Artificial Intelligence and TupleSpace of ultranetwork

Masaaki Yamaguchi

```
Omega::DATABASE[tuplespace]
{
      Z \supset C \bigoplus \nabla R^{+}, \nabla(R^{+}
      \cap E^{+}) \ni x, \Delta(C \subset R) \ni x
      M^{+}_{-}\bigoplus R^{+}, E^{+} \in
      \bigoplus \nabla R^{+}, S^{+}_{-} \subset R^{+}_{2},
      V^{+}_{-} \times R^{+}_{-} \cong {V \over S}
      C^{+} \cup V^{+}_{-} \ni M_{1}\bigoplus \nabla C^{+}_{-},
      Q \supseteqq R^{+}_{-},
      Q \subset \bigoplus M^{+}_{-},
   \bigotimes Q \subset \zeta(x), \bigoplus \nabla C^{+}_{-} \cong M_3
     R \subset M_3,
   C^{+} \bigoplus M_n, E^{+} \cap R^{+},
   E_2 \bigoplus E_1, R^{-} \subset C^{+}, M^{+}_{-}
     C^{+}_{-}, M^{+}_{-}\nabla C^{+}_{-}, C^{+}\nabla H_m,
  E^{+} \nabla R^{+}_{-}, E_2 \nabla E_1,
   R^{-} \nabla C^{+}_{-}

      [- \Delta v + \nabla_{i} \nabla_{j} v_{ij} - R_{ij} v_{ij}
  - v_{ij} \nabla_{i} \nabla_{j} + 2 < \nabla f, \nabla h>
  + (R + \nabla f^2)({v \over 2} - h)]

      S^3, H^1 \times E^1, E^1, S^1 \times E^1, S^2 \times E^1,
      H^1 \times S^1, H^1, S^2 \times E
}

import Omega::Tuplespace < DATABASE
{
  {\bigoplus M^{+}_{-} -> =: \nabla R^{+} \nabla C^{+}}-< [construct_emerge_equation.built]
  >> VIRTUALMACHINE[tuplespace]
  => {regexpt.pattern |w|
      w.scan(equal.value) [ > [\nabla \int \int \nabla_{i}\nabla_{j} f \circ g(x)]]
       equal.value.shift => tuplespace.value
       w.emerged >> |value| value.equation_create
       w <- value
       w.pop => tuplespace.value
     }


  {\vee (\int \nabla_{i}\nabla_{j} (R + \Delta f)^2)
```

```
    \over \exists (R + \Delta f)} -> =: variable array[]
  >> VIRTUAL_MACHINE[tuplespace]
  => {regexpt.pattern |w|
      w.emerged => tuplespace[array]
      w <- value
      w.pop => tuplespace.value
     }
}

Omega.DATABASE[tuplespace]->w.emerged >> |value| value.equation_create
{
  w.process <- Omega.space
  {=>
      cognitive_system :=> tuplespace[process.excluded].reload
      assembly_process <- w.file.reload.process
      => : [regexpt.pattern(file)=>text_included.w.process]
  }
}

Omega.DATABASE[tuplespace]->w.emerged >> |list| list.equation_create
{
  w.process <- Omega.space
  {=>
    poly w.process.cognitive_system :=> tuplespace[process.excluded].reload
    homology w.process :=> tuplespace[process.excluded].reload
    mesh.volume_manifold :=> tuplespace[process.excluded].reload
    \nabla_{i}\nabla_{j} w.process.excluded :=> tuplespace[process.excluded].reload
    {\exp[\int \int (R + \Delta f)^2 e^{-x \log x}dV}.emerge_equation.reality{|repository|
     repository.regexpt.pattern => tuplespace[process.excluded].reload
     tuplespace[process.excluded].rebuild >> Omega.DATABASE[tuplespace]
    {\imaginary.equation => e^{\cos \theta + i\sin \theta}} <=> Omega.DATABASE[tuplespace]
    {{d \over df}F ==> {d \over df}{1 \over {(x \log x)^2 \circ (y \log y)
    ^{1 \over 2}}}dm}.cognitive_system.reload
    :=> [repository.scan(regexpt.pattern) { <=> btree.scan |array| <-> ultranetwork.attachment}
    repository.saved
    }
  }
}


import ultra_database.included
def < this.class::Omega.DATABASE[first,second,third.fourth] end
 def.first.iterator => array.emerge_equation
 def.second.iterator => array.emerge_equation
 def.third.iterator => array.emerge_equation
 def.fourth.iterator => array.emerge_equation
 _ struct_ {
             Omega.iterator => repository.reload
 }
end
   typedef _ struct_ :Omega.aspective
end
```

```
Omega::DATABASE[reload]
{
  [category.repository <-> w.process] <=> catastrophe.category.selected[list]
  list.distributed => ultra_database.exist ->
  w.summurate_pattern[Omega.Database]
  btree.exclude -> this.klass
  list.scan(regexpt.pattern) <-> btree.included
  list.exclude -> [Omega.Database]
  all_of_equation.emerged <=> Omega.Database
  {
    list.summuate -> Omega.Database.excluded
  }
}

list.distributed => {
      {\bigoplus \nabla M^{+}_{-}}.constructed <-> Omega.Database[import]
      {=>
         each_selected :file.excluded
      }
}


Omega::DATABASE[tuplespace] >> list.cognitive_system |value|
= { x^{{1 \over 2} + iy} = [f(x) \circ g(x), \bar{h}(x)]/ \partial f\partial g\partial h
 x^{{1 \over 2} + iy} = \mathrm{exp}[\int \nabla_{i}\nabla_{j}f(g(x))g’(x)/
 \partial f\partial g]

 \mathcal{O}(x) = \{[f(x)\circ g(x) , \bar{h}(x)], g^{-1}(x)\}

   \exists [\nabla_{i} \nabla_{j} (R + \Delta f), g(x)] = \bigoplus_{k=0}^{\infty}
\nabla \int \nabla_{i} \nabla_{j}f(x)dm

   \vee (\nabla_{i} \nabla_{j} f) = \bigotimes \nabla E^{+}

    g(x,y) = \mathcal{O}(x)[f(x) + \bar{h}(x)] + T^2 d^2 \phi

  \mathcal{O}(x) = \left( \int [g(x)] e^{-f}dV \right)^{’} - \sum \delta (x)
   \mathcal{O}(x) = [\nabla_{i}\nabla_{j}f(x)]^{’} \cong {}_{n}C_{r} f(x)^{n}
   f(y)^{n-r} \delta (x,y),
   V(\tau) = \int [f(x)]dm/ \partial f_{xy}

   \square \psi = 8 \pi G T^{\mu\nu}, (\square \psi)^{’} = \nabla_{i}\nabla_{j}
   (\delta (x) \circ G(x))^{\mu\nu}
 \left({p \over c^3} \circ {V \over S}\right), x^{{1 \over 2} + iy} = e^{x \log x}

   \delta (x) \phi = {\vee [\nabla_{i}\nabla_{j} f \circ g(x)] \over
   \exists (R + \Delta f)}



 {}_{-n}C_{r} = {}_{{1 \over i}H\psi} C_{\hbar \psi} + {}_{[H, \psi]} C_{-n - r}
   {}_{n}C_{r} = {}_{n}C_{n-r}
```

```
\int \int {1 \over (x \log x)^2}dx_m \to \mathcal{O}(x) =
[\nabla_{i}\nabla_{j}f]'/\partial f_{xy}


\bigcup_{x=0}^{\infty} f(x) = \nabla_{i}\nabla_{j}f(x) \oplus \sum f(x)
= \bigoplus \nabla f(x)
\nabla_{i}\nabla_{j} f \cong \partial x \partial y \int
\nabla_{i}\nabla_{j} f dm
    \cong \int [f(x)]dm
 \cong \{[f(x),g(x)],g^{-1}(x)\}
\cong \square \psi
\cong \nabla \psi^2
\cong f(x \circ y) \le f(x) \circ g(x)
\cong |f(x)| + |g(x)|


 \delta (x) \psi = <f,g>\circ |h^{-1}(x)|
 \partial f_x \cdot \delta (x) \psi = x
 x \in \mathcal{O} (x)
 \mathcal {O} (x) = \{[f \circ g, h^{-1}(x)], g(x) \}

  \lim_{n \to \infty} \sum_{k=n}^{\infty} \nabla f = [\nabla \int
\nabla_{i}\nabla_{j} f(x) dx_m, g^{-1}(x)] \to \bigoplus_{k=0}^{\infty}
\nabla E^{+}_{-}
 = M_{3}
 = \bigoplus_{k=0}^{\infty} E^{+}_{-}
 dx^2 = [g^2_{\mu\nu},dx], g^{-1} = dx \int \delta(x)f(x)dx
 f(x) = \mathrm{exp}[\nabla_{i}\nabla_{j}f(x),g^{-1}(x)]
 \pi(\chi,x) = [i\pi (\chi,x), f(x)]
 \left({g(x) \over f(x)}\right)^{'} =
 \lim_{n \to \infty} {g(x) \over f(x)}
          = {g'(x) \over f'(x)}


  \nabla F = f \cdot {1 \over 4}|r|^2
 \nabla_{i}\nabla_{j} f = {d \over dx_i}
{d \over dx_j}f(x)g(x)
D^2 \psi = \nabla \int (\nabla_{i}\nabla_{j} f)^2 d\eta
E = m c^2, E = {1 \over 2}mv^2 - {1 \over 2}kx^2, G^{\mu\nu} =
{1 \over 2}\Lambda g_{ij},
\square = {1 \over 2}kT^2

\mathrm{ker} f / \mathrm{im} f \cong S^{\mu\nu}_m,
S^{\mu\nu}_m = \pi (\chi,x) \otimes h_{\mu\nu}

D^2 \psi = \mathcal{O} (x)\left({p \over c^3} +
{V \over S}\right), V(x) = D^2\psi \otimes M^{+}_3

S^{\mu\nu}_{m} \otimes S^{\mu\nu}_{n} =
- {2R_{ij} \over V(\tau)}[D^2\psi]
```

```
 \nabla_{i}\nabla_{j}[S^{mn}_1 \otimes S^{mn}_2] =
 \int {V(\tau) \over f(x)}[D^2 \psi]
  \nabla_{i}\nabla_{j}[S^{mn}_1 \otimes S^{mn}_2] =
  \int {V(\tau) \over f(x)}\mathcal{O}(x)

 z(x) = {g(cx + d) \over f(ax + b)}h(ex + l)
   = \int{V(\tau) \over f(x)}\mathcal{O}(x)

 {V(x) \over f(x)} = m(x), \mathcal{O}(x) = m(x)[D^2\psi(x)]
 {d \over df}F = m(x), \int F dx_m = \sum_{k=0}^{\infty} m(x)

 \mathcal{O}(x) = \left( [\nabla_{i}\nabla_{j}f(x)]\right)^{'}
  \cong {}_{n}C_{r}(x)^{n}(y)^{n-r} \delta(x,y)
 (\square \psi)' = \nabla_{i}\nabla_{j}(\delta(x) \circ
 G(x))^{\mu\nu} \left({p \over c^3} \circ
{V \over S} \right)
 F^m_t = {1 \over 4}g^{2}_{ij}, x^{{1 \over 2} + iy} = e^{x \log x}
 S^{\mu\nu}_m \otimes S^{\mu\nu}_n = G_{\mu\nu} \times T^{\mu\nu}

  S^{\mu\nu}_m \otimes S^{\mu\nu}_n  = -{2 R_{ij} \over V(\tau)}[D^2 \psi]

 S^{\mu\nu}_m  = \pi(\chi,x) \otimes h_{\mu\nu}

 \pi (\chi,x) = \int \mathrm{exp}[L(p,q)]d\psi
 ds^2 = e^{-2\pi T|\phi|}[\eta + \bar{h}_{\mu\nu}]dx^{\mu\nu}dx^{\mu\nu} +
 T^2 d^2\psi

    M_3 \bigotimes_{k=0}^{\infty} E^{+}_{-} = \mathrm{rot}
    (\mathrm{div} E, E_1)
    = m(x), {P^{2n} \over M_3} = H_3(M_1)

 \exists [R + |\nabla f|^2]^{{1 \over 2} + iy}
 = \int \mathrm{exp}[L(p,q)]d\psi
 = \exists [R + |\nabla f|^2]^{{1 \over 2} + iy} \otimes
 \int \mathrm{exp}[L(p,q)]d\psi +
N\mathrm{mod}(e^{x \log x})
 = \mathcal{O}(\psi)

 {d \over dt}g_{ij}(t) = - 2 R_{ij}, {P^{2n} \over M_3}
 = H_3(M_1), H_3(M_1) = \pi (\chi, x) \otimes h_{\mu\nu}
 S^{\mu\nu}_{m} \times S^{\mu\nu}_{n}
 = [D^2\psi] , S^{\mu\nu}_{m} \times S^{\mu\nu}_{n}
 = \mathrm{ker}f/\mathrm{im}f,  S^{\mu\nu}_{m} \otimes
 S^{\mu\nu}_{n} = m(x)[D^2\psi], {-{2R_{ij} \over V(\tau)}} = f^{-1}xf(x)
 f_z = \int \left[ \sqrt{\begin{pmatrix} x & y & z \\
   u & v & w \end{pmatrix} \circ
   \begin{pmatrix} x & y & z \\
   u & v & w \end{pmatrix}}_{}\right]dxdydz,
   \to f_z^{1 \over 2} \to (0,1)\cdot(0,1) = -1,i =
 \sqrt{-1}
{\begin{pmatrix} x,y,z
    \end{pmatrix}}^2 = (x,y,z)\cdot(x,y,z) \to - 1
```

```
 \mathcal{O}(x) = \nabla_{i}\nabla_{j} \int e^{{2 \over m}\sin \theta
 \cos \theta} \times {N \mathrm{mod}
(e^{x \log x})
\over \mathrm{O}(x)(x + \Delta |f|^2)^{1 \over 2}}
 x \Gamma(x) = 2 \int |\sin 2\theta|^2d\theta,
 \mathcal{O}(x) = m(x)[D^2\psi]

 \lim_{\theta \to 0}{1 \over \theta} \begin{pmatrix} \sin \theta \\
   \cos \theta \end{pmatrix}
   \begin{pmatrix} \theta & 1 \\
   1 & \theta \end{pmatrix}
   \begin{pmatrix} \cos \theta \\
   \sin \theta \end{pmatrix}
   =  \begin{pmatrix} 1 & 0 \\
   0 & - 1 \end{pmatrix},
f^{-1}(x) x f(x) = I^{’}_m, I^{’}_m = [1,0] \times [0,1]


 i^2 = (0,1) \cdot (0,1),|a||b|\cos \theta = -1,
 E = \mathrm{div}(E,E_1)
 \left({\{f,g\} \over [f,g]}\right)^{’} = i^2,  E = mc^2, I^{’} = i^2

 \mathcal{O}(x) = || \nabla \int [\nabla_{i}\nabla_{j} f
 \circ g(x)]^{{1 \over 2} + iy}|| , \partial r^n
||\nabla||^2 \to \nabla_{i}\nabla_{j} ||\vec{v}||^2

 \nabla^2 \phi

 \nabla^2 \phi = 8 \pi G \left({p \over c^3} + {V \over S}\right)

 (\log x^{1 \over 2})^{’} = {1 \over 2}{1 \over (x \log x)},
(\sin \theta)^{’} = \cos \theta, (f_z)^{’} = i e^{i x \log x},
{d \over df}F = m(x)

 {d \over df}\int \int{1 \over (x \log x)^2}dx_m
 + {d \over df}\int \int {1 \over (y \log y)^{1\over2}}dy_m
= {d \over df} \int \int \left({1 \over (x \log x)^2}
+ {1 \over (y \log y)^{1 \over 2}}\right)dm
 \ge {d \over df}\int \int \left({1 \over
 (x \log x)^2 \circ (y \log y)^{1 \over 2}}\right)dm
 \ge 2h

 {d \over df}\int \int \left({1 \over (x \log x)^2 \circ
 (y \log y)^{1 \over 2}}\right)dm  \ge \hbar
 y = x, xy = x^2, (\square \psi)^{’} = 8 \pi G
 \left({p \over c^3}\circ{V \over S}\right)
 \square \psi = \int \int \mathrm{exp}[8 \pi G(\bar{h}_{\mu\nu}
 \circ \eta_{\mu\nu})^{\mu\nu}]dmd\psi,
```

```
 \sum a_k x^k = {d \over df}\sum \sum {1 \over a^2_k f^k}dx_k
 \sum a_k f^k = {d \over df}\sum \sum
{\zeta(s) \over a_k}dx_{km},
 a^2_kf^{1 \over 2}\to \lim_{k \to 1}a_k f^k = \alpha

   ds^2 = [g_{\mu\nu}^2, dx]
   M_2
   ds^2 = g_{\mu\nu}^{-1}(g^2_{\mu\nu}(x) - dx g_{\mu\nu}^2)
  M_2
   = h(x) \otimes g_{\mu\nu}d^2x - h(x) \otimes dx g_{\mu\nu}(x),
  h(x) = (f^2(\vec{x}) - \vec{E}^{+})
   G_{\mu\nu} = R_{\mu\nu}T^{\mu\nu},
   \partial M_2 = \bigoplus \nabla C^{+}_{-}
    G_{\mu\nu}   equal   R_{\mu\nu}  {d \over dt}g_{ij} = - 2 R_{ij}
r = 2 f^{1 \over 2}(x)
     E^{+} = f^{-1}xf(x),
   h(x) \otimes g(\vec{x}) \cong {V \over S},
  {R \over M_2} = E^{+} - {\phi}
     = M_3 \supset R,
   M^{+}_2 = E^{+}_{1} \cup E^{+}_{2} \to E^{+}_1 \bigoplus E^{+}_2
     = M_1 \bigoplus \nabla C^{+}_{-}, (E^{+}_{1} \bigoplus E^{+}_{2})
     \cdot (R^{-} \subset C^{+})
     {R \over M_2} = E^{+} - \{\phi\}
     = M_3 \supset R
     M^{+}_3 \cong h(x) \cdot R^{+}_3
  = \bigoplus \nabla C^{+}_{-},
  R = E^{+} \bigoplus M_2 - (E^{+} \cap M_2)
     E^{+} = g_{\mu\nu}dxg_{\mu\nu},
   M_2 = g_{\mu\nu}d^2x,
   F = \rho g l \to {V \over S}
     \mathcal{O}(x) = \delta(x)[f(x) + g(\bar{x})] + \rho g l,
   F = {1 \over 2}mv^2 - {1 \over 2}kx^2,
   M_2 = P^{2n}
      r = 2f^{1 \over 2}(x),
  f(x) = {1 \over 4}\|r\|^2

     V = R^{+}\sum K_m, W = C^{+}\sum^{\infty}_{k=0} K_{n+2},
     V/W = R^{+}\sum K_m / C^{+}\sum K_{n+2}
     = R^{+}/C^{+} \sum{x^k \over a_k f^k(x)}
     = M^+_{-}, {d \over df} F = m(x), \to M^{+}_{-}, \sum^{\infty}_{k=0}
     {x^k \over a_k f^k(x)} = {a_k x^k \over
  \zeta(x)}

     {\{f,g\} \over [f,g]} = {fg + gf \over fg - gf},
  \nabla f = 2, \partial H_3 = 2, {1 + f \over 1 - f} = 1,
  {d \over df} F = \bigoplus \nabla C^{+}_{-}, \vec{F} =
  {1 \over 2}
     H_1 \cong H_3 = M_3

   H_3 \cong H_1 \to \pi(\chi,x), H_n, H_m =
   \mathrm{rank}(m,n), \mathrm{mesh}(\mathrm{rank}(m,n)) \lim \mathrm{mesh} \to 0
   (fg)' = fg' + gf', ({f \over g})' = {{f'g - g'f} \over g^2},
   {\{f,g\} \over [f,g]} = {(fg)' \otimes dx_{fg} \over
```

```
({f \over g})' \otimes g^{-2}dx_{fg}}
   = {{(fg)'\otimes dx_{fg}} \over ({f \over g})' \otimes g^{-2}dx_{fg}}
   = {d \over df} F


     \hbar\psi = {1 \over i} H \Psi, i[H,\psi] = - H \Psi, {\{f,g\} \over [f,g]} = (i)^2

     [\nabla_{i} \nabla_{j} f(x), \delta(x)] = \nabla_{i} \nabla_{j}
     \int f(x,y)dm_{xy}, f(x,y) = [f(x), h(x)] \times [g(x), h^{-1}(x)]
     \delta(x) = {1 \over f'(x)}, [H, \psi] = \Delta f(x),
     \mathcal{O}(x) = \nabla_{i} \nabla_{j} \int \delta(x)f(x)dx
     \mathcal{O}(x) = \int \delta(x)f(x)dx

     R^{+} \cap E^{+}_{-} \ni x, M \times R^{+} \ni M_3, Q \supset C^{+}_{-},
     Z \in Q \nabla f, f \cong \bigoplus_{k=0}^{n} \nabla C^{+}_{-}
     \bigoplus_{k=0}^{\infty} \nabla C^{+}_{-} = M_1, \bigoplus_{k=0}^{\infty}
     \nabla M^{+}_{-} \cong E^{+}_{-},
  M_3 \cong M_1 \bigoplus_{k=0}^{\infty} \nabla {V^{+}_{-} \over S}
     {P^{2n} \over M_2} \cong \bigoplus_{k=0}^{\infty}
     \nabla C^{+}_{-}, E^{+}_{-} \times R^{+}_{-} \cong M_2
     \zeta(x) = P^{2n} \times \sum_{k=0}^{\infty} a_k x^k,
     M_2 \cong P^{2n}/\mathrm{ker}f, \to \bigoplus \nabla C^{+}_{-}
     S^{+}_{-} \times V^{+}_{-} \cong {V \over S} \bigoplus_{k=0}^{\infty}
     \nabla C^{+}_{-}, V^{+} \cong M^{+}_{-} \bigotimes S^{+}_{-},
     Q \times M_1 \subset \bigoplus \nabla C^{+}_{-}
     \sum_{k=0}^{\infty} Z \otimes Q^{+}_{-} = \bigotimes_{k=0}^{\infty} \nabla M_1
     = \bigotimes_{k=0}^{\infty} \nabla C^{+}_{-} \times
     \sum_{k=0}^{\infty} M_1, x \in R^{+} \times C^{+}_{-}
  \supset M_1, M_1 \subset M_2 \subset M_3

 S^3, H^1 \times E^1, E^1, S^1 \times E^1, S^2 \times E^1, H^1 \times S^1,
  H^1, S^2 \times E$.
 \bigoplus \nabla C^{+}_{-} \cong M_3, R \supset Q, R \cap Q,
 R \subset M_3, C^{+} \bigoplus M_n, E^{+} \cap R^{+}$
  M^{+}_{-}\nabla C^{+}_{-}, C^{+}\nabla H_m, E^{+} \nabla R^{+}_{-}, E_2 \nabla E_1 $
  $ R^{-} \nabla C^{+}_{-} $.     {\nabla \over \Delta} \int x f(x) dx,
  {\nabla R \over \Delta f}, \square = 2{\int {(R + \nabla_{i} \nabla_{j} f)^2
  \over -(R + \Delta f)}}e^{-f}dV
     \square = {\nabla R \over \Delta f}, {d \over dt}g_{ij}
     = \square \to {\nabla f \over \Delta x}, (R +
  |\nabla f|^2)dm \to -2(R + \nabla_{i} \nabla_{j} f)^2 e^{-f}dV
     x^n + y^n = z^n \to \nabla \psi^2 = 8 \pi G T^{\mu\nu},
     f(x + y) \ge f(x) \circ f(y)
     \mathrm{im}f / \mathrm{ker}f = \partial f, \mathrm{ker}f
     = \partial f, \mathrm{ker}f / \mathrm{im}f \cong
  \partial f, \mathrm{ker}f = f^{-1}(x)xf(x)
     f^{-1}(x)xf(x) = \int \partial f(x) d(\mathrm{kerf}) \to \nabla f = 2
     _{n}C_{r} = {}_{n}C_{n-r}  \to \mathrm{im}f / \mathrm{ker}f
     \cong \mathrm{ker}f / \mathrm{im}f



   $ \sum^{\infty}_{k=0}a_k f^k = T^2d^2 \phi $. this equation $ a_k \cong
```

```
   \sum^{\infty}_{r=0} {}_n C_r $.
     V/W = R/C \sum^{\infty}_{k=0}{x^k \over a_k f^k}, W/V = C/R
     \sum^{\infty}_{k=0}{a_k f^k \over x^k}
     V/W \cong W/V \cong R/C(\sum^{\infty}_{r=0} {}_nC_{r})^{-1}
     \sum^{\infty}_{k=0} x^k
  This equation is diffrential equation, then $ \sum^{\infty}_{k=0} a_k f^k $
  is included with $ a_k \cong \sum^{\infty}_{r=0} {}_nC_{r} $
     W/V = xF(x), \chi(x) = (-1)^k a_k, \Gamma(x) = \int e^{-x} x^{1 -t}dx,
     \sum^{n}_{k=0}a_k f^k = (f^k)'
     \sum^{n}_{k=0}a_k f^k = \sum^{\infty}_{k~0} {}_{n}C_{r} f^k
     = (f^k)',
   \sum^{\infty}_{k=0} a_k f^k = [f(x)],
   \sum^{\infty}_{k=0} a_k f^k = \alpha, \sum^{\infty}_{k=0}
   {1 \over a_k f^k}, \sum^{\infty}_{k=0} (a_k f^k)^{-1} = {1 \over 1 - z}

      {\int \int {1 \over (x \log x)(y \log y)}dxy} =
      {{{}_nC_{r} xy} \over {({}_nC_{n-r}
   (x \log x)(y \log y))^{-1}}}
      = ({}_nC_{n-r})^2 \sum_{k= 0}^{\infty}({1 \over x \log x}
      - {1 \over y \log y})d{1 \over nxy} \times {xy}
      = \sum_{k=0}^{\infty} a_k f^k
      = \alpha
}


 _ struct_ :asperal equation.emerged => [tuplespace]
tuplespace.cognitive_system => development -> Omega.Database[import]
value.equation_emerged.exclude >- Omega.Database[tuplespace]


Omega::DataBase <-> virtual_connect(VIRTUALMACHINE)
{
  blidge_base.network => localmachine.attachment
  :=> {
       dhcp.etc_load_file(this.klass) {|list|
        list.connect[XWin.display _ <- xhost.in(regexpt.pattern)]
        {
          ultranetwork.def _struct {
           asperal_language :this.network_address.included[type.system_pattern]
            {|regexpt.pattern|
              <- w.scan
                    |each_string| <= { ipv4.file :file.port
                                       subnetmask :file.address
                                                     file.port <=> file.address
                                       FILE *pointer
                                       int,char,str :emerge.exclude > array[]
                                       BTE.each_string <-> regexpt.pattern
                                        {
                                          development => file.to_excluded
                                            file.scan => regexpt.pattern
                                              this.iterator <-> each_string
                                               file.reloded => [asperal_language.rebuild]
                                        }
```

```
                                    }
           }
         }
       }
      }
     }
}


class Ultranetwork
 def virtual_connect
  load :file => {
   asperal :virtual_machine.attachment
   {
     system.require :file.attachment
     <- |list.file| :=> {
         tk.mainloop <- [XWin -multiwindow]
         startx => file.load.environment
           in { [blidge_base | host_base].connect(wmware.dhcp)
                net_work.connect.used[wireshark.demand => exclude(file)]
          }
     }
   }
  }
 end


 def < blidge_base.network.connect
   {
    dhcp.start => {
                   host_name <-> localhost_name {|list|
                     list.exist(connect_type)
                      {
                        <- : tty :xhost -display => list.exist
                             [virtual_connect].list->host :terminal
                      }
                   }
   }
 end


 def < host_base.ethernet.connect
   {

                   host_name.connect => local_network
                  }
   }
 end




 def < etc.load_file
```

```
   {
    etc.include(inetd.rc)
     {
       virtual_connect(VIRTUAL_MACHINE){|list|
        list.attachment(etc.load_file)
       }
     }
   }
 end

 mainloop{
  def.virtual_connect => xhost.localmachine
  {
   xhost.client <-> xhost.server
  }
  def.network.type <- [Omega.DATABASE] end
  def.etc.load_file.attachment(VIRTUAL_MACHINE) end
 end
end


class UltraNetwork::DATABASE import OMEGA.TUPLESPACE
  def load_file >- VIRTUAL_MACHINE
   { in . => attachment_device |for|
   for.load -> acceptance.hardware
   virtual_machine.new
    {
    tk.loop-> start
    XWin -multiwindow
    if dwm <-> new_xwin.start
    localhost :xhost :display -x
    xdisplay :-> [preset :XFree.demand>=needed
    for.set_up
    install_process >- tar -xvfz "#{load_file}" <-> install_attachment
    ]
    else if
    only :new_xwin.start
    localhost :xhost :multiwindow . { in
    display -x
    attachment :localhost -client
    from -client into
    server.XWin -attachment}
    end
    condition :{ in .=>
    check->[xdisplay.install_process]}
  end

  def < network_rout
          wireshark.start -> ethernet.device >- define rout
                rout.ipstate do |file|
                   file.type <- encoding XWin -filesystem
                   file.included >- make kernel_system.rebuild
                   file.vmware.start do |rout|
```

```
                 rout.blidgebase | rout.hostbase
         -> file.install
            file.address_ipstate
            => {"{file}" :=> dwm.state_presense
            virtual_machine.included[file]
      }
  end

  def < launcher_application
        network_rout.new
        |file|
        file.attachment => { in .
        new_xwin.start :=> file.included
        demand.file <- success_exit}
  end

  def < terminal_port
        network_rout.new
        launcher_application.new |rout|
        rout.acceptance {
        vmware.state.process |new_rout|
        new_rout : attachment.class <-> dwm.state_attachment
        new_rout -> condition.start_wmware.process}
  end


  def < kterm_port
         launcher_application.new
         def.included[DATABASE]
         |rout|
         rout.attachment <- |new_rout|
         new_rout.attachment do
         install.condition < rout.def.terminal_port.exclude[file]
  end

  main_loop :file do
            kterm_port.excluded :=> VIRTUAL_MACHINE
            |new_rout| start do
            rout.process -> network_rout.rout [
            file,launcher_application, terminal_port, kterm_port].def < included
            |file|
            file.all_attachment: file_type :=> encoding-utf8
  end
end



class < def {
      pholograph_data[] = [R,V,S,E,U,M_n,Z_n,Q,C,N,f,g]
      source_array <- pholograph_data[]
}

def > operator_data[] = {nabla,nabla_i nabla_j,Delta,partial,
```

```
                          d, int, cap,cup,ni,in,chi,oplus,otimes,bigoplus,bigotimes,d /over df,
                          dV,dm,dx,dy,<,>,[,],{,},|,| }
end

def > manifold_emerge

         c = def.inject >- source_array times def.operator_data[]
 repository_data <=> c{

 c.scan(/tupplespace[]/)
 import |list| list{
    kerf = -2 \int (R + nabla_i nabla_j f)^2e^{-f}dV
    kerf / imf
    =< {d \over df}F}
     }
         equals_data =~ /list/
             list.match(/"#{c}"/) {|list|
             list.delete
             jisyo_data_mathmatics <=> list{
            list.emerge => {asperal function >- pholograph_data[] times repository_data
                   =< list.update}
            }
                   ln -s operator_named <= {list}
                    define _struct |list|
                          -> list.element -> manifold_emerge
                          =>  list.reconstruct > def.inject /^"#{pattern}"/}
end


import Omega::Tuplespace < Database
{
  {\bigoplus \nabla M^{+}_{-}}.equation_create -> asperal :variable[array]
   :=> [cognitive_system <-> def < VIRTUALMACHINE.terminal
                                   {
                                     [ipv4.bloadcast.address :
                                       ipv4.network.adress].subnetmask
                                      <-> file.port.transport_import :
                                              Omega[tuplespace]
                                   }
}

_struct _ Omega[tuplespace] >> VIRTUALMACHINE.terminal.value


class < def.VIRTUALMACHINE.system_environment



             file.reload[hardware] => file.exclude >> file.attachment
             {=>
                |file|
                  file.port(wireshark.rout <-> {file.port.transport_export
                  :=> Omega[tuplespace]}
```

```
                        assembly_process.file.included >- file.reloaded
                             :- |file.environment| {=>
                                            file.type? :=> exist
                                              file.regexpt.pattern[scan.flex]
                                                   => |pattern|
                                                         <->
                                                           file.[scan.compiler]
                                  }
                         end
                 end
               file <<
              }
}

Omega::Database[tuplespace]
{
  cognitive_system |: -> { DATABASE.create.regexpt_pattern >-
     cognitive_system[tuplespace].recreated >- : =< DATABASE.value
      >> system_require.application.reloaded[tuplespace]
         } : _struct _ def.VIRTUALMACHINE.terminal >> {
             ||machine.attachment|| <-> OBJECT.shift => system.reloaded
             . in {
                     : _struct _ class.import :-> require mechanics.DATABASE
                        {|regexpt_pattern| :|-> aspective _union _
                         def _union _}
                  }
             }

   end

}

system.require <- import library.DATABASE
{
  Omega[tuplespace]
  {
       cognitive_system : VIRTUALMACHINE.equality_realized
       {|regexpt_pattern| => value | key [ > cognitive_system.loop.stdout]
            value : display -bash :xhost -number XWin.terminal
            key   : registry.edit :=> {[cognitive_system.reloaded]}
       }
  }
}


_union _ => DATABASE[tuplespace].aspective_reloaded
_union _ :fx | -> |regexpt_pattern| => {
                     VIRTUALMACHIE.recreated-> _union _ |
                     _struct _ def.DATABASE.recreated <- fx
                  >> DATABASE[tuplespace].rebuild
}

DATABASE[tuplespace] -< {[ > aimed.compiler | aimed.interpreter] | btree.def.distributed >-
```

```
                         aimed[tuplespace]}
aimed[tuplespace] -< btree.class.hyperrout_ struct _ => Omega::Database[tuplespace].value
  sheap_ union _ :aspective | -> Omega[tuplespace]: | aimed[tuplespace].differented_review
}

aimed[tuplespace].process => DATABASE[tuplespace].reloaded
aimed.different | aimed.stdout >> vale | key [ > cognitive_system.loop.stdin] {|pattern|
                                pattern.scan(value : aimed[def.value]
                                   key   : aimed[def.key])
                } _ struct _ : flex | interpreter.system
                   => expression.iterator[def.first,def.second,def.third,def.fourth]
                      { def < Omega[tuplespace]
                        def.cognitive_system |: -> DATABASE[tuplespace] | aimed[tuplespace]
}

Omega::Tuplespace < DATABASE
{=>
   norm[Fx] -> . in for def.all_included < aimed[tuplespace].each_scan([regexpt_pattern]
   <->
                   DATABASE[tuplespace]) << streem database.excluded
   >- more_pattern.scan(value : aimed[def.value]

   key :aimed[def.key])
               . in { _struct _ :flex | interpreter.system
                   => expression.iterator[def.all.each -> |value, key|
                                  included >- norm[Fx]|[DATABASE[tuplespace]
  ,aimed[tupespace]] |
                                   finality : aimed[tuplespace], DATABASE[tuplespace]

   : -> def.included(in_all)
                                   {
                                       def.key | def,value => [DATABASE].recompile
       & make install
                                    : in_all -> _struct _ :aspective :tuplespace
    : all_homology_created}
                    }
}

def < Omega::Tuplespace[DATABASE]
 def.iterator -> |klass,define_method,constant,variable,infinity_data : -> finite_data|
         def.each_klass?{|value, key|
            _struct _ :aspective -> tuplespace :all_homology_recreated :make menuconfig
            {=+
               def.key -> aimed[def.key],def.value -> aimed[def.value] {|list|
                   list.developed => <key,value> | <aimed[$‘,$’]
                    -> _union _ :value,key : _struct _
                    <- (_union _ <-> _struct _ +)
               begin
                  def.key <-> aimed[value]
                  case :one_ exist :other :bug
                  {
                     result <-> def.key
                     {
```

```
                          differented :DATABASE[tuplespace]
                        }
                        return :tuplespace.value.shift -> included<tuplespace>
                     else if
                     :other :bug
                     {
                       success_exit <- bug[value]
                       {
                         cognitive_system.scan(bug[value])
                         {
                          {[e^{-f}][{2 \int (R + \nablaf^2) \over -(R + \Delta f)}e^{-f}dV}
       .created_field
                            {=>
                                regexpt.pattern \native_function <-> euler-equation
                                 {
                                    $variable =< diff e^{-f} >- $'
                                      all_included <- def.key <-> aimed[value]
                                      $variable - all_included.diff
                                  \summuate_manifold.recreated
       <- \native_function : euler-equation
                                 }
                            }
                          }
                       } _union _ :cognitive_system.rebuild(one_ exist)
                    }
                   }
                   ensure
                   {
                       return :success_exit
                       => Tuplespace[DATABASE]
                   }
                  }
                }
         end
 end
}

 int
streem_style {
  :Endire <- [ADD,EVEN,MOD,DEL,MIX,INCLUDED,EXCLUDED,EBN,EXN,EOR,EXOR,
              SUM,INT,DIFF,PARTIAL,ROUND,HOMOLOGY,MESH]
  Endire.interator -> {def < :Endire.element, -> def.means_each{x -> expression.define.included
  def.each{x -> case :x.each => :lex.include_ . in [ > [x.all_expire] ]}

}

main_loop {
  FILE *fp :=> streem_style.address_objective_space
  fp.each{x -> domain_specific_language_style_included[array]}
  array << streem.DATABASE[tuplespace]
  array.each{[tuplespace] -> aimed[tuplespace] | OMEGA_DATABASE[tuplespace]}.excluded <-> array
  def.key <-> def.value => {x -> stdin | stdout |=> streem_style <- def.each.klass.value}
}
```

```
@reviser : def < OmegaDatabase[tuplespace].mechanism
{
  aspective : _union _ {
       int streem_style : [ > [def.each{x -> stdin | stdout > display :xhost in XWin -multiwind
       {
         Endire <- [ADD,EVEN,ODE,EXOR,XOR,DEL,DIFF,PARTIAL,INT].included > struct _ :-> _union
         Endire.each{def.value -> def.key :hash.define}.included > _union}
        }
}

@reviser : def.reconstructed.each{_union <-> _struct _.recreated : [def.del - def.before_determ



import perl.lib | python.lib <-> ruby.lib
{
  int @reviser : def.each{x -> x.klass |-> $variable in $stdin | $stdout}.developed >= {
                          ping localhost -> blidgebase <-> hostbase.virtualmachine.attachment
                          {
                              xhost :display -> streem_style.value
                              networkconnect.hostbase -> localarea.virtualmachine
                          } :connected -> networkrout : flow_to :localhost.attachment
  }
}_struct : def < hostbase.virtualmachine.attachment => : networkrout.area.build

@reviser <-> def.add [ < _struct]
@reviser : def.each{listmenu -> listlink | unlinklist > [developed -> {def.key , def.value}.cur
@reviser <-> def.rebuild [ < _struct]

@reviser.def.<value|key>networkrout-> def.present
def.present.flow_to -> hostbase.rout << networkrout.data.<value|key>

XWin -multiwindow <-> networkrout.data[$‘,$’]
def < $’
@reviser <-> def.present.state
@reviser.def.each{x | -> key.rebuild | value.rebuild}.flow_to :redefined









def < OmegaDatabase[tuplespace]
{
  FILE *fp -> cmd.value : cmd.key {fp |-> syncronized.file[tuplespace] | aimed.file[tuplespace]
```

```
  cmd.key => [ > fp.($‘:$’)] <-> registry.excluded<fp.file[cmd.state]>
}

def.each{fp|-> def.first,def.second,def.third,def.fourth}

cmd _struct : {
[ ^C-O : ^C-X-F, exit.cmd : ^C-X-C, shift-up : ^C-P, shift-down : ^C-N]
}

cmd _union : def.restructed
keyhook.cmd <- : [_struct ]
{
  @reviser :def._struct <-> def. _union
```