**NARS in Python – Technical Documentation**

**Interface**

**Input Channel**

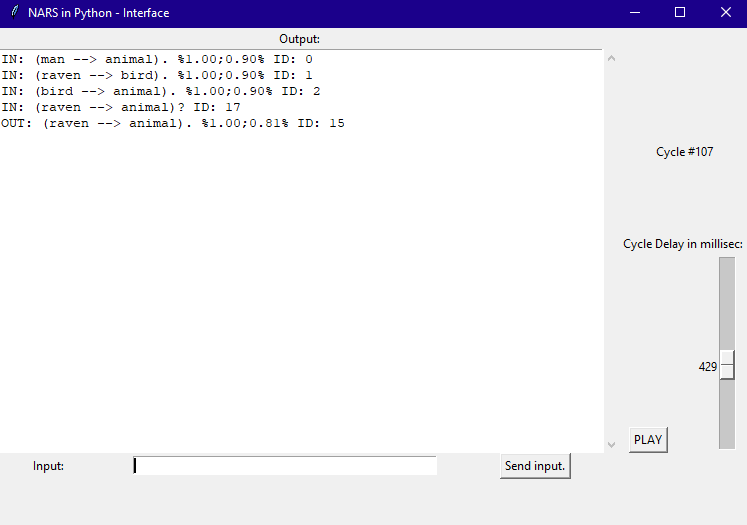
A string from the input channel is parsed into a Sentence, which is then encapsulated in a Task and placed into the system’s Overall Task Buffer. Statement brackets are written using parentheses (i.e. (S-->P).)

The input channel accepts these Narsese Sentences:

* Judgment
* Question

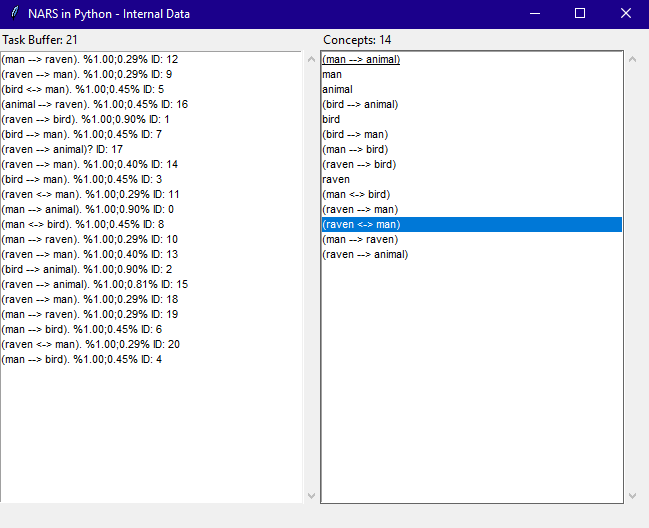
Compound Terms must be entered in prefix format (i.e. (\*,a,b)) since infix format is not currently supported (i.e. (a\*b)). The exception is for Statement Terms, which are a special form of Compound Term: the copula should be between the subject and predicate Terms.

**GUI Interface**

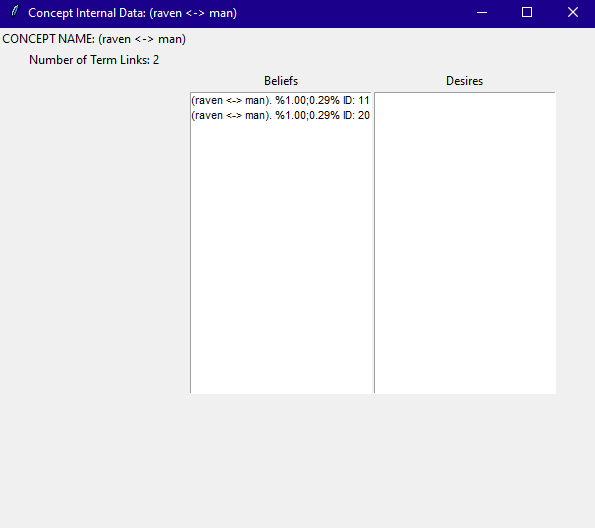
****

The GUI interface can be used to send inputs, pause / play system execution, and speed up / slow down the system’s execution.

**GUI Internal Data**



The system’s internal data can be viewed in the internal data GUI. This displays the contents of the Task Buffer (Tasks) and the Memory (Concepts) sorted by Priority. You can click on a Concept for more info:



**Object Classes and Data Structures**

**Task**A Task can be *input* or *derived*. Derived Tasks contain sentences which have 2 or more pieces of evidence in its evidential base.

**Sentence and Stamp**

Each Sentence has a Stamp, which contains the Sentence’s metadata.

The Sentence’s Evidential Base is an array of pointers to the Sentences from which the Sentence was derived.

**Concept***Conceptualizing* is the process of creating a new Concept, which is named by a term.

Each Concept contains:

* A *belief* Table holding processed *judgments* about the Concept. Atomic Concepts (that contain no copula) will have empty belief Tables, but are term-linked to Statement Concepts which may have non-empty Tables.
* A *desire* Table holding processed *goals* about the Concept. Atomic Concepts (that contain no copula) will have empty desire Tables, but are term-linked to Statement Concepts which may have non-empty Tables.
* A Bag of *task-links*, which link to Tasks related to the Concept.
* A Bag of *term-links*, which link to other Concepts related to the Concept by a shared immediate term.

**Symmetric Statements:**

***Symmetric Statement like S<->P and its reverse Statement P<->S are always treated as equivalent – equality ‘==’ between 2 symmetric StatementTerms checks for this*.** If the Concept for it does not exist, it will be stored in the Concept for its reverse Statement. If neither exist, the Concept is created. Therefore, the order of S and P for the symmetric Statement Concept is determined by the order the system saw first.

**Tables**Tables (belief table and desire table) are stored in Concepts. They are double-ended priority queus that store Narsese Sentences sorted by Confidence. When the Table overflows, the Sentence with the lowest Confidence is purged.

**Bag**The Bag consists of an array of buckets (1-100), and a pointer that points to the currently selected bucket. Objects can be placed into the bag, where they are first wrapped inside an Item with a Priority value, and then inserted into the corresponding bucket based on Priority.

When an item is to be randomly removed from the bag:

* The pointer moves to the next non-empty bucket
* A random number is generated
* If the random number passes the bucket’s probability threshold, an item is removed randomly (uniformly) from the bucket. Otherwise, the pointer moves to the next non-empty bucket.
* This process is repeated until an item is removed.

Items are also stored inside a lookup table, where the Item’s *key* is the hash of the Item’s contained object -- unless it is a Concept.A Concept is named by its term, so its *key* is simply the term converted to a string.

**Priority**Each item in the Bag has Priority. The Priority decays by a multiplied factor when the item is returned to the bag.

**Buffer**A Buffer is a double-ended priority queue that holds Items sorted by Priority. The highest-priority Item is removed when taking from the Buffer. If the Buffer overflows, the lowest-priority items are purged.

**Algorithms**

**Main Control Loop:**

The system either *Observes* a task from its experience buffer, or it *Considers* a Concept from its Memory. The proportion of time the system spends on either process depends on a system parameter, its *Mindfulness*.

**Task Processing:***Initial processing* occurs the first time a task is selected.

*Continued processing* occurs after initial processing, and subsequently whenever the task is selected again.

* **Judgment:**
  + *Initial Processing*
    1. The Judgment’s immediate subterms (subject and predicate) are conceptualized.
    2. The Judgment itself is conceptualized, and bidirectionally term-linked to its subject and predicate concepts.
    3. The Judgment undergoes Revision with the most confident belief in
    4. The Judgment is added directly to the belief table.
    5. **END PROCESSING**
  + *Continued Processing*
    1. First, the Judgment’s corresponding Concept is peeked.
    2. Then, a belief is pulled from a semantically related Concept (this may be from the Task’s Concept itself, which will result in Revision)
    3. Both the Task and related belief are fed into the ***Inference Engine***, which returns derived Tasks.
    4. **END PROCESSING**
* **Question:**
  + *If there is at least one belief matching the Question:*
    1. If it’s a Question from the Input Channel that has not already been answered, print the most confident belief to the output channel.
       1. The Question is marked as answered
    2. If there was an answer to the question:
       1. Perform inference using the answer and a semantically related belief.
       2. **END PROCESSING**
    3. If there are no answers to the Question:
       1. Derive new Questions using backwards inference with a semantically related belief.
       2. **END PROCESSING**
* **Goal:**
  + *Initial Processing*
    1. TBD
  + *Continued Processing*
    1. TBD

**Inference Engine:**

The Inference Engine takes as input one Task and one Belief, and outputs one or more Tasks resulting from the inputs. It assumes the Sentences from the input have distinct (non-overlapping) Evidential Bases, and does not check for this; however, it does merge the parents’ Evidential Bases into the derived Sentences’ Evidential Base.

1. Sentences that will result in Tautologies are discarded. (TODO: pick better beliefs so this the engine doesn’t get sentences like this)
2. The relation between the engine’s input Task and Belief sentences is identified, to determine the appropriate inference rules.
3. One or more derived Tasks are generated using the appropriate inference rules.
4. The engine’s input Task is marked as having interacted with the engine’s input Belief so that they won’t interact again.