~~Thought~~ Synthesis of Semantic Structures

D. Gueorguiev 8/27/2020

Particle model for thought synthesis

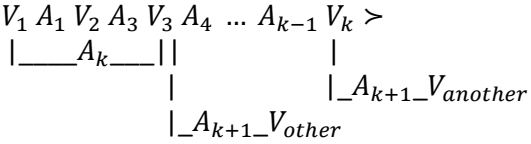
Thought and Particle Signatures

Let P be a thought particle. A thought particle can be an *object particle* denoted with V (corresponding to a vertex in the thought DAG) or a *connecting particle* denoted with A (corresponding to an arc in the thought DAG). The object particle is also known as *V-particle* and the connecting particle is known as *A-particle*.
When parsing a new thought we construct the thought path which is also known as the *main thought path*:

Fig 1
$\prec V_1\ A_1\ V_2\ A_3\ V_3\ A_4\ \ldots\ A_{k-1}\ V_k \succ$

When a new thought is introduced to the environment various processes take place spontaneously.
In the process *Association* the thought path is augmented to a DAG which may look like:

Fig 2



Compound object particle is a subset of a thought path starting with object particle V_{start} followed by connecting particle A_1, etc, and ending with object particle V_{end}. It is denoted by square brackets in which the sub-particles are enclosed:

$[\ V_{start}\ A_1\ V_1\ \ \ldots\ V_{end}\ ]$

Thought radical is a piece of a thought which starts and/or ends with connecting particle. It is denoted by braces in which the sub-particles are enclosed:

$\{\ A_{start}\ V_1\ \ldots\ A_{end}\ \}$

The radical can be considered a generalized connecting particle analogous to the compound object particle.

Signature of a thought particle is a matrix $S[i,j]$ where $0\ <\ i\ <\ m, 0\ <\ j\ <\ n$
Rule for calculation of the thought signature of a compound thought particle

$V_{comp}\ =\ [\ V_{start}\ A_1\ V_2\ \ldots\ V_{end}\ ]$
$sig(V_{start})\ =\ S_{start}[i,j]$

$sig(A_1) = S_1[i,j]$
$sig(V_2) = S_2[i,j]$
$sig([V_{start}\ A_1\ V_2]) = \mathcal{L}A_1(sig(V_{start})) + \mathcal{R}A_1(sig(V_2))$ which is also $m \times n$ matrix

The last line can be written with the following syntax using radicals:

$$sig([V_{start}\ A_1\ V_2]) = sig(\{V_{start}\ A_1\}) + sig(\{A_1\ V_2\})$$

Obviously, the signature of every compound thought particle is $m \times n$ matrix.
Every connecting particle signature encodes the operation $\mathcal{L}A$ which will be applied to the object particle on the left and the operation $\mathcal{R}A$ which will be applied to the object article on the right. Those operations $\mathcal{L}A$ and $\mathcal{R}A$ will preserve the original information contained in the signature of object particle which is being operated on with an additional information pertaining to the link particle.

Arcs which are not on the main thought path and which connect V-particles on the main thought path are modeled via A-particles. The latter have a special property turned on - `*remote*` of type `*link*` for arcs connecting V-particle on the main thought path and a V-particle in a different thought. The property `*secondary*` of type `*link*` is turned on for A-particle which connects two V-particles on the main thought path. We write this statement as:

$$prop(A_k, \text{`}link\text{`}, \text{`}secondary\text{`}) = true$$
$$prop(A_{k+1}, \text{`}link\text{`}, \text{`}remote\text{`}) = true$$

When computing the signature of a compound particle P with secondary path given for example with:

$P = [\ V_1\ A_1\ V_2\ A_3\ V_3\ ]$
$\quad\quad |\_\_\_\_A_k\_\_\_|$

We serialize the two paths

$$sig(P) = sig([V_1\ A_1\ V_2\ A_3\ V_3]) + sig([V_1\ A_k\ V_3])$$

The *secondary* and *remote* link properties will affect the $\mathcal{L}A$ and the $\mathcal{R}A$ operators in a very specific way. For the case of the secondary link $\mathcal{L}A_k(V_1)$ and $\mathcal{R}A_k(V_3)$ will shift the signatures of V_1 and V_3 such that the final matrix $sig(V_1) + \mathcal{L}A_k(V_1) + sig(V_3) + \mathcal{R}A_k(V_3)$ will contain information for this secondary link and the particles it connects.

Similar expansion holds for the case of remote link. For instance, for the subparticle $P = [V_1\ A_1\ V_2\ A_3\ V_3]$ shown on Fig 2 we can write

$$sig(P) = sig([V_1\ A_1\ V_2\ A_3\ V_3]) + sig([V_1\ A_k\ V_3]) + sig([V_3\ A_{k+1}\ V_{other}])$$

Here $sig([V_3\ A_{k+1}\ V_{other}]) = \mathcal{L}A_{k+1}(V_3) + \mathcal{R}A_{k+1}(V_{other})$

For the case of the remote link $\mathcal{L}A_{k+1}(V_3)$ and $\mathcal{R}A_{k+1}(V_{other})$ will shift the signatures of V_3 and V_{other} such that the final matrix $sig(V_3) + \mathcal{L}A_{k+1}(V_3) + sig(V_{other}) + \mathcal{R}A_{k+1}(V_{other})$ will contain information for this remote link as well as the particles it connects including the remote object particle

V_{other}. For more up-to-date description of semantic signature see document The Signature Of Semantic Structures.

Particle properties and encoding them within particle signature

Properties are encoded in specific regions in the particle signature. Each property has a type, a name and value. Property can be thought of as magnifying lens which exposes particular region from the particle signature. Not all properties expose bits from the particle signature which are adjacent and some properties may expose overlapping areas from the signature. In other words:
Let us have p defined properties on particle P. Then it is possible that

$$prop(P, ptype_i, pname_i) \cap prop(P, ptype_j, pname_j) \neq \emptyset \; for \; some \; 0 < i, j < p$$

Let $PReg = prop(P, ptype, pname)$ and $LReg = len(PReg)$. Then for some property P there may exists a continuous block B of bits from the particle signature such that:

$addr(B[0]) = addr(PReg[0])$ and $addr(B[len(B) - 1]) = addr(PReg[LReg - 1])$
$prop(P, ptype, pname) \cap B \neq \emptyset$

The tendency of a particle to establish attraction/repellence toward another particle is achieved through the mediation of *match-seeking* particle (aka M-particle) or *repelling particle* (aka R-particle).

Match-seeker particle
Match-seeker particle is denoted with M aka M-particle. We do not usually depict this kind of particles in our graph representations.
Attaches to a specific property with specific prop-name and prop-value. The property value is a vector which is a key allowing the match-seeker particle to be attached to this property. The match-seeker particle exposes a pattern serving as an attraction of an association link particle which would recognize the pattern and attach to the match-seeker. Each match-seeker particle has a property *`charge`* (type *`default`*) with a value indicating the strength of the charge.

$$M(key\_pattern, pvalue) - V(pvalue)$$
$$|$$
$$A(key\_pattern, key\_pattern2) \text{————} M(key\_pattern2, pvalue2) - V_{other}(pvalue2)$$

Repelling particle
Repelling particle is denoted with R aka R-particle. We do not usually depict this kind of particles in our graph representations.
Attaches to a specific property with prop-value, a vector allowing a repelling particle to be attached to this property. The repelling particle exposes a pattern serving to repel an *association-link* particle which has a property matching the pattern. Each repelling particle has a property *`charge`* (type *`float`*) with a value indicating the strength of the charge.

To each property of object particle V can be attached multiple M-particles and R-particles.

Synthesizing new properties and removing existing properties
Property removal and addition of new properties can happen at *t-execution*.
The syntax for creating a new property and removing/unregistering an existing one is:

```
prop_new = register-property(particle, property_type, property_name)
unregister-property(particle, property_type, property_name)
```

A property is represented by property matrix
$P[i,j] = prop(V, prop\_type, prop\_name), 0 < i < m_p;\ 0 < j < n_p$ such that $m_p < m$ and $n_p < n$. Also for each i and j there exist i_1 and j_1 such that $P[i,j] = S[j_1, j_1]$. Here $S = sig(V)$ and $0 < i_1 < m, 0 < j_1 < n$.

Property affinity
If two properties of two different particles have affinity for each other there will be either M-particle or R-particle which will attach to those properties.

Adding properties at *t-recombination*
Properties can be cloned from another existing particle when a new particle is created.

At the time of the *initial t-processing,* naked particles are created which are wired-up in *t-path*.
Naked particle – particle which have only couple of properties - a property of type *`text`* and property of type *`association`*. For a naked particle the *t-association* phase has not commenced yet.
In the *t-analysis* phase through a set of sequential optimization steps the naked particles are determined their type (V-particles, A-particles) and are "dressed-up" with properties.

Thought Association (*t-association*)
Identify particles with similar signatures and link them through M-particles and R-particles using *similarity-association*.

Example:
Context:
Apostrophe plus s denotes possession.
I know John who is my friend. John has children.
New thought:
Ivan is John's son.
The we can write in symbolic notation:
$\prec [V_{101}\ A_{101}\ V_{102}\ A_{102}\ V_{103}]\ A_{103}\ V_{104}\ A_{104}\ V_{105} \succ$
$\prec V_1\ A_1\ V_2\ A_2\ V_3\ A_3\ V_4\ A_5\ V_5\ A_6\ [V_6\ A_7\ V_8] \succ$
$\prec V_3\ A_8\ V_9\ A_9\ V_{10} \succ$

Let $V_{apostrophe} = [V_{101}\ A_{101}\ V_{102}\ A_{102}\ V_{103}]$ with
$prop(V_{apostrophe}, `text`, `all\ text\ values`) = list(\text{“}Apostrophe\ plus\ s\text{”}, \text{“}'s\text{”})$

The new thought contains the following naked particles (N-particles):
$\prec N_{11}\ N_{12}\ N_{13}\ N_{14} \succ$

$text(N_{11}) = \text{“}Ivan\text{”}$
Here we utilized the shortcut $text(N_{11}) = prop(N_{11}, `text`, `text`)$. Similarly:
$text(N_{12}) = \text{"}is\text{"}$
$text(N_{13}) = \text{"}John's\text{"}$
$text(N_{14}) = \text{"}son\text{"}$

t-association process starts by sending *default-association* particles to attach to each naked particle on the left and on the right as follows:
$\prec N_{11}\ DA_1\ N_{12}\ DA_2\ N_{13}\ DA_3\ N_{14} \succ$
Then a set of M-particles are created attaching to the *key-match:name* property of each naked particle. Each M-particle which binds to the same *key-match:name* property of the naked particle creates slightly different binding pattern based on the textual representation of the naked particle. Each of those M-particles will attempt to attract *similarity-association* particle (SA-particle) bound to already processed V-particles. For the case of N_{13} one of those M-particles will attract SA-particle associated with V_3 and another one of those M-particles will attract SA-particle associated with $V_{apostrophe}$.

Details on the algorithm performing similarity association and dressing of naked particles are elaborated and discussed in *Supplement-9-5-29 (page1-page10)*.

Partially dressed particle
Naked particle which has been identified by name and its position in the thought path with respect to the other partially dressed particles has been identified

Property inference in partially dressed particles
Let us assume we have a partially dressed particle N together with other partially dressed particles N1, N2, … , Nk which are part of the same thought path. We would like to infer as many properties of those as possible.

S – the set of all allowed property-value pairs for particle N
S1, S2, … , Sk are subsets of S
Existing chain C1: N has the pairs (pi,vj) where (i,j) belongs to S1
Existing chain C2: N has the pairs (pi,vj) where (i,j) belongs to S2
. . .
Existing chain Ck: N has the pairs (pi,vj) where (i,j) belongs to Sk

New chain Cnew: N has only (p0,v0) where p0 is the *name* property and v0 is the *name* value.
Goal: find out the set Snew containing the indices (I,j) of all pairs (pi,vi) which belong to N in the new chain Cnew.
~~One can obtain Snew by constructing Bayesian network from C1, C2, … , Ck and the sets S1, S2, … , Sk.~~