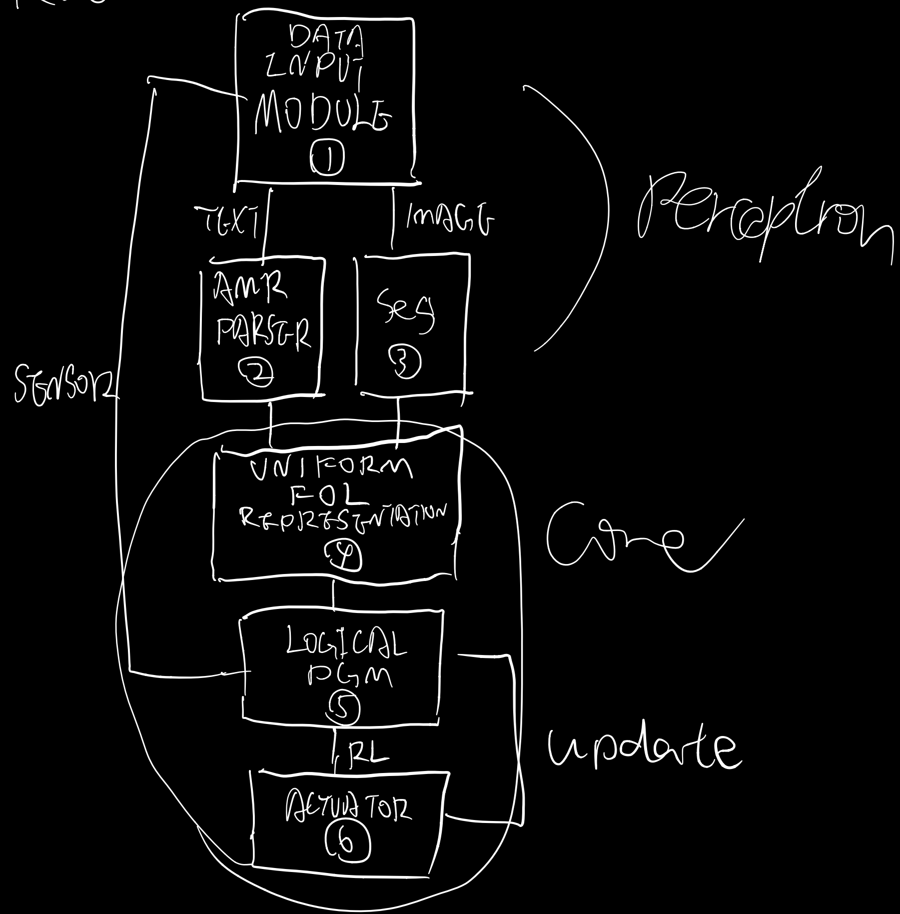
Calculus

I STRUCTRUE



Architecture Page 3

II Perceptron

DATA TYPE

1. SEQUENTIAL DATA

eg. TEXTS ref: AMR BiLSTM

2. MATRIX DATA

eg. IMAGES ref: Mask* RCNN

3. SENSOR DATA

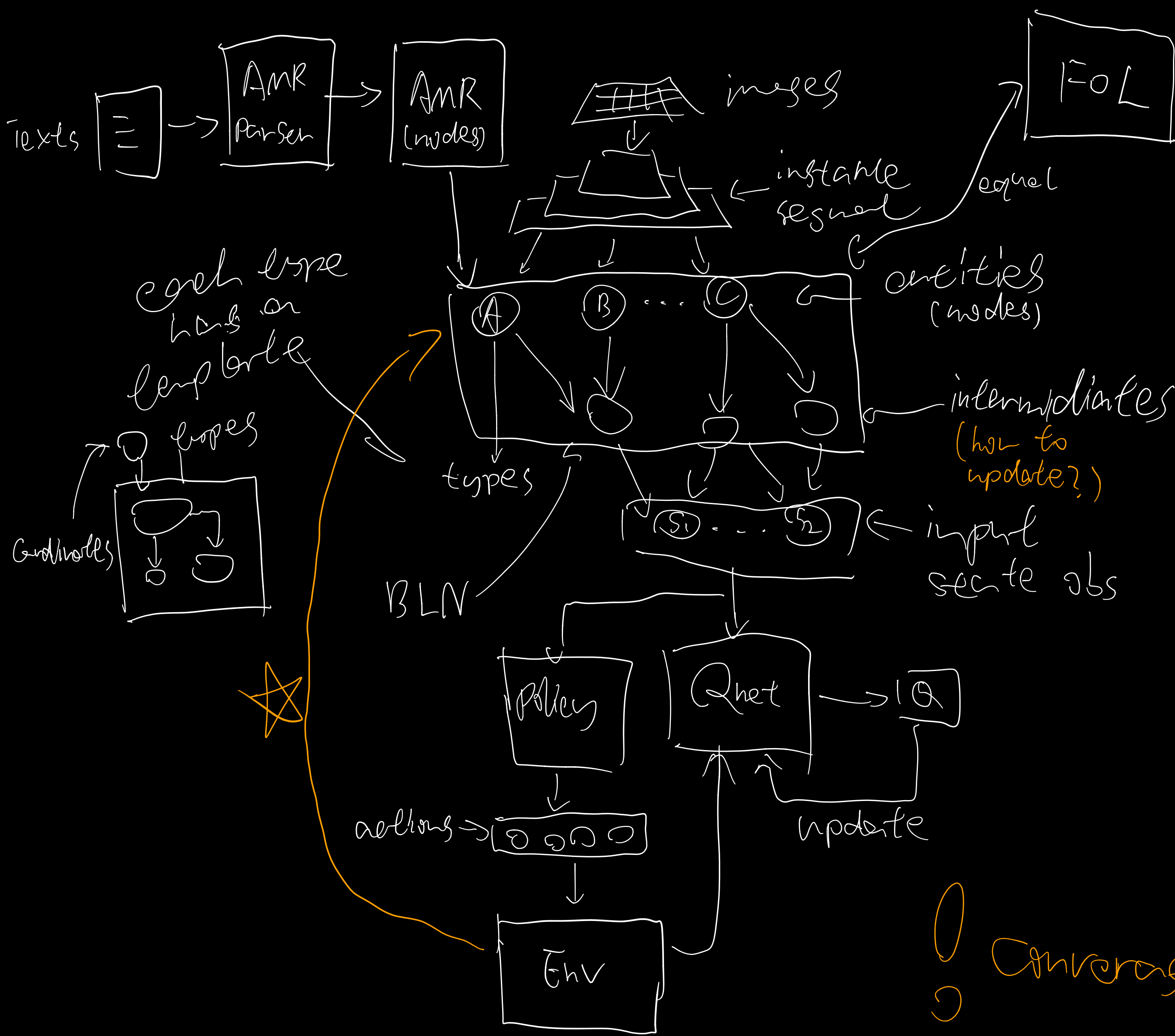
e.g. LIDAR   directly input

4. HUMAN ORDERS

e.g. control queue, change structure

☆ III Reasoning & Planning

ARCHITECTRUE



! Converge?

☆ Update BLN according to the Reward
1. Find reason of a reward
2. abductive based on axioms

OVERVIEW

BLN predicate: A, B

entity: a, b, c

$a, b \in A \quad c \in B$

e.g. a, b, c are instances seperated

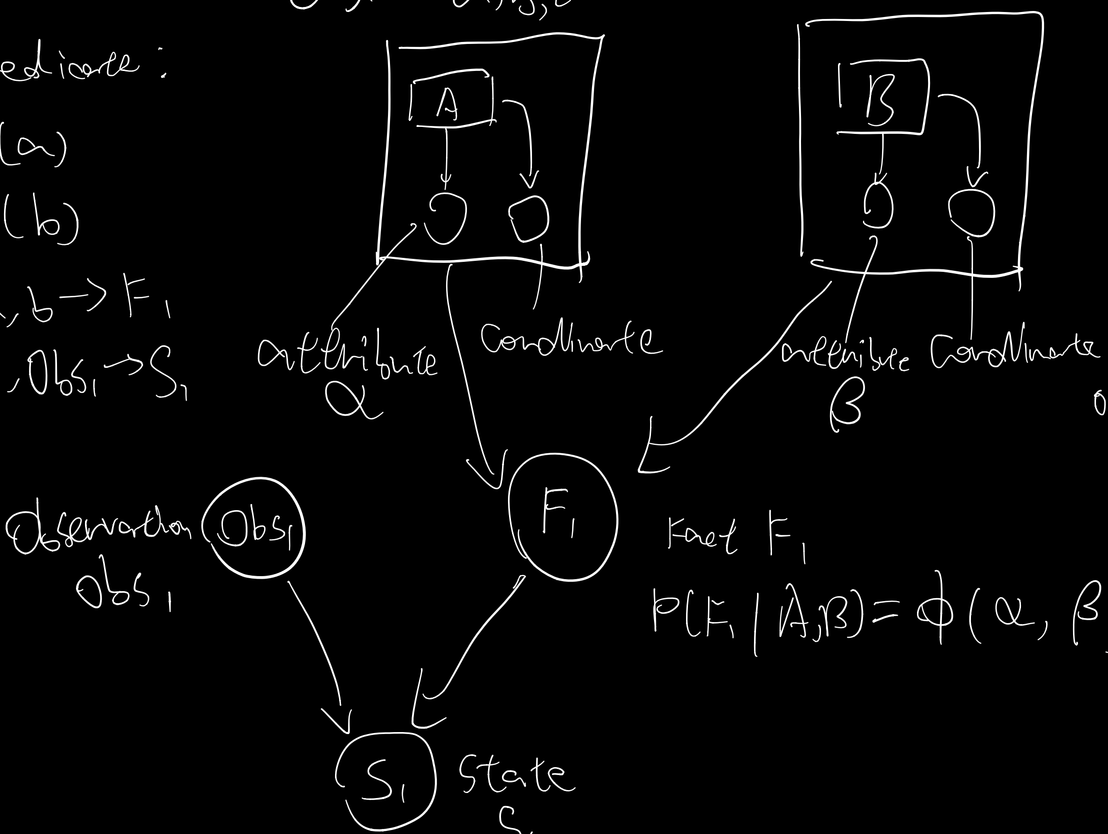
predicate:

$A(a)$

$B(b)$

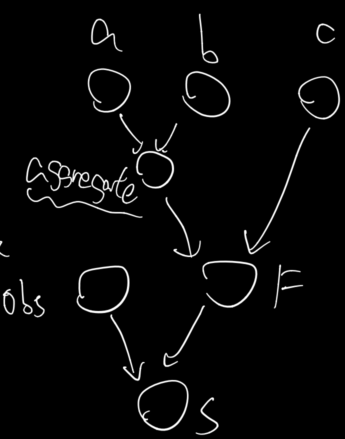
$a, b \to F_1$

$F_1, Obs_1 \to S_1$



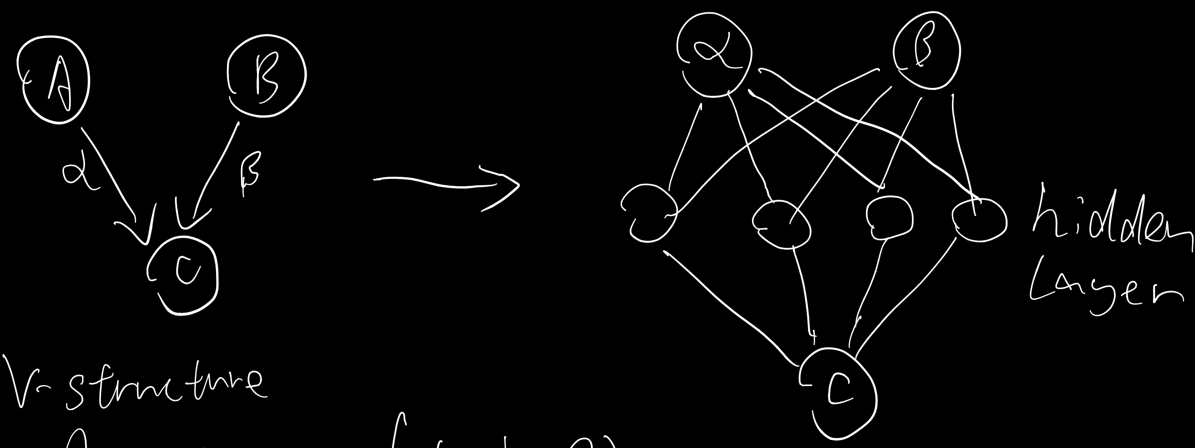
$P(F_1 \mid A, B) = \Phi(\alpha, \beta)$

$P(S_1 \mid Obs_1, F_1) = \Phi(Obs_1, F_1)$



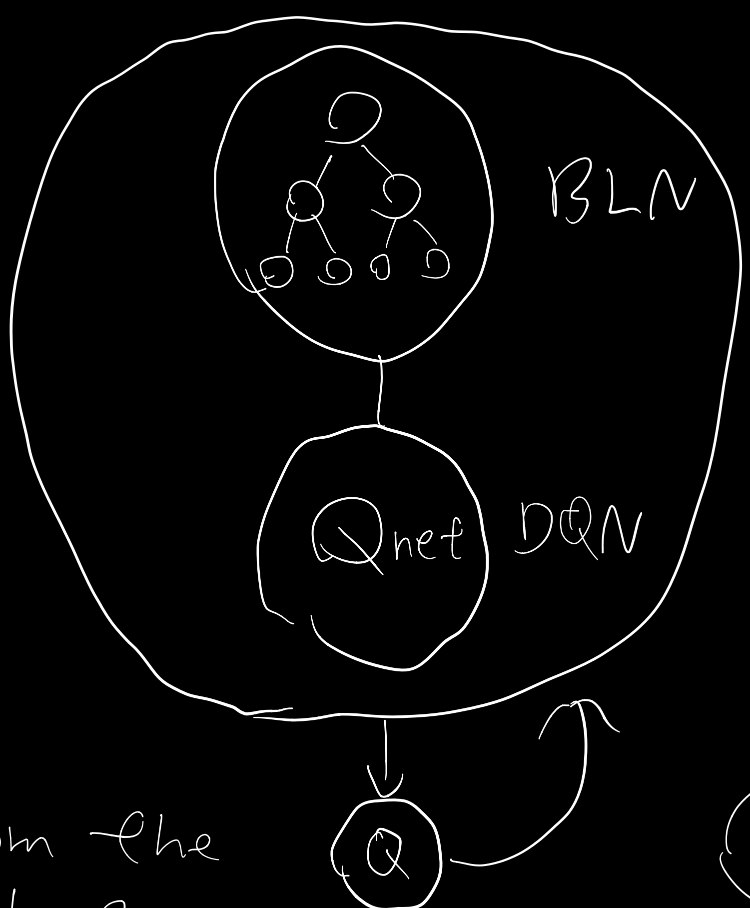
! Using Neural Network to fit Φ ?

$$Q_{n+1} = R_{S_n} + \gamma \int_a \Phi(a, S_{n+1}) R_{S_{n+1}}$$



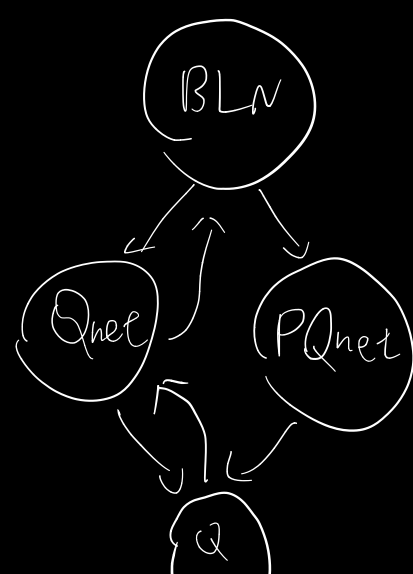
V-structure

$$\Phi(C \mid A, B) = f(\alpha, \beta)$$



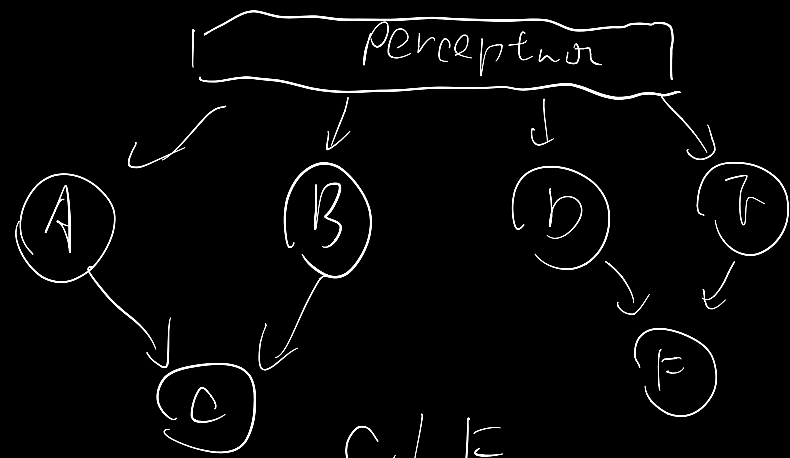
BP from the root of BLN to output of Qnet

Or



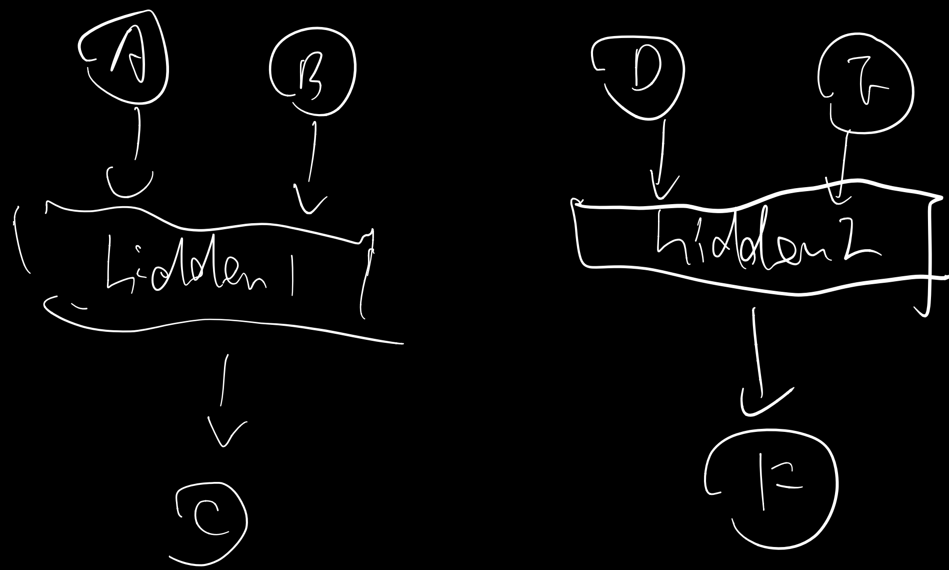
PROVEMENT

e.g.



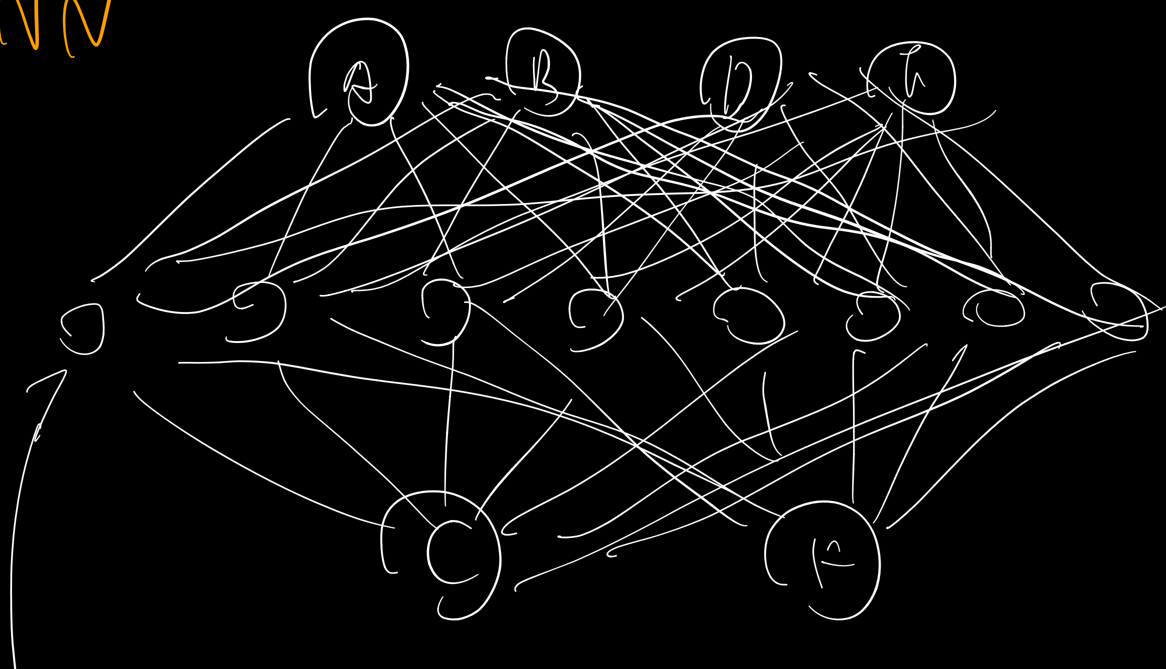
$C \perp F$

in BLN



C and F are fitted seperately

in NN



Let the values of nodes in hidden layer as h_i

$\therefore h_n = f_n(w_n A, w_n B, w_n C, w_n D)$

$H = \begin{bmatrix} h_1 \\ \vdots \\ h_n \end{bmatrix}$

$\therefore C = w_1 h_1 + \cdots + w_n h_n = W_C H_C$

$F = w_1' h_1 + \cdots + w_n' h_n = W_F H_F$

$\because H_C = H_F$

$\therefore H_C$ and H_F are linear correspondent

$\therefore C \not\perp F \Leftrightarrow C \perp F$

$\therefore$ NN lost the information about the independence between output values in this case

DATA STRUCTRUE

Entity Frag

es Template

Reason Frag

es m Frag

ENTITY

ATTRIBUTE

REASON

1 ROOT NODES OF ENTITY FRAG
ONLY HAVE OUT-EDGE

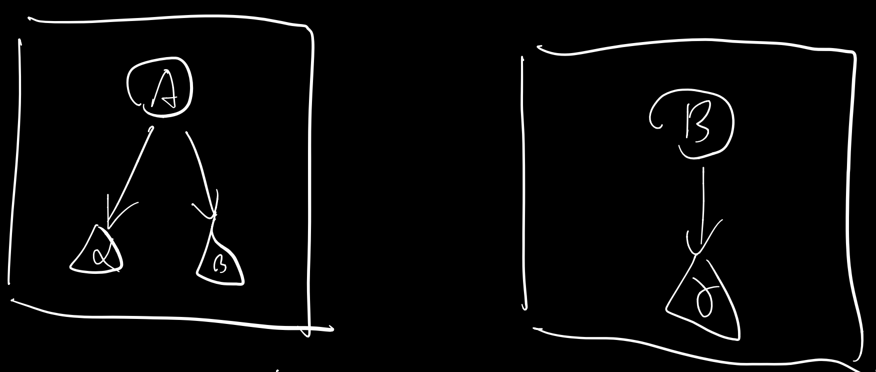
2 LEAVE NODES OF REASON FRAG
ONLY HAVE IN LINE

3 EACH ENTITY FRAG
ONLY HAS ONE ROOT

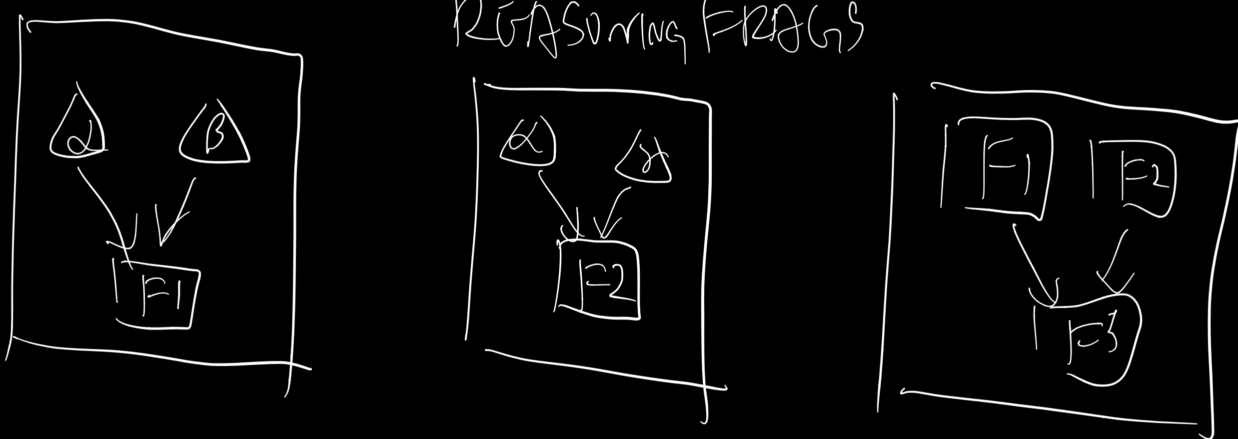
4 EACH REASON FRAG
ONLY HAS ONE LEAVE

ENTITY FRAGS

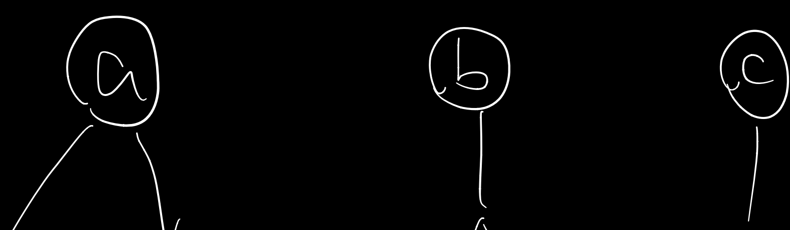
e.g.



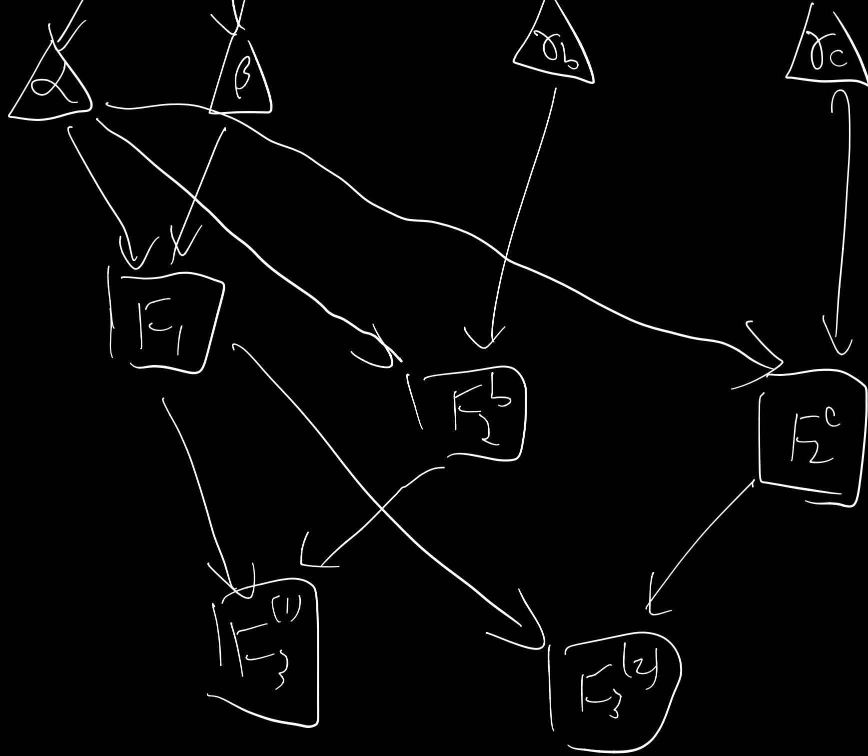
REASONING FRAGS



$a \in A \quad b \in B \quad c \in B$



instantiation



GOAL IN DATA STRUCTURE

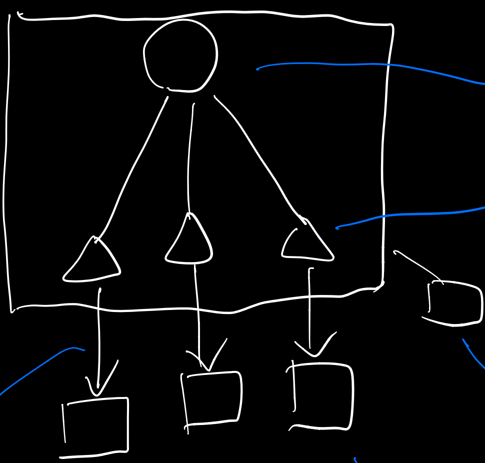
1. NO AMBIGUITY

2. FULLY EXPRESS FOL

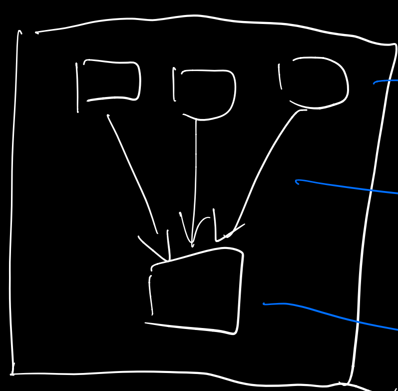
INSTANTIATION

OPTION I STATIC

EFrag

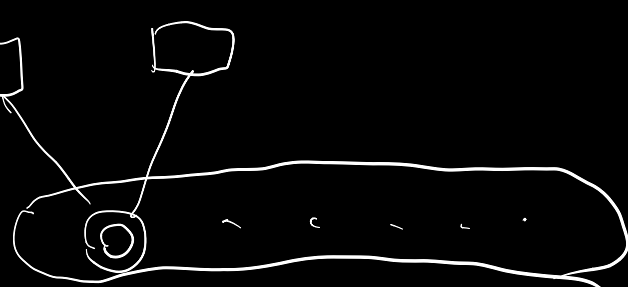


RFrag

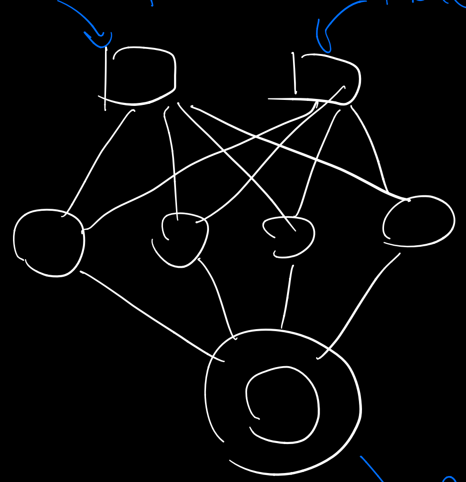


OPTION II DYNAMIC

input state



Fact A Fact B



Fact A has m copies

Fact B has n copies

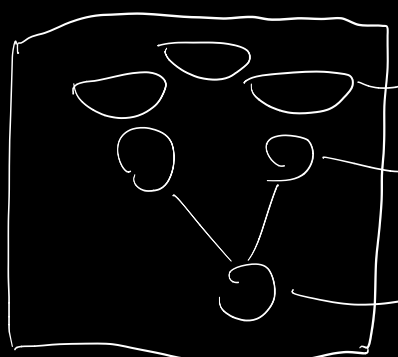
output states

Sum up output values of the combinations of all copies

e.g. output of fc layer $y = \sum_i w x_i$

x_i means activate values in one copy

FROM MFRAG TO GFRAG AND RFRAG



MFrag

Context — Condition of instantiating a template

Input

Resident

Output

For GFrag

the context means whether an entity is observed for RFres the context could be multi conditions

ADDITIONAL RULES

The output values of the template not instantiated are zero

Only the templates that are instantiated will be updated

REASONING

INPUT STATUS OF QNET

☆ How to aggregate multiple results from the entities of the same type?

1. entities of same type are IID
2. Guarantee the output node fixed

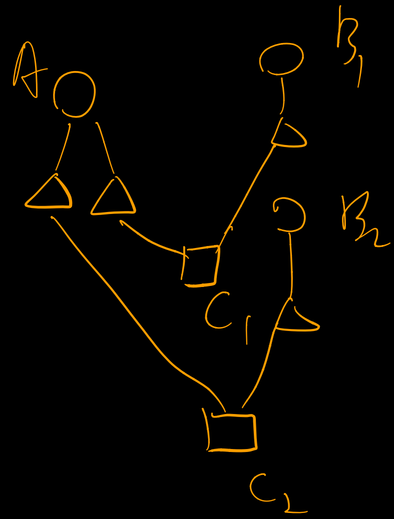
POTENTIAL SOLUTION

1 static network

2 dynamic network

TIME COMPLEXITY

for dynamic network



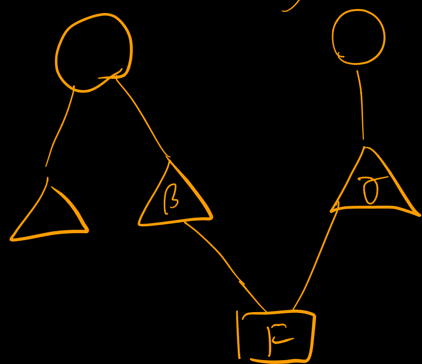
$N_A = 1$

$N_B = 2$

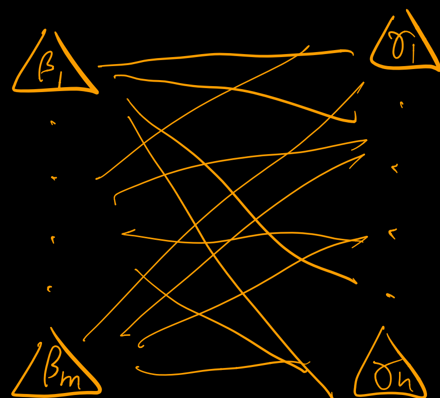
$C = 2$

① Computational amount for all combinations are equal

instantiation



Combinations between copies



num of copies of A

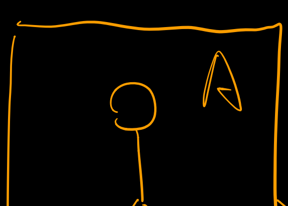
$N_A = m$ $N_{B_2} = n$

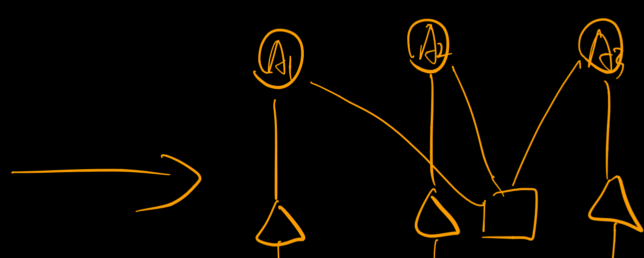
Num of Types T

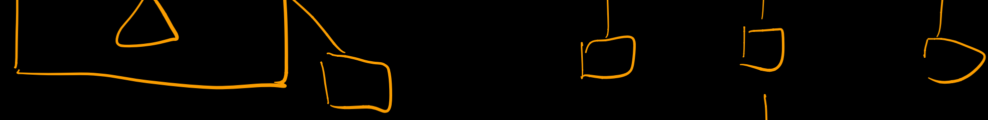
$C = mn$

Num of Combinations $C = \prod_{t \in T} N_t$

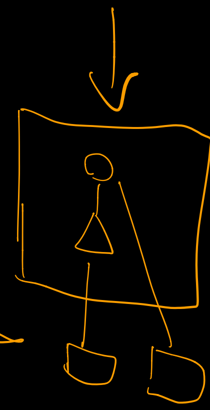
for static network







Num of Combination = 1



OUTPUT

☆ Aggregate Rule
(Need Human Design)

① less computation amount
② but need good design

Max, Min, Average, Random, Sum
Grouping, fuzzy function ...

☆ The input state of Qnet should be fixed

REALIZATION

Tags

↓ Find GFrag ← Hash of Tags

GFrag

↓

Instantiation

↓

Add Attributes to Fact List

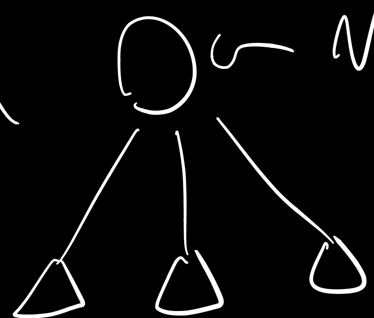
↓ check the list

Instantiating the RFrags

↓

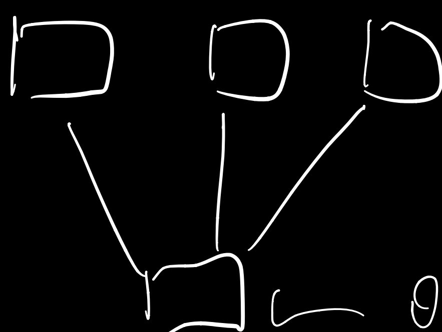
Add New Fact to List

until all facts checked



Name

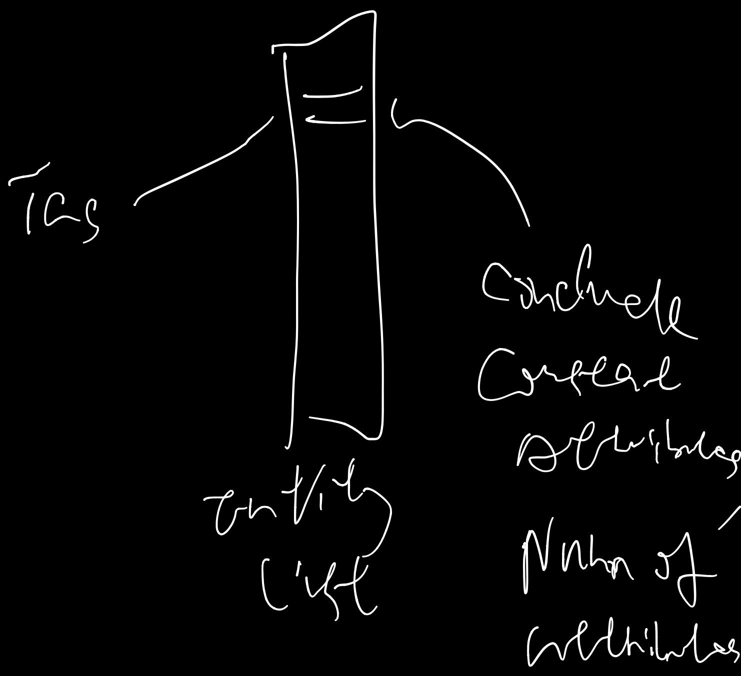
Attribute List

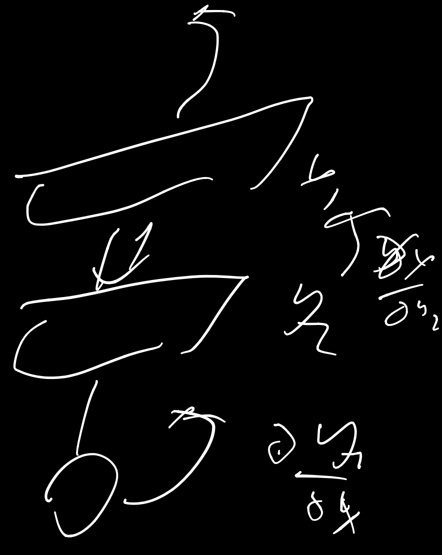


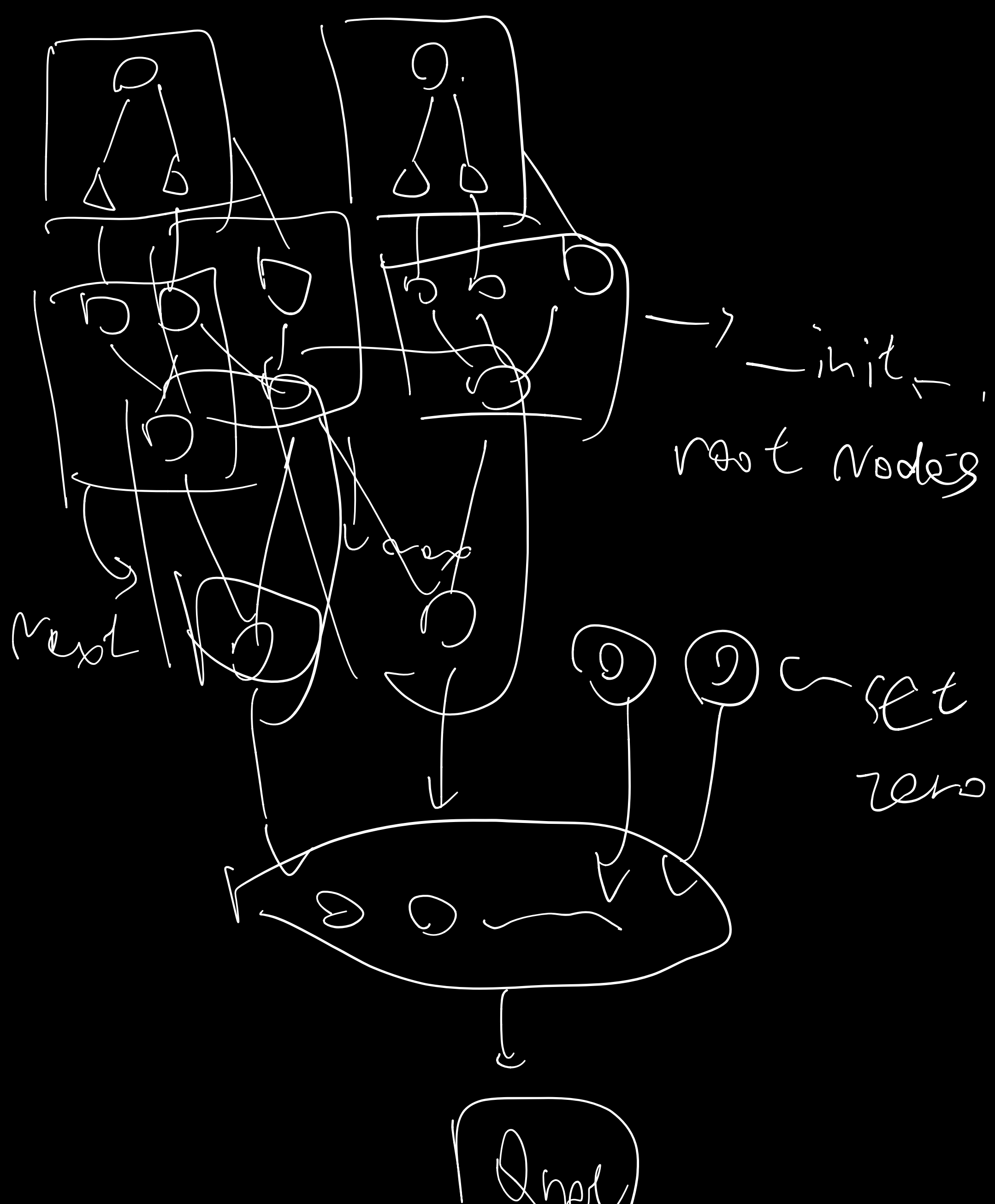
input Fact

Output Fact

whether instantiate only depend on Context


Tags
Entity List
Concrete
Compact
attributes
Number of
attributes


$\frac{\partial y}{\partial x}$


__init__,
root nodes
loop
next
set
zero
Queue