. *Implementation* (IMP) $\triangleq$ refers to discussion of how to actually implement that which is being discussed. This can also refer to things that are necessary for the implementation of the thing being discussed.

Dijkstra's Algorithm

Substantive meanings for Dijkstra's algorithm can be one of:

1. *Motivation* (MOT) $\triangleq$ refers to discussions of the motivations for the thing being discussed
2. *Problems* (PRB) $\triangleq$ refers to explicit problems with the thing being discussed
3. *Complexity* (COM) $\triangleq$ refers to explicit discussion of the computational complexity of the thing being discussed
4. *Application* (APP) $\triangleq$ refers to discussions of the application of the thing being discussed, concrete or abstract.
5. *Algorithm* (ALG) $\triangleq$ refers to general statements about the algorithm being discussed. When the substantive meaning is unclear or traverses many substantive meanings this coding is used.
6. *Background* (BKG) $\triangleq$ refers to discussion of the history, background, people who created, were involved with, or are notable, for the thing being discussed.