Programmed Conceptual Deconstruction Instructions

Phase 1: Initialization

1. Define initial concept C_i.j.k.
   Example: “Ancient Greece”.
   - i: Depth level of the concept.
   - j: Numbering of the concept within level i.
   - k: Value j of the concept from which it comes. If it is the initial concept, then k = 0.
2. Establish a maximum depth level N:
   Determines up to which level the initial concept will be subdivided.
   Example: N = 3
3. Define the maximum number of subconcepts per concept M:
   Limits the number of subconcepts generated in each deconstruction.
   Example: M = 5
4. Initialize the necessary lists and data structures
   List of pending concepts: Cp.
   List of analyzed concepts: Ca.
   Waitlist: W.
   Non-return list: Nr.
   List of Conceptual Quarks: QC.

Initialization example: Cp = [Ancient Greece_0.1.0], Ca = [ ], W = [ ], Nr = [ ], QC = [ ].

Phase 2: Processing

1. Deconstruction iteration:
   For each concept C_i.j.k in the list of pending concepts:
   1. Decompose C_i.j.k into subconcepts [C_i.1.k, C_i.2.k, ..., C_i.j.k] by employing the definition of C_i.j.k and extracting relevant terms and of a lower level of complexity, without j > M.
      - Where 'k' is the value 'j' of the concept from which it comes and 'j' is the numbering of each concept at this level i of depth.
      - Insert all new concepts in the waitlist.
        Example: W = [History_1.1.1, Philosophy_1.2.1, Politics_1.3.1, Art_1.4.1, ..., Mythology_1.j.1]
      - If it cannot be deconstructed into relevant subconcepts, insert C_i.j.k in the no return list, in the QC list and in the Ca list.

   2. Add C_ijk to the list of analyzed concepts.
      Example: Cp = [ ], Ca = [Ancient Greece_0.1.0], W = [History_1.1.1, Philosophy_1.2.1, Politics_1.3.1, Art_1.4.1, ..., Mythology_1.j.1], Nr = [ ], QC = [ ].

   3. Check loops:
      - If inside W a concept C_i.j.k represents the same concept C as one of those inside Ca:
        Example: W = [Ancient Greece_4.1.1.1], Ca = [Ancient Greece_0.1.0, History_1.1.1, Philosophy_2.1.1, Socrates_3.1.1].
        Check whether the two concepts are linked by employing the Conceptual Closure algorithm:
        1. Create a temporary loop list: Bt = [ ] 2.
        2. Select the repeating concept C_i.j.k within the list W and add it to Bt.
           Example: Bt = [Ancient Greece_4.1.1] 3.
        3. If the added concept is C_i.j.k, add the concept to the list with: i = i-1; j = k;
           Example: Bt = {Ancient Greece_4.1.1, Socrates_3.1.1}
        4. Repeat until i = 0 or until C_i.j.k = C_i2,j2,k2,:

- If you reach depth level 0 and have not found a C_i2,j2,k2 with the same value C, there is no concept loop.
- If you reach a concept C_i2,j2,k2with the same value C, you have found a conceptual loop.

Example of a conceptual loop: Bt = [{Ancient Greece_0.1.0, History_1.1.1, Philosophy_2.1.1, Socrates_3.1.1, Ancient Greece_4.1.1}]]

- If a conceptual loop exists, insert each element of that loop into a list within the QC and Bt list, and also add to the no return list and Ca the element within Bt with 'i' of greater value, also remove that element from the waitlist.

  Example: Nr = [Ancient Greece_4.1.1], QC = [{Ancient Greece_0.1.0, History_1.1.1, Philosophy_2.1.1, Socrates_3.1.1, Ancient Greece_4.1.1}]

4. Check depth:
   - If the value of 'i' is less than N:

     Move concepts within the waitlist that are not in the no return list to the pending concepts list.

     Example: Cp = [History_1.1.1, Philosophy_1.2.1, Politics_1.3.1, Art_1.4.1, ..., Mythology_1.j.1], Ca = [Ancient Greece_0.1.0], W = [ ], Nr = [ ], QC = [ ].
   - If the depth level is equal to N:

     Delete subconcepts within the waitlist.

     Example: Cp = [ ], Ca = [Ancient Greece_0.1.0], W = [ ], Nr = [ ], QC = [ ].

Phase 3: Output

A report should be generated with the following elements:
List of Analyzed Concepts: Ca.
List of Conceptual Quarcks: QC