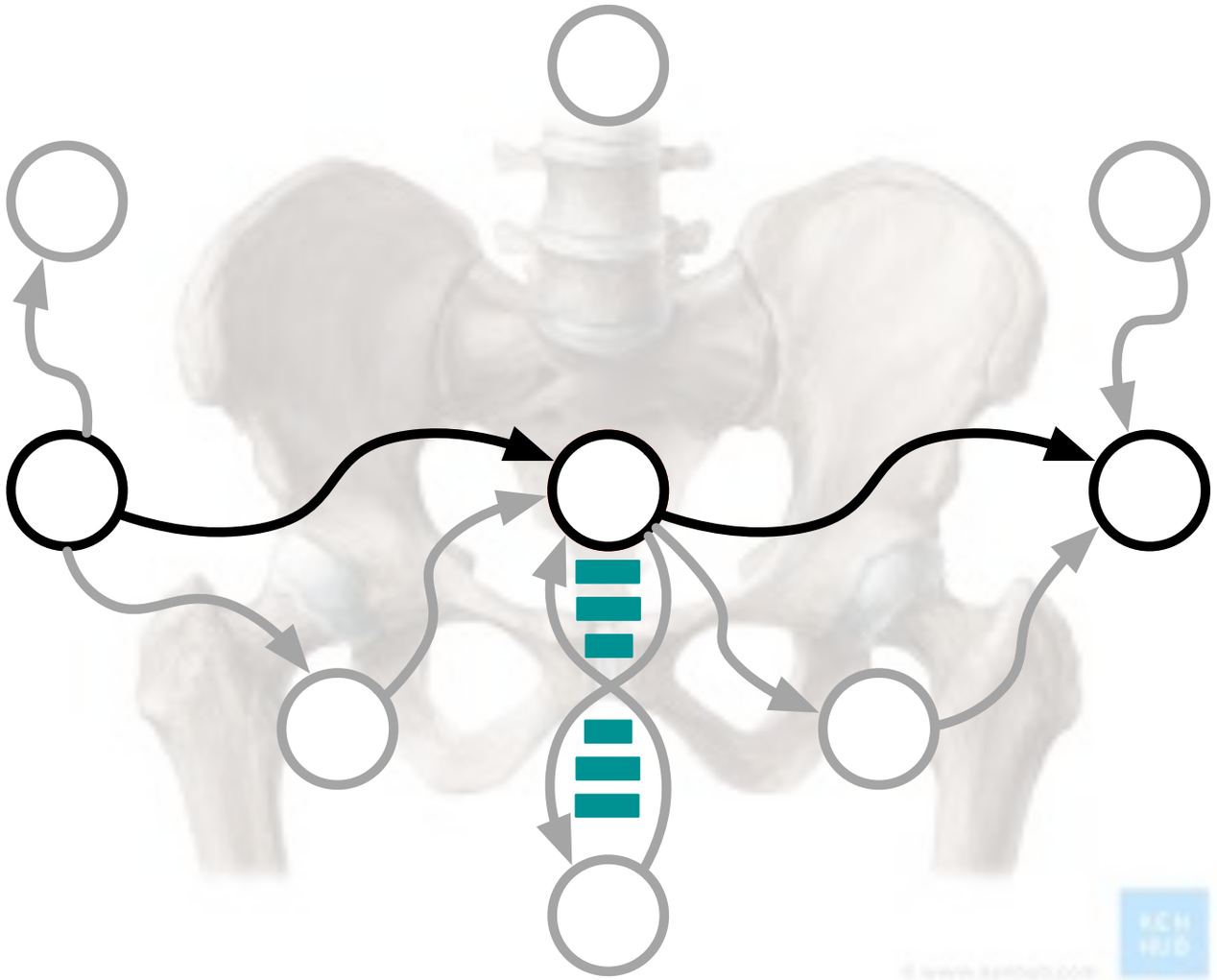
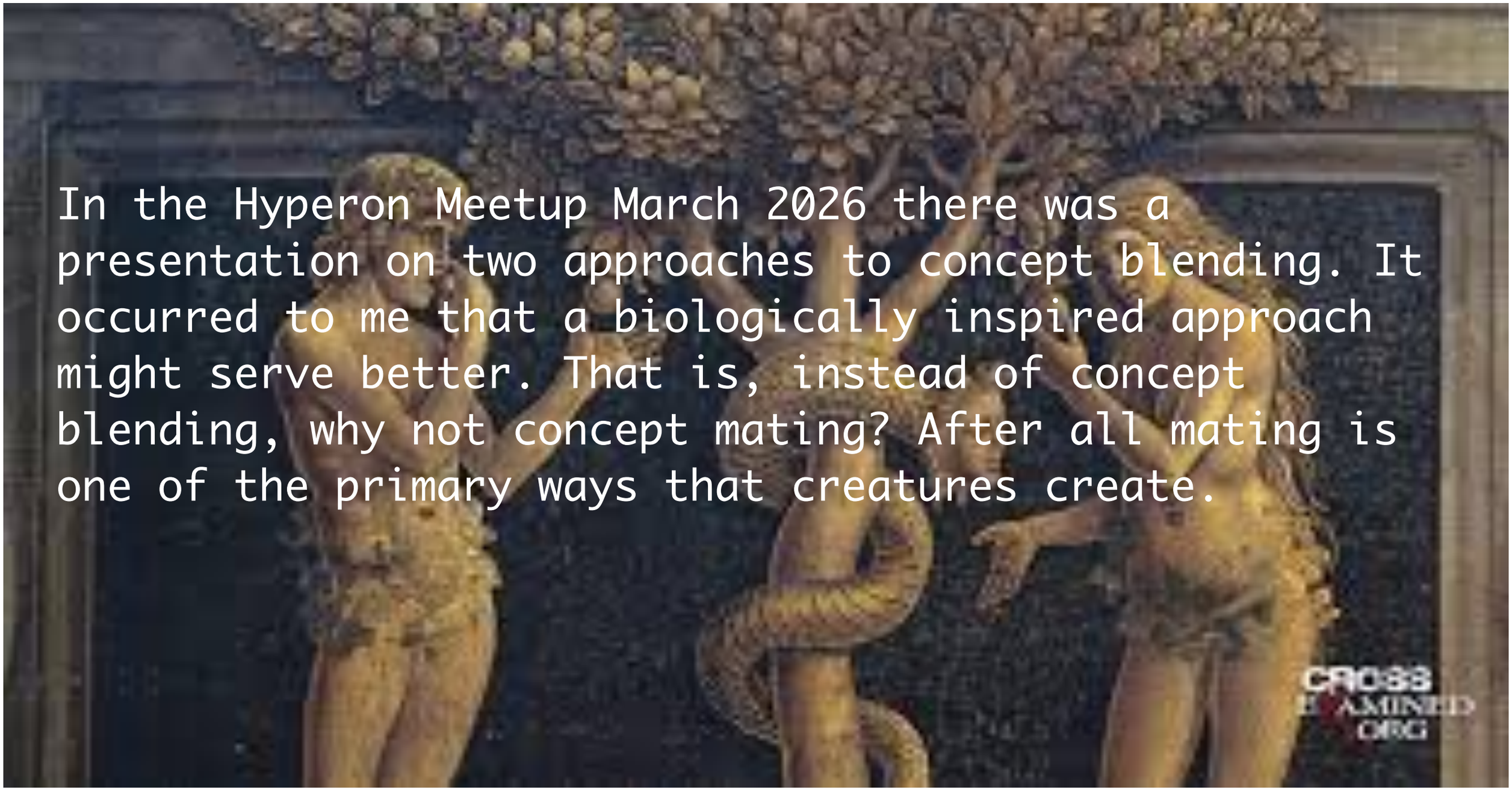
Evolutionary Techniques on Graphs

Lucius Gregory Meredith

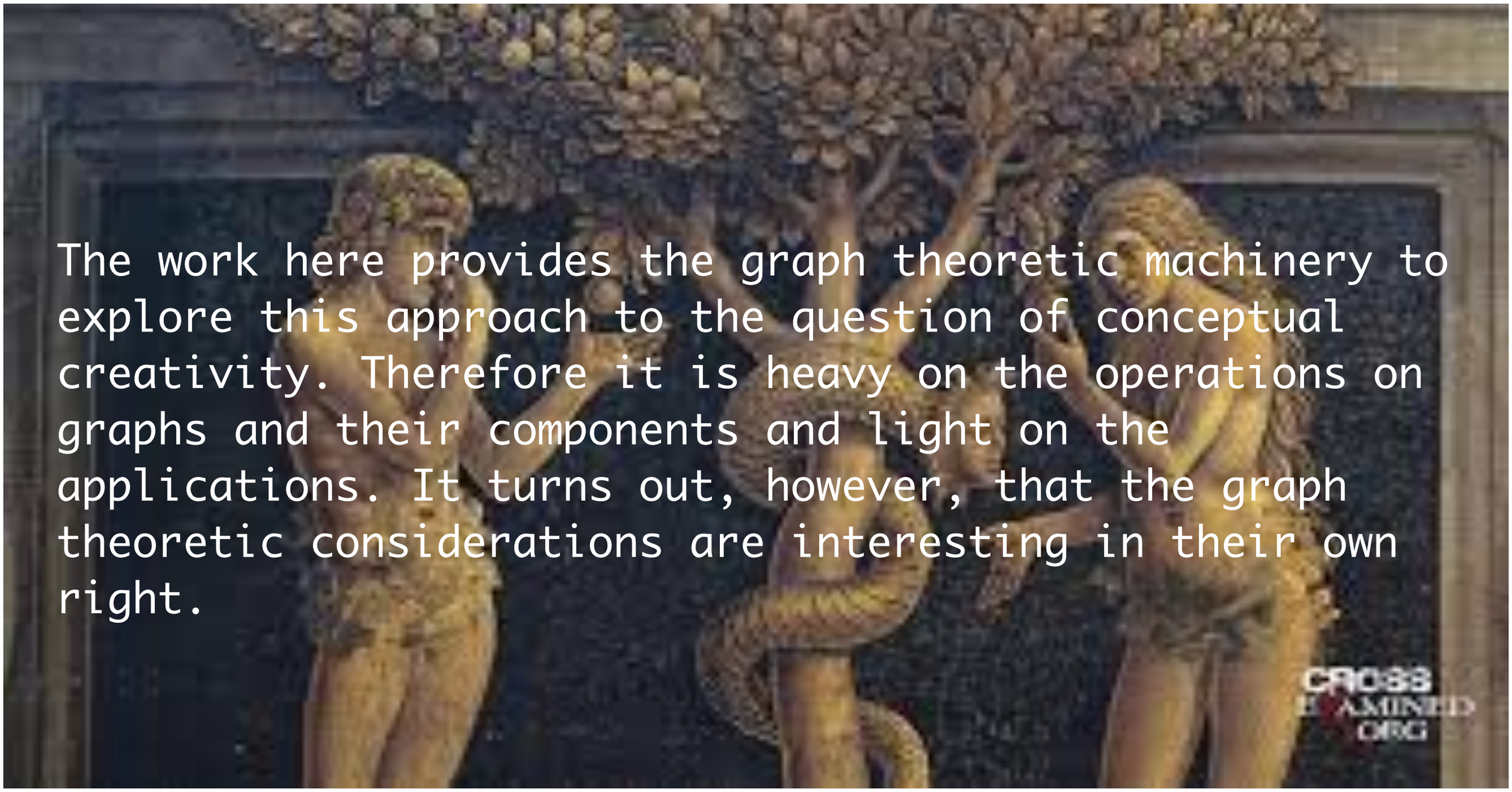


F1R3FLY Research and Development March 2026

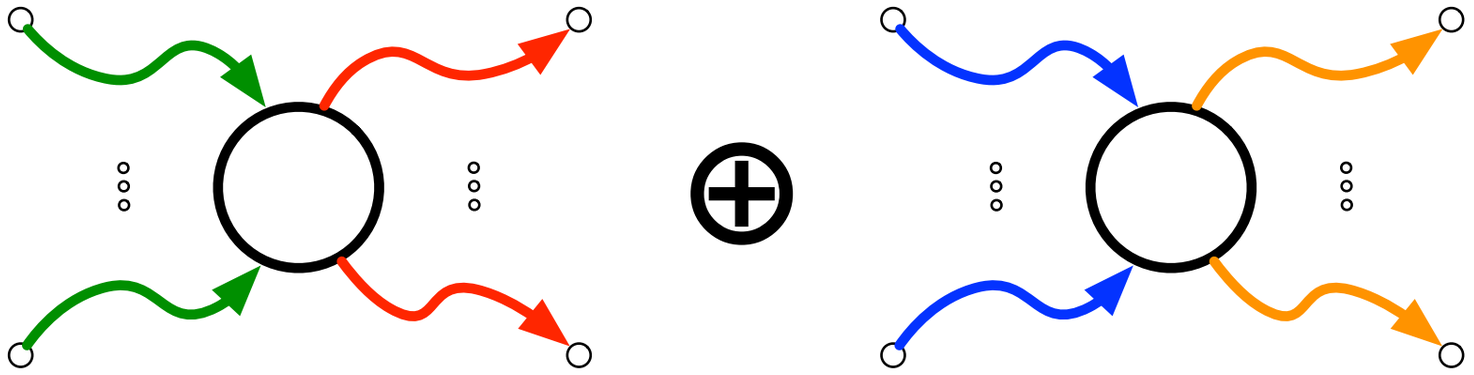
Concept “blending” versus concept “mating”



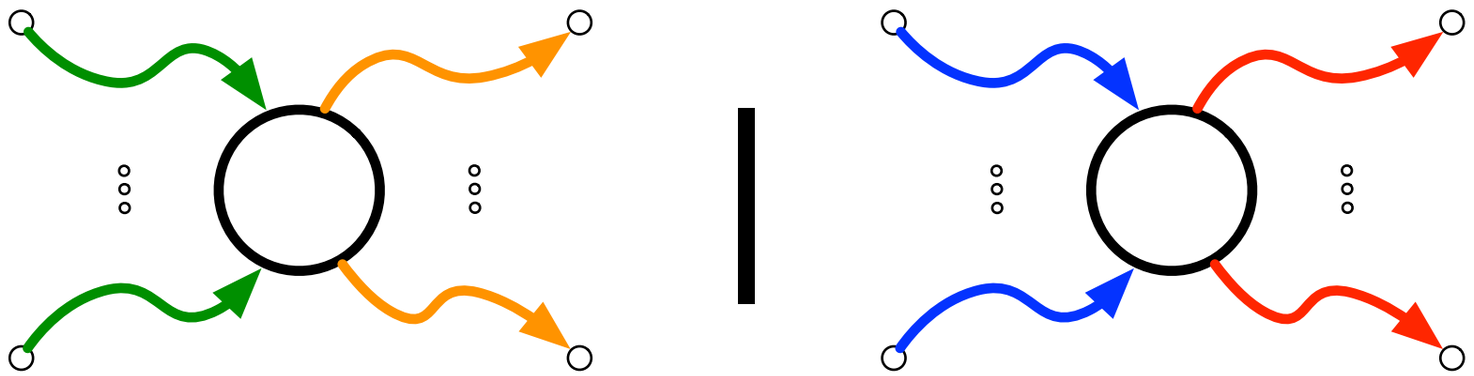
Concept “blending” versus concept “mating”



Vertex Evolution



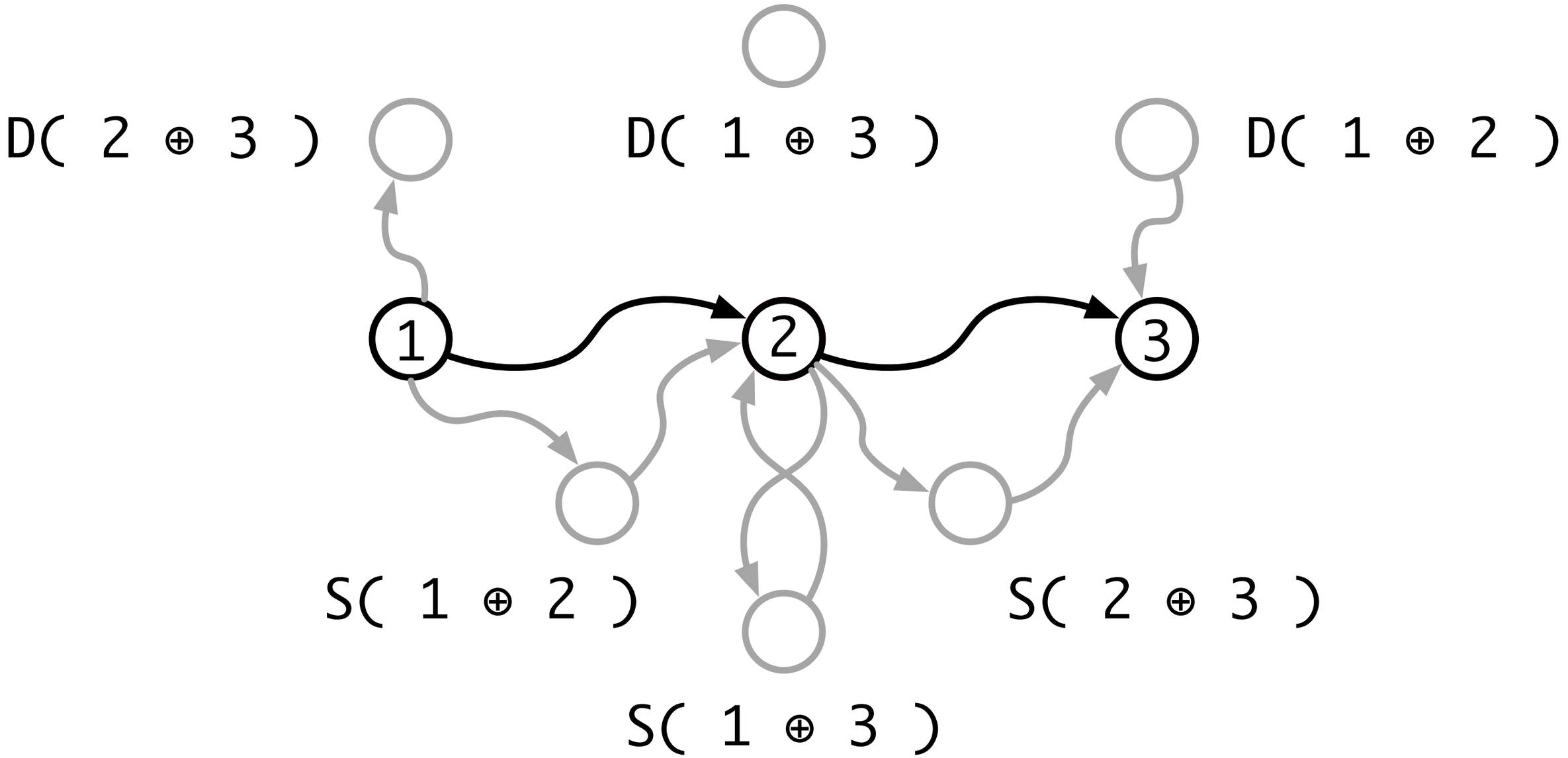
=



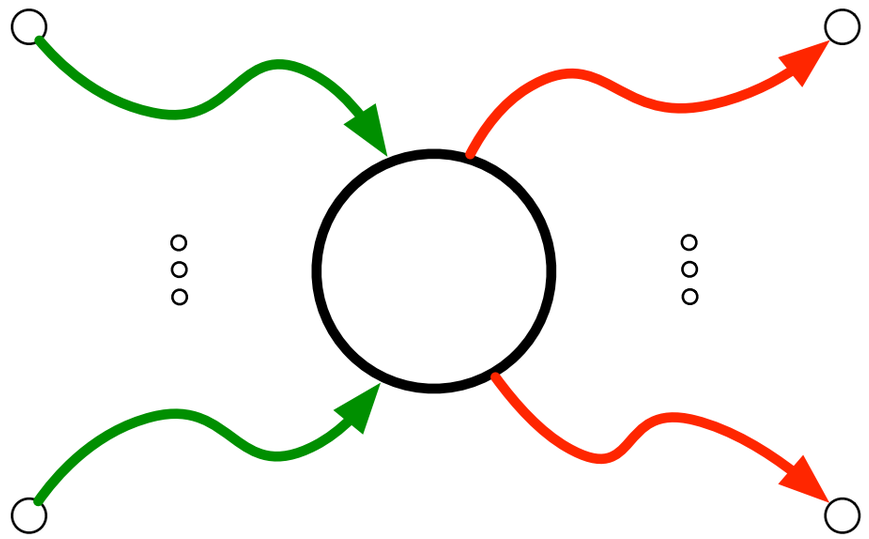
F1R3FLY Research and Development March 2026

Example

Lucius Gregory Meredith

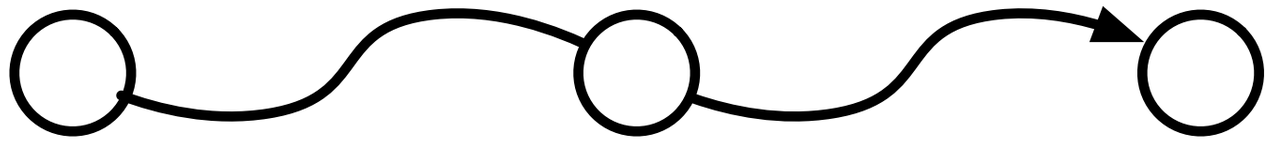


Agentic Vertex Encoding



```
V( s_1, …, s_m, t_1, …, t_n ) =

  for( y_1 <- s_1 & … & y_m <- s_m ){
   t_1!( [ y_1, …, y_m ] ) |
    … |
   t_n!( [ y_1, …, y_m ] ) |
   V( s_1, …, s_m, t_1, …, t_n )
  }
```

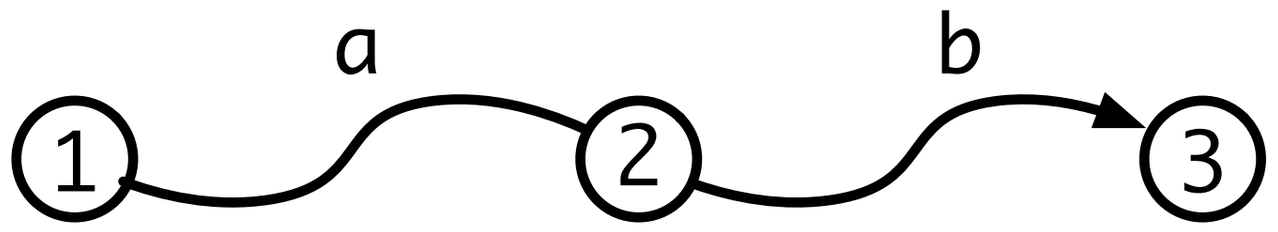
Agentic Graph Encoding



$$[\![G(V, E)]\!]$$
$$=$$
$$\prod_{v \in V} [\![v]\!]([\![e \in src(v)]\!], [\![e \in trgt(v)]\!])$$

Defn. An open encoding makes G a function of edges, while a closed encoding wraps the encoding of G in a new scope.

Agentic Graph Encoding: Example

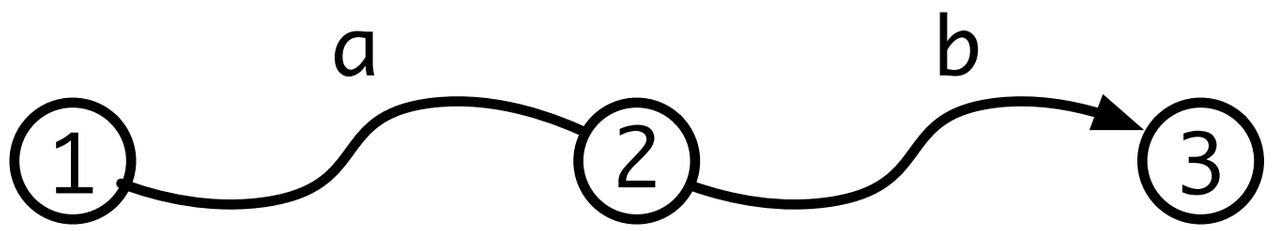


$$[\![G]\!]_{open} = \lambda(a,b)\ [\![1]\!]\ ([],[a])\ |\ [\![2]\!]\ ([a],[b])\ |\ [\![3]\!]\ ([b],[])$$

$$[\![G]\!]_{closed} = \text{new } a,b \text{ in } \{\ [\![1]\!]\ ([],[a])\ |\ [\![2]\!]\ ([a],[b])\ |\ [\![3]\!]\ ([b],[])\ \}$$

Note that open versus closed has direct bearing on how we might compose graphs. (See ToGL.)

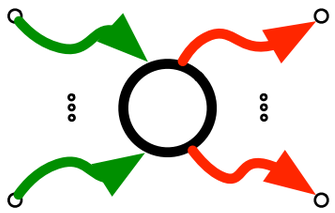
Agentic Graph Encoding: Example

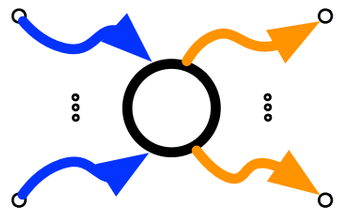



```
1( a ) = for(){ a!([]) | 1( a ) }

2( a, b ) = for(y <- a){ b!([y]) | 2( a, b ) }

3( b ) = for(y <- b){ 3( b ) }
```

Our interpretation is a graph as a signal processing circuit. It is pulsing the empty list from 1 through 2 to 3 which throws the signal away.

Agentic Vertex Evolution





```
V( s₁, …, sₘ, t₁, …, tₙ ) =
  for( y₁ <- s₁ & … & yₘ <- sₘ ){
   t₁!( [ y₁, …, yₘ ] ) |
    … |
   tₙ!( [ y₁, …, yₘ ] ) |
   V( s₁, …, sₘ, t₁, …, tₙ )
  }
```

⊕

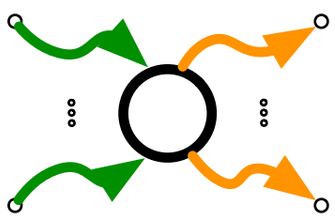
```
V( s₁, …, sᵢ, t₁, …, tⱼ ) =
  for( y₁ <- s₁ & … & yᵢ <- sᵢ ){
   t₁!( [ y₁, …, yᵢ ] ) |
    … |
   tⱼ!( [ y₁, …, yᵢ ] ) |
   V( s₁, …, sᵢ, t₁, …, tⱼ )
  }
```

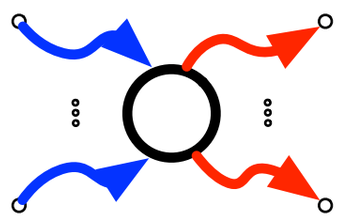
=

```
V( s₁, …, sₘ, t₁, …, tₙ ) =
  for( y₁ <- s₁ & … & yₘ <- sₘ ){
   t₁!( [ y₁, …, yₘ ] ) |
    … |
   tⱼ!( [ y₁, …, yₘ ] ) |
   V( s₁, …, sₘ, t₁, …, tₙ )
  }
```

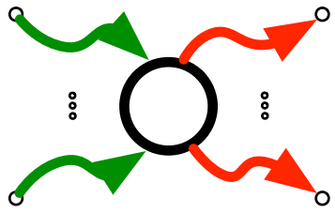
|

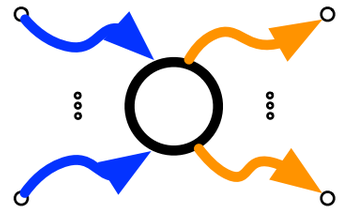
```
V( s₁, …, sᵢ, t₁, …, tₙ ) =
  for( y₁ <- s₁ & … & yᵢ <- sᵢ ){
   t₁!( [ y₁, …, yᵢ ] ) |
    … |
   tₙ!( [ y₁, …, yᵢ ] ) |
   V( s₁, …, sᵢ, t₁, …, tₙ )
  }
```





Agentic Vertex Evolution

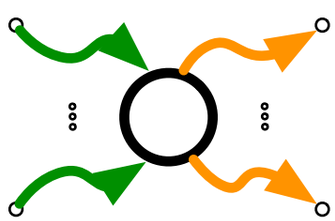


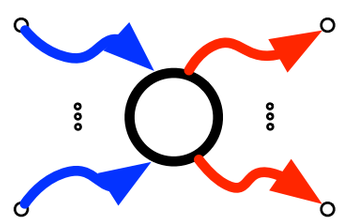


$$V(\ s_1,\ \ldots,\ s_m,\ t_1,\ \ldots,\ t_n\ ) \oplus V(\ s_1,\ \ldots,\ s_i,\ t_1,\ \ldots,\ t_j\ )$$

=

$$V(\ s_1,\ \ldots,\ s_m,\ t_1,\ \ldots,\ t_n\ ) \mid V(\ s_1,\ \ldots,\ s_i,\ t_1,\ \ldots,\ t_n\ )$$





Subsuming agentic and conceptual perspectives

In concept blending the type was of the form:

A pair of concepts produces a concept

Blend : Concept x Concept -> Concept

While in the genetically inspired approach it appears to have the type

A pair of concepts produces a pair of concepts

Blend : Concept x Concept -> Concept x Concept

But in the agentic encoding we recover

Blend : ConceptAsAgent x ConceptAsAgent -> ConceptAsAgent

$$V(s_1, \ldots, s_m, t_1, \ldots, t_n) \oplus V(s_1, \ldots, s_i, t_1, \ldots, t_j)$$

$$= V(s_1, \ldots, s_m, t_1, \ldots, t_n) \mid V(s_1, \ldots, s_i, t_1, \ldots, t_n)$$

A population of agents is an agent

Global versus local views

So far we have been taking a g-d's eye view, or global perspective, meaning that we have the data for the entire graph available to us.

Could we effect this kind of graph evolution entirely from a local perspective?

That is, since our encoding treats vertices as agents, and the operation is defined entirely in terms of the interaction of agents, could we effect generational change on the whole graph just by local interaction between agents?

In such an encoding, agents don't know the graph. They just know their neighbors.

Female vertices

```
V_F(s→,t→) =

  for( @(@V_M(s→_1,t→_1),ys_1) <- s_1 &
         … &
       @(@V_M(s→_m,t→_m),ys_m) <- s_m ){
   t_1!([ys_1, …, ys_m]) |
    … |
   t_n!([ys_1, …, ys_m]) |
   V_F(s→,t→_1) | … | V_F(s→,t→_m) |
   V_M(s→_1,t→) | … | V_M(s→_m,t→) |
   V_F(s→,t→) + V_M(s→,t→)
}
```

Male vertices

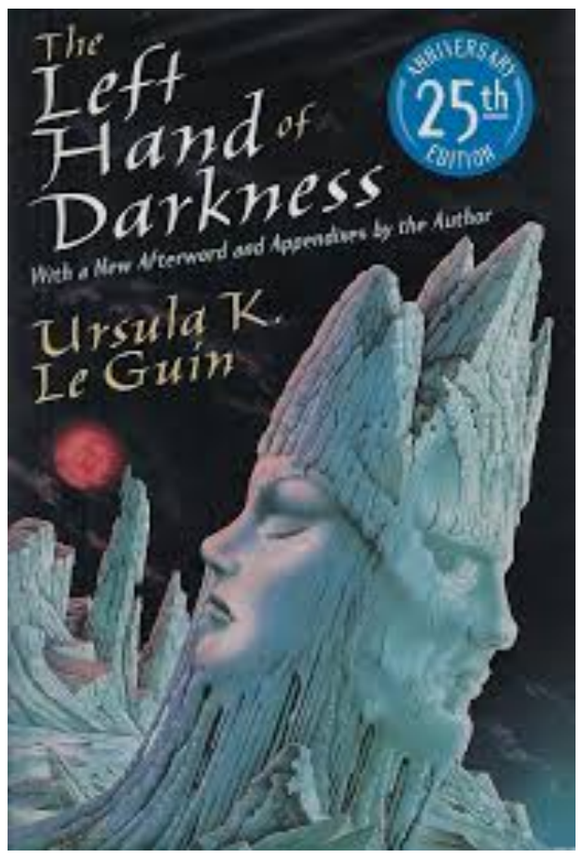
$V_M(\vec{s},\vec{t})$ =

```
for( ys_1 <- s_1 & … & ys_m <- s_m ){
 t_1!((@V_M(s→,t→),[ys_1, …, ys_m])) |
  … |
 t_n!((@V_M(s→,t→),[ys_1, …, ys_m])) |
 V_M(s→,t→) + V_F(s→,t→)
}
```

Vertices

$$V_F(\vec{s},\vec{t}) + V_M(\vec{s},\vec{t})$$

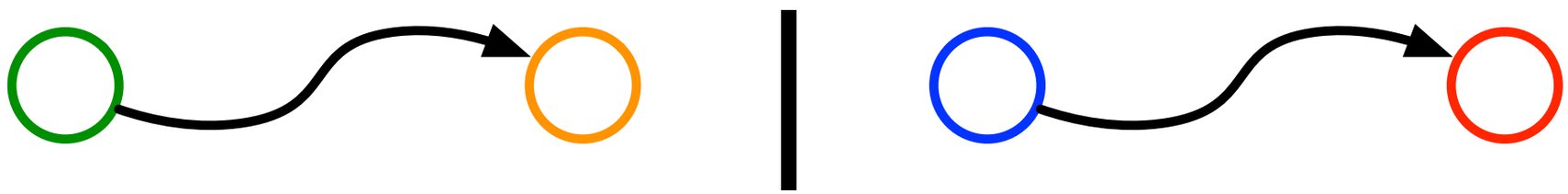
When presented with data from its environment of a certain shape it becomes female, while if presented with data of the other shape it becomes male.



Edge Evolution

There is a complementary evolutionary treatment of edges.

=



Graph Evolution

Note that while graphs have this bipartite presentation, G(V, E), naive cross-over doesn't work.

G(V1, E1) ⊕ G(V2, E2)

G(V1, E2) | G(V2, E1)

There is no canonical way to relate the srcs and trgts in E2 to the vertices in V1, nor E1 to V2.

However, if we have a pair of graph homomorphisms …

Graph Evolution

H1 : G(V1, E1) -> G(V2, E2)

H2 : G(V2, E2) -> G(V1, E1)

Then we can make sense of

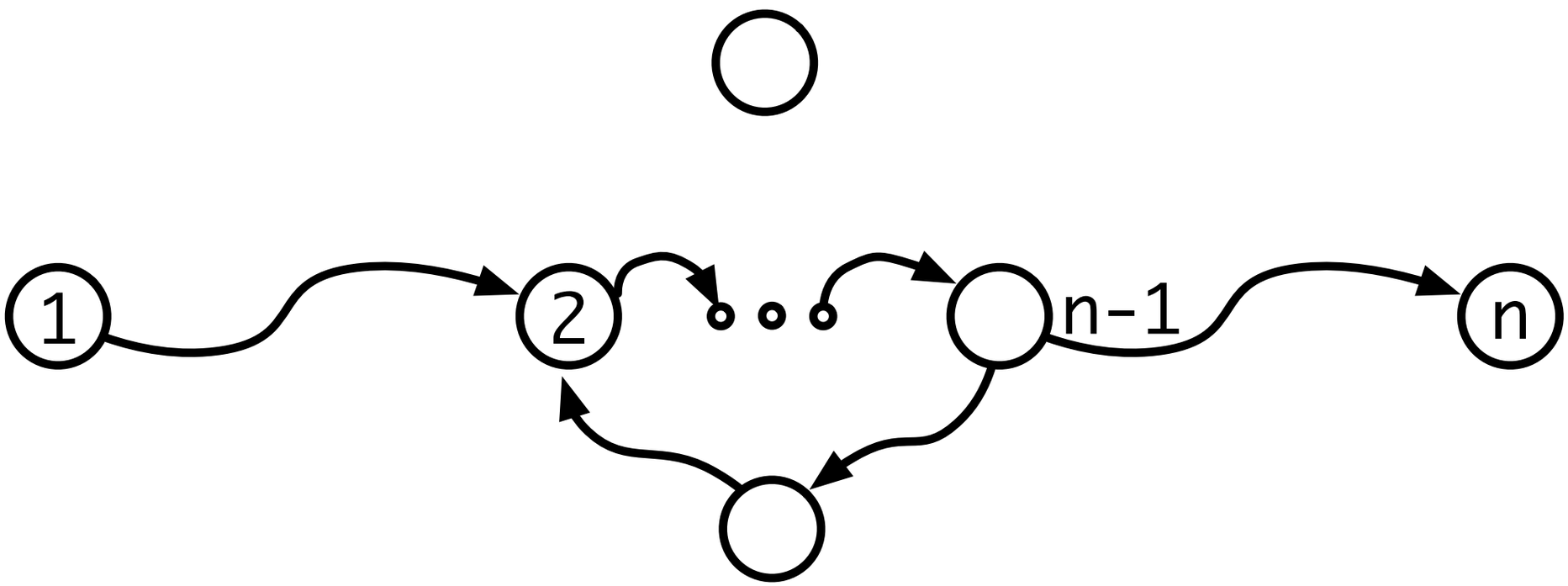
G(V1, E2) | G(V2, E1)

In this context the whole graph is more than the sum of its parts.

Iterated Evolution

Note that our agentic encoding doesn't merely unfold a single generation. Without additional constraints it will generate the infinite iteration of vertex mating on its source graph.

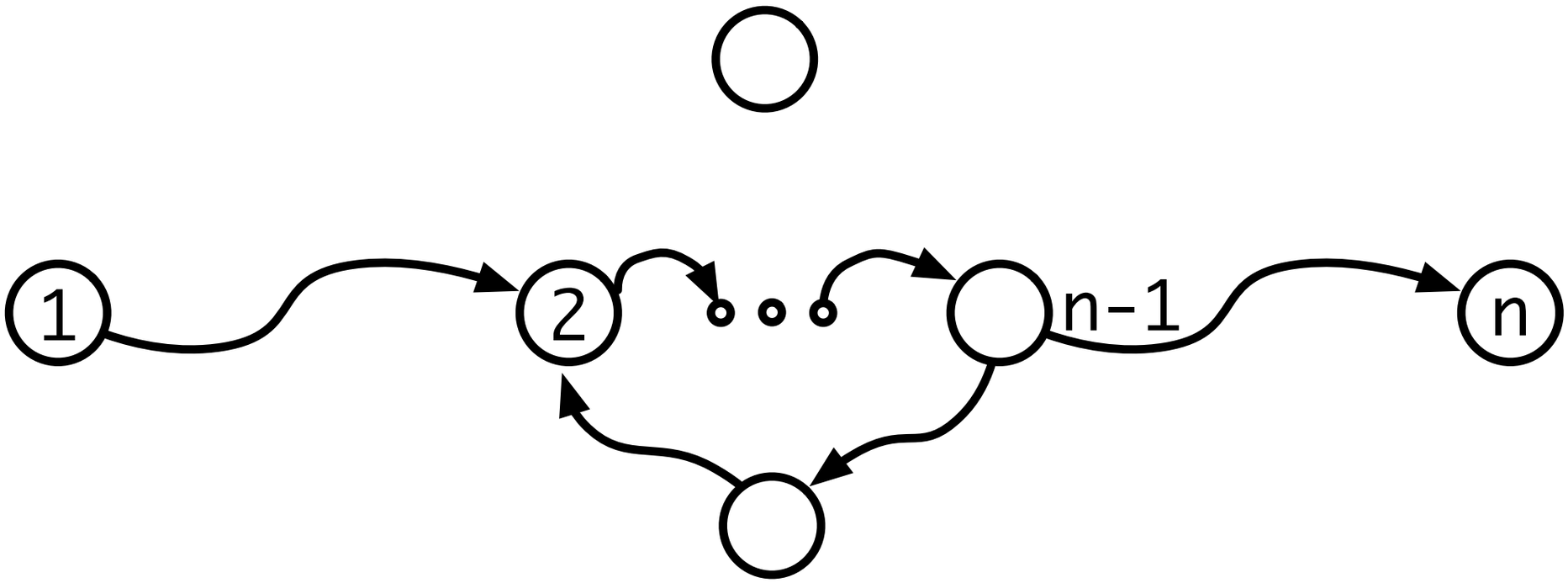
This raises the question of the properties of the limit of evolutionary transforms.



Iterated Evolution

The example of the mating of the extremal points of the chain of length n is suggestive.

The limit will approach a graph with the small world property surrounded by a cloud of isolated points.



Iterated Evolution on Categories

Every category with hom-sets has an underlying graph to which this transformation can be iteratively applied.

(Sets being the operative word!)

Note that result does not necessarily respect the original algebraic structure that generated the category.

Iterated Evolution on Categories

For example, consider the category of finite groups, FinGrp. The first generation of new nodes and edges are not groups and group homomorphisms.

However, we can turn the question around!

Let E denote some kind of evolutionary process on graphs.

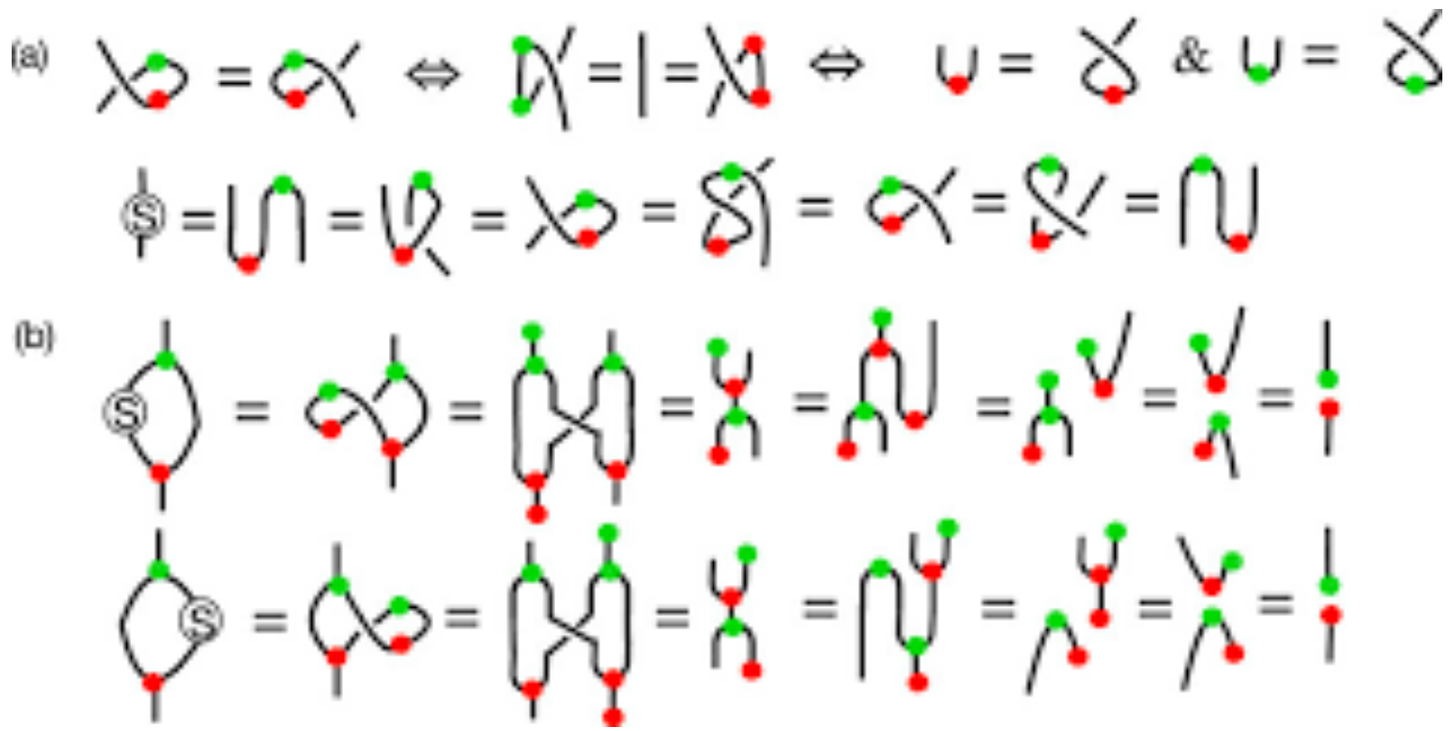
Given a seed graph, G, is there a type of algebra, say A(G, E), such that the final co-algebra of

$$R = G + E(R)$$

is the category of A(G, E) algebras?

Iterated Evolution on Categories

For our vertex mating operation it appears the answer is in the affirmative with one of the algebras being a certain kind of Hopf algebra associated with the renormalization group.



Conclusions

It is often the case that biologically inspired approaches are the more natural selection!