ATLAS.ti Report

XOP\_Encoding\_Proj

Codes grouped by Code groups

Report created by dyg on Aug 20, 2017

# icon Actions

20 Codes:

## ● Abstraction

**Created:** 7/24/17 by dyg, **Modified:** 8/8/17 by dyg

Comment:

This code denotes that the document is abstracting or generalizing that which the context is set to. Ex: “Now we can generalize this problem to the shortest-path problem"

## ● Assumption

**Created:** 7/25/17 by dyg, **Modified:** 8/8/17 by dyg

Comment:

This code denotes that the text is giving the reader or telling the reader an assumption about the context

## ● Base Case

**Created:** 7/25/17 by dyg, **Modified:** 8/8/17 by dyg

Comment:

This code denotes the text is giving an explicit base case in an inductive procedure in relation to the context

## ● Cases

**Created:** 7/10/17 by dyg, **Modified:** 8/8/17 by dyg

Comment:

This code denotes that the text is breaking down the context into chunks of information, or cases. Often used after a “Base Case” code.

## ● Comment

**Created:** 7/31/17 by dyg, **Modified:** 8/8/17 by dyg

Comment:

This code is a dummy code whose use is just to provide a binding for an “Aside”, “Caveat”, or “Meta” modifier

## ● Conclusion

**Created:** 8/3/17 by dyg, **Modified:** 8/3/17 by dyg

Comment:

This code denotes that the text is making a conclusion about the context

## ● Contrast

**Created:** 8/4/17 by dyg, **Modified:** 8/4/17 by dyg

Comment:

This code denotes that the text is contrasting the context with something else. This code may no longer be viable because it expresses in a worse way what the combination of “Explanandoid” “Relation” modifiers express.

## ● Definition

**Created:** 7/10/17 by dyg, **Modified:** 8/20/17 by dyg

Comment:

This code denotes that the text is defining some term about the context.

## ● Derivation

**Created:** 7/27/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the text is making a derivation about something in relation to the context

## ● Description

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code is the most general Action code. It denotes that the text is describing the context in some manner.

## ● Example

**Created:** 7/10/17 by dyg, **Modified:** 8/7/17 by dyg

Comment:

This code denotes that the text is giving, or providing an example in relation to the context

## ● Implication

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the text is giving an implication about the context. This could be anything that fits the logical connective if..then..else..

## ● In vivo term introduction

**Created:** 7/10/17 by dyg, **Modified:** 8/7/17 by dyg

Comment:

This code denotes that the text is defining a term that practitioners of algorithms would be familiar with. In Vivo terms are the “language of the trade” so to speak.

## ● Legend

**Created:** 7/17/17 by dyg, **Modified:** 8/7/17 by dyg

Comment:

This code denotes that the text is giving a legend to understand something. For example if the text content is expressed in a graphic, there will commonly be a paragraph that describes, in detail, all things of interest in the graph. Here is a textual example: “The blue nodes are the recursive calls of merge sort, the yellow are the calls to the merge suboperation"

## ● Observation

**Created:** 8/7/17 by dyg, **Modified:** 8/7/17 by dyg

Comment:

This code denotes the text making a general observation about the context. Example: “At node 13 the tree has height 0” or “We can see that the last loop will never finish and will continue processing in infinitum"

## ● Outline

**Created:** 7/31/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the text is giving a bulleted list of the content the document will go through

## ● Proof

**Created:** 8/4/17 by dyg, **Modified:** 8/4/17 by dyg

Comment:

This code denotes the text giving a mathematic or logical proof about the context

## ● Proposal

**Created:** 7/24/17 by dyg, **Modified:** 8/4/17 by dyg

Comment:

This code is similar to “Assumption” however this code denotes the text suggesting a path forward. For example: We can solve this problem with BFS algorithm

## ● Solicitation

**Created:** 7/25/17 by dyg, **Modified:** 8/8/17 by dyg

Comment:

This code denotes that the text is explicitly asking something of the reader. These statements are typically formed as questions.  Ex: “If you find the recursion confusing, make sure you go back and understand it before moving forward"

## ● Summary

**Created:** 7/25/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes a concluding block of text that summarizes the previous contexts

# icon Concepts

10 Codes:

## ● ADT

**Created:** 8/3/17 by dyg, **Modified:** 8/8/17 by dyg

## ● divide and conquer

**Created:** 7/25/17 by dyg, **Modified:** 8/8/17 by dyg

## ● FIFO

**Created:** 7/24/17 by dyg, **Modified:** 8/8/17 by dyg

## ● Greedy

**Created:** 7/24/17 by dyg, **Modified:** 8/8/17 by dyg

## ● Insertion

**Created:** 7/10/17 by dyg, **Modified:** 8/8/17 by dyg

## ● Invariant

**Created:** 7/10/17 by dyg, **Modified:** 8/8/17 by dyg

Comment:

Another in vivo term that may not be transmissible

## ● LIFO

**Created:** 7/24/17 by dyg, **Modified:** 8/8/17 by dyg

## ● rebalance

**Created:** 7/10/17 by dyg, **Modified:** 8/8/17 by dyg

Comment:

rebalance is an in vivo term for Trees in general. This means it will not be able to be applied to other algorithms, perhaps this code should be subsumed into SubOp code.

## ● Recursive

**Created:** 7/10/17 by dyg, **Modified:** 8/8/17 by dyg

## ● Symmetry

**Created:** 8/7/17 by dyg, **Modified:** 8/8/17 by dyg

# icon Content Expression

6 Codes:

## ● Cartoon

**Created:** 7/10/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

The content is represented in a drawn or animated graphic

## ● Code

**Created:** 7/10/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

The content is represented as a block of code from some programming language

## ● Mathematic

**Created:** 7/10/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

The content is represented using Mathematic formulae, variables, or equations

## ● PseudoCode

**Created:** 8/17/17 by dyg, **Modified:** 8/20/17 by dyg

Comment:

The content is represented in pseudocode. That is a programming-like language that is not an actual programming language.

## ● Sequence

**Created:** 8/17/17 by dyg, **Modified:** 8/17/17 by dyg

Comment:

This code denotes that the content is being expressed in a ordered, bullted or punctuated way. An Example would be describing an algorithm, in text, without giving explicit code or pseudocode e.g. do x, then d y, then if y do z. Often times these blocks of text belong to bulleted lists.

## ● Table

**Created:** 8/4/17 by dyg, **Modified:** 8/4/17 by dyg

Comment:

The content is explicitly displayed in a Table

# icon Document Level Codes

3 Codes:

## ● Black Box

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

## ● Fill the mold

**Created:** 7/25/17 by dyg, **Modified:** 8/1/17 by dyg

## ● self-teaching

**Created:** 7/25/17 by dyg, **Modified:** 8/1/17 by dyg

# icon Elements of Explanatory Structure

20 Codes:

## ● Advantages

**Created:** 8/7/17 by dyg, **Modified:** 8/7/17 by dyg

Comment:

This code denotes the pros, the upside or the advantages to that which the context is set to.

## ● Algorithm

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the text is discussing an algorithm in general. This could be the same as the thesis topic, it could be a related algorithm (and have a “Related” modifier) etc.

## ● Application

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This scoping tag denotes that the text is now discussing the Use cases, or applications of the current context. Defaults to the Thesis Topic.

## ● Class

**Created:** 8/3/17 by dyg, **Modified:** 8/7/17 by dyg

Comment:

This code denotes the explicit discussion of a group, set or class of some thing.

## ● Complexity

**Created:** 7/10/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the text is discussing the computational complexity of whatever the context is set to.

## ● Condition

**Created:** 7/10/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the text is discussing a condition that whatever the context was set to, has, that must be satisfied.

## ● Constituent

**Created:** 7/31/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code can be seen as a more general form of “Operation”. This code denotes that the text is discussing some constituent part of whatever the context was set to. This is typically used to denote data structures required for an algorithm.

## ● Data Structure

**Created:** 8/3/17 by dyg, **Modified:** 8/10/17 by dyg

Comment:

This code denotes that the text is discussing a Data Structure, this is an analog to the Algorithm code.

## ● Design

**Created:** 7/25/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the text is talking about the design, or design considerations of that which the context is set to.

## ● Disadvantages

**Created:** 8/7/17 by dyg, **Modified:** 8/7/17 by dyg

Comment:

This code denotes that the text is discussing the downsides, the cons or shortfalls of whatever the context was set to.

## ● Goal

**Created:** 7/25/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the text is discussing the goal, the end game, that which is the desired outcome, of whatever the context was set to.

## ● History

**Created:** 7/10/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the text is discussing the history of whatever the context was set to.

## ● Implementation

**Created:** 7/25/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the text is discussing implementation details of that which the context is set to. These are things such as “typically algorithm x is not implemented with a fibonnacci heap because of y and z”. This code does not refer to things that are required for the implementation of an algorithm, such as a priority queue for Dijkstra’s. Such things would either be Constituent or Operation tags.

## ● Motivation

**Created:** 7/10/17 by dyg, **Modified:** 8/7/17 by dyg

Comment:

This code denotes that the text is discussing the motivation for whatever the current context is set to.

## ● Operation

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotest that the text is discussing an Operation that is a requisite and central part of whatever the current context is set to.

## ● Problem

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the text is discussing a problem that is solved by whatever the context is referring to. This does not say what kind of problem it is, such things are handled through modifiers. This code can also refer to the introduction of a sub problem, that will be solved, again done this is done by modifiers

## ● Property

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the text is now discussing some property of whatever is the current context.

## ● Solution

**Created:** 7/10/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the text is now discussing a solution to whatever is in the current context. This is always used to “close” a previous “Problem” context.

## ● State

**Created:** 8/7/17 by dyg, **Modified:** 8/7/17 by dyg

Comment:

This scoping operator denotes that the text is now discussing something related to the state or the state of whatever is in the current context. Defaults to Thesis Topic

## ● Structuring

**Created:** 7/10/17 by dyg, **Modified:** 8/1/17 by dyg

# icon In Vivo Terms

39 Codes:

## ● balanced

**Created:** 8/3/17 by dyg, **Modified:** 8/3/17 by dyg

## ● Binary Search Trees

**Created:** 8/3/17 by dyg, **Modified:** 8/8/17 by dyg

## ● breadth first search

**Created:** 7/24/17 by dyg, **Modified:** 8/8/17 by dyg

## ● combine

**Created:** 7/25/17 by dyg, **Modified:** 8/8/17 by dyg

## ● comparison sorts

**Created:** 8/2/17 by dyg, **Modified:** 8/8/17 by dyg

## ● completed vertices

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

## ● conquer

**Created:** 7/25/17 by dyg, **Modified:** 8/1/17 by dyg

## ● constraints

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

## ● depth first search

**Created:** 7/24/17 by dyg, **Modified:** 8/8/17 by dyg

## ● directed graph

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

## ● distance matrix

**Created:** 7/31/17 by dyg, **Modified:** 8/1/17 by dyg

## ● divide

**Created:** 7/25/17 by dyg, **Modified:** 8/1/17 by dyg

## ● divide and conquer

**Created:** 7/25/17 by dyg, **Modified:** 8/8/17 by dyg

## ● dynamic programming

**Created:** 8/1/17 by dyg, **Modified:** 8/1/17 by dyg

## ● dynamic programming functional equation

**Created:** 8/1/17 by dyg, **Modified:** 8/1/17 by dyg

## ● external sorting

**Created:** 8/20/17 by dyg, **Modified:** 8/20/17 by dyg

## ● frontier

**Created:** 7/24/17 by dyg, **Modified:** 8/8/17 by dyg

## ● Greedy

**Created:** 7/24/17 by dyg, **Modified:** 8/8/17 by dyg

## ● height

**Created:** 8/16/17 by dyg, **Modified:** 8/16/17 by dyg

## ● Insertion

**Created:** 7/10/17 by dyg, **Modified:** 8/8/17 by dyg

## ● internal paths

**Created:** 7/24/17 by dyg, **Modified:** 8/7/17 by dyg

## ● Invariant

**Created:** 7/10/17 by dyg, **Modified:** 8/8/17 by dyg

Comment:

Another in vivo term that may not be transmissible

## ● length

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

## ● merging

**Created:** 8/16/17 by dyg, **Modified:** 8/16/17 by dyg

## ● path

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

## ● principle of optimatlity

**Created:** 8/1/17 by dyg, **Modified:** 8/1/17 by dyg

## ● priority queue

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

## ● rebalance

**Created:** 7/10/17 by dyg, **Modified:** 8/8/17 by dyg

Comment:

rebalance is an in vivo term for Trees in general. This means it will not be able to be applied to other algorithms, perhaps this code should be subsumed into SubOp code.

## ● recurrence

**Created:** 7/25/17 by dyg, **Modified:** 8/8/17 by dyg

## ● Recursive

**Created:** 7/10/17 by dyg, **Modified:** 8/8/17 by dyg

## ● running time

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

## ● Search

**Created:** 7/10/17 by dyg, **Modified:** 8/7/17 by dyg

## ● shortest-path

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

## ● simple paths

**Created:** 7/31/17 by dyg, **Modified:** 8/1/17 by dyg

## ● stable sorting

**Created:** 7/25/17 by dyg, **Modified:** 8/1/17 by dyg

## ● stopping rule

**Created:** 8/1/17 by dyg, **Modified:** 8/1/17 by dyg

## ● undirected graph

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

## ● unvisited

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

## ● weighted

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

# icon Modifiers

13 Codes:

## ● ->

**Created:** 8/4/17 by dyg, **Modified:** 8/4/17 by dyg

Comment:

Subscoping operator. This means that scopes are nested. For example, if you have a code at position x that reads “Constituent, Description” and then a paragraph/line/s later, at position y, you have “->, Complexity” then you are no denoting teh complexity of the consituent being discussed. This is opposed to having just “Complexity” at position y which would denote the complexity of whatever the previous structuring move was.

## ● <-

**Created:** 8/7/17 by dyg, **Modified:** 8/7/17 by dyg

Comment:

Subscoping lifting operator. This operator ends subscoping context, returning the scope to whatever the prior scope was. For example if you had “Operation” then a few lines down you have “->” “Complexity” then the scope would be talking about Operation Complexity. If after a few more lines you had “<-“ “Observation” Then you are now making a general observation about the Operation, not the Operation complexity. This is in contrast to just “Observation” which, in this case, would then be making an Observation of the Operation Complexity

## ● <<-

**Created:** 8/19/17 by dyg, **Modified:** 8/19/17 by dyg

Comment:

Ultimate scope closing operator. This modifier can be orphaned (no input) or paired with a subscoping tag. In either case this operator returns to root node (scope) of the document. If a subscoping operator is used orphaned (with no accompanying tags) then that is synonymous with “<<-“ and that tag. For ex. We have “Data Structure” and then "->” “Operation” then later, “->” “Motivation” to denote we are discussing the motivation for an operation of a data structure. If we then have just “<<-“, then we return to the Data Structure Scope. Or, if we have “History” then we return to the data structure scope and sub scope into history to discuss the history of the data structure. The equivalent code is “<<-“, then “->” “History”.

## ● Aside

**Created:** 7/25/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This is a special modifier code. It denotes that the input tags describe text that is not directly related to the scope the input tags refer to. Cases where this modifier is used without any accompanying tag are considered equivalent to the tag set “Aside” “Comment”. That is to say, if Aside occurs with no input, then the Comment tag is considered as default input

## ● caveat

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This modifier takes any number of tags of any type. It denotes that the purpose of the text at hand is to further clarify a point, provide extra detail, or point out a consequence of some aspect of that which is being discussed. Ex: In general, this process will require many rotations for an insert because...

## ● Erroneous

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This code denotes that the purpose of the text, upon which the input tags represent, is to be incorrect, however that may be defined.

## ● Explanandoid

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This modifier tags takes a scoping tag as input. If denotes that the scoping tag, typically “problem” is used to illustrate something that is close to the thesis topic, but not quite the same. This tag is often used with the “Related” modifier to denote the discussion of a related problem that is similar to the Thesis Topic but is not the thesis topic. Ex: Considering, discussing and explaining the shortest-path problem in relation to the motivating problem for dijkstra’s algorithm.

## ● Focus

**Created:** 7/24/17 by dyg, **Modified:** 8/20/17 by dyg

Comment:

This modifier denotes the purpose of the text that it is attached to is to further refine some aspect of that which is being discussed. To “focus in” to some aspect of that which is being discussed. This code is most likely no longer useful given that we now have explicit subscoping operators.

## ● Meta

**Created:** 7/31/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This modifier takes any number of tags of any type. It denotes text that is not directly related to explaining the scope at hand. Ex: This algorithm is hard for many people to understand because of its simplicity.

## ● Pedagogical

**Created:** 8/1/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

A modifier, whose purpose is to denote that the specific purpose of a statement is pedagogical in nature. This modifier can be applied to any number of tags of any type.

## ● Related

**Created:** 7/24/17 by dyg, **Modified:** 8/1/17 by dyg

Comment:

This modifier takes any number of arguments of any type. It denotes that the tags it is attached to are substantially related to the thesis topic in some manner. Ex: The more general shortest-path problem, is related, to the motivating problem for Dijkstra’s algorithm.

## ● Review

**Created:** 7/10/17 by dyg, **Modified:** 8/3/17 by dyg

Comment:

This modifier takes any number of tags of any type. It denotes that the purpose of those tags is to provide a pedagogical review of the material to the reader.

## ● |

**Created:** 8/10/17 by dyg, **Modified:** 8/10/17 by dyg

Comment:

This code is a sister to the subscoping operators <- and ->. It denotes a sibling scope. For example, if one has “History” followed by “->” “Operation” then one could use “|” “Design” to set the context to the “History” “-> Design”. Basically “|” is just syntactic sugar for a “<-“ “->” “Design” tag.

# No code group

9 Codes:

## ○ Big Picture

**Created:** 8/3/17 by dyg, **Modified:** 8/3/17 by dyg

## ○ double rotation

**Created:** 8/10/17 by dyg, **Modified:** 8/10/17 by dyg

## ○ Industry

**Created:** 7/31/17 by dyg, **Modified:** 7/31/17 by dyg

## ○ relaxation

**Created:** 8/17/17 by dyg, **Modified:** 8/17/17 by dyg

## ○ rotations

**Created:** 8/15/17 by dyg, **Modified:** 8/15/17 by dyg

## ○ single rotation

**Created:** 8/10/17 by dyg, **Modified:** 8/10/17 by dyg

## ● Thesis Topic

**Created:** 7/24/17 by dyg, **Modified:** 8/8/17 by dyg

## ○ | Application

**Created:** 8/16/17 by dyg, **Modified:** 8/16/17 by dyg

## ○ | Operation

**Created:** 8/16/17 by dyg, **Modified:** 8/16/17 by dyg