CYBER RESILIENCE, TRANSPARENCY AND MONOTONICITY UNDER ZERO-TRUST

Hideto Tomabechi drtomabechi@me.com
George Mason University* and Carnegie Mellon University**
*Visiting Professor, C4I and Cyber Center **Adjunct Fellow, CyLab

GEORGE MASON UNIVERSITY

Cognitive Research Laboratories

Carnegie Mellon University
Security and Privacy Institute
CyLab

My brief background in Computer Science areas

1985 - 1987
Yale CS, AI Lab and CogSci Proj. AI and Massively Parallel Processing

1987 - 1993
SCS, RI, CMT, LCL, Carnegie Mellon. AI, NLP, Computational Linguistics

1993 - 1998
Tokushima Univ. and Justsystems. Monotonicity and Intelligence

1998 – 2014
Japanese government projects, AI, Architectures, Cyber Security

2008 - current
Adjunct Fellow, CyLab, Carnegie Mellon University

2014 - 2019/3
Independent Advisor to Chief of Joint Staff, Admiral Kawano, JSDF
and liaison for CMU

Introduction of Concepts:

Cyber
- You are experts. I spare explanations.

Resilience
- I spare explanations. Please ask Dr. Wells

Transparency and Monotonicity
- Basic paradigm under zero-trust and quantum.

Transparency - from Crypt-Currency experiences

Nontransparent Off-Chain Exchanges
- Chinese and Japanese Exchanges

Cryptocurrency stolen
from Japanese off-chain exchanges in 2018
・500 mil USD equivalent of NEM stolen
from Coincheck (UN says by North Koreans)
・50 mil USD equivalent of Bitcoin stolen from Zaif

Transparency vs Non-transparency in Crypto Blockchain

Crypt is a misunderstood word. "Thou shalt not encrypt".
- We hash, not encrypt
- Private vs Public
- Sovereign vs Public

Current resilience of blockchain comes from transparency

What is Monotonicity

Amount of information only increases never decreases

Nakamoto type Blockchain is pseudo monotonic
(using hash and digital signature)

Purely monotonic linear time algorithms exist
(e.g. Tomabechi 90&91)

Why is Monotonicity Essential for Resilient Architecture

Like accounting books, modifications are always added.
No deletion allowed.

Naturally, traces of all modifications are preserved
at data structure level

All modifications are non-destructive
and original data is recovered constant/linear time

Some of my Monotonicity Projects

Old

- mid 90's Bechi Unit at JUSTSYSTEMS.

Mid

- 1998 -2004 Japanese government projects for "Bullet Proof" servers

New

- Crypto Asset Central Bank Foundation, advising some governments

Monotonicity and Layered Transparency

Basic data structure for quantum proof
next generation computer architecture

Linear time monotonic algorithms
and embedded layered access models

Different levels of exposure,
e.g. to law enforcement, medical, tax agency, etc.

Still fastest -- Linear-Time Monotonic Algorithms (Tomabechi Algorithms)

Quasi-Destructive Graph Unification (Tomabechi 1990)

Quasi-Destructive Graph Unification with Structure Sharing (Tomabechi 1991)

Both are contained in my 1993 CMU Ph.D. Thesis (readable online, 🔍 "Tomabechi Ph.D Thesis")
http://maxpec.net/amazoncampaign_present/Dr_Tomabechi_treatise.pdf

Used in NLP grammar in 90s, now usable as basic data structure (100 million times fast CPUs)

What is Feature Structures

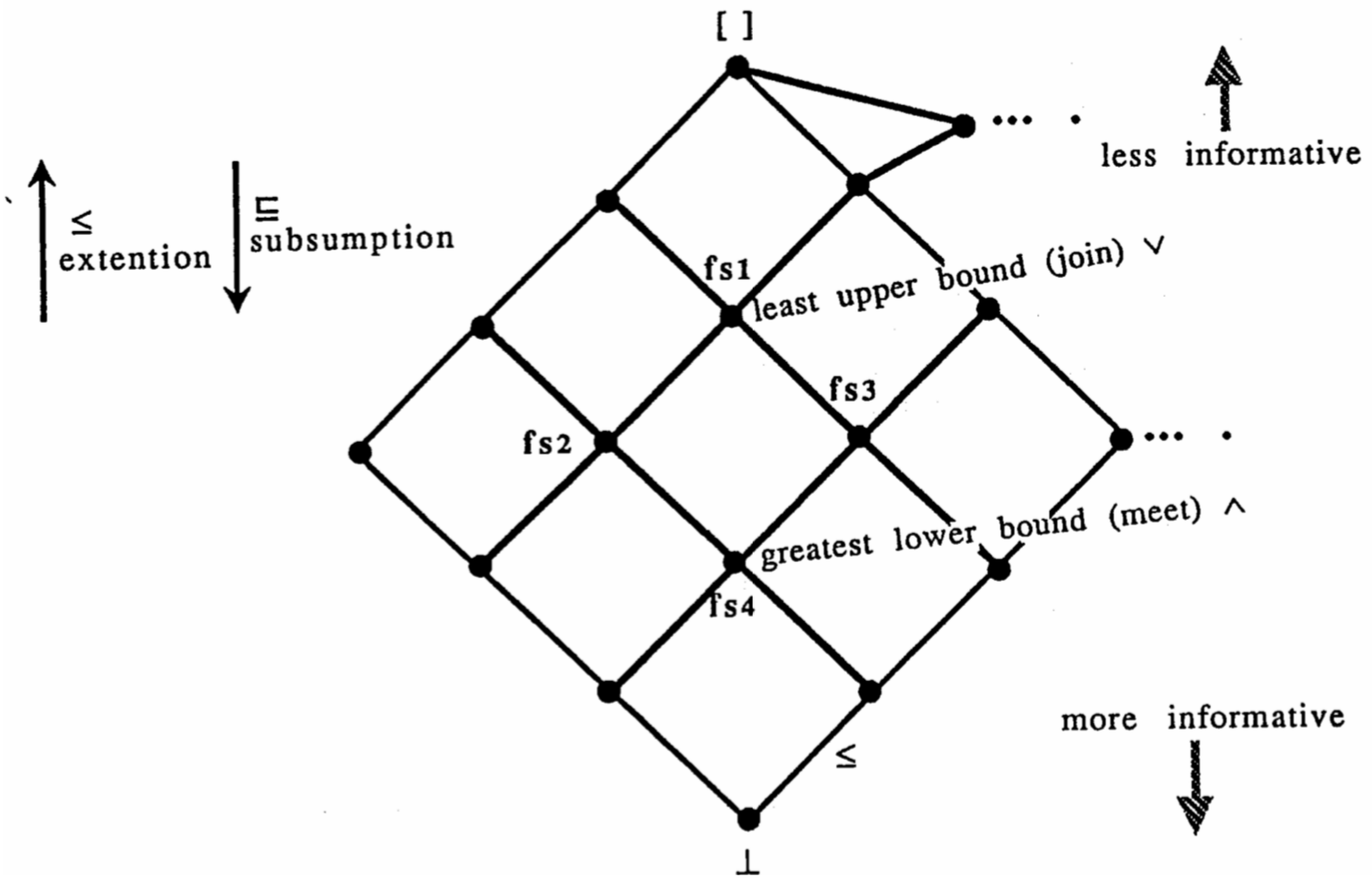
Set of labels and values
[[label1 value1]
[label2 value2]]

Value can be embedded feature structures
[[label1 value1]
[label2 [[label3 value3]
[label4 value4]]]

Value can be contradiction (Bottom) or variable (Top [ ])
[[label1 value1]
[label2 Bottom]
[label3 [ ]]]

Feature Structures as Set-Theoretic Partial Functions

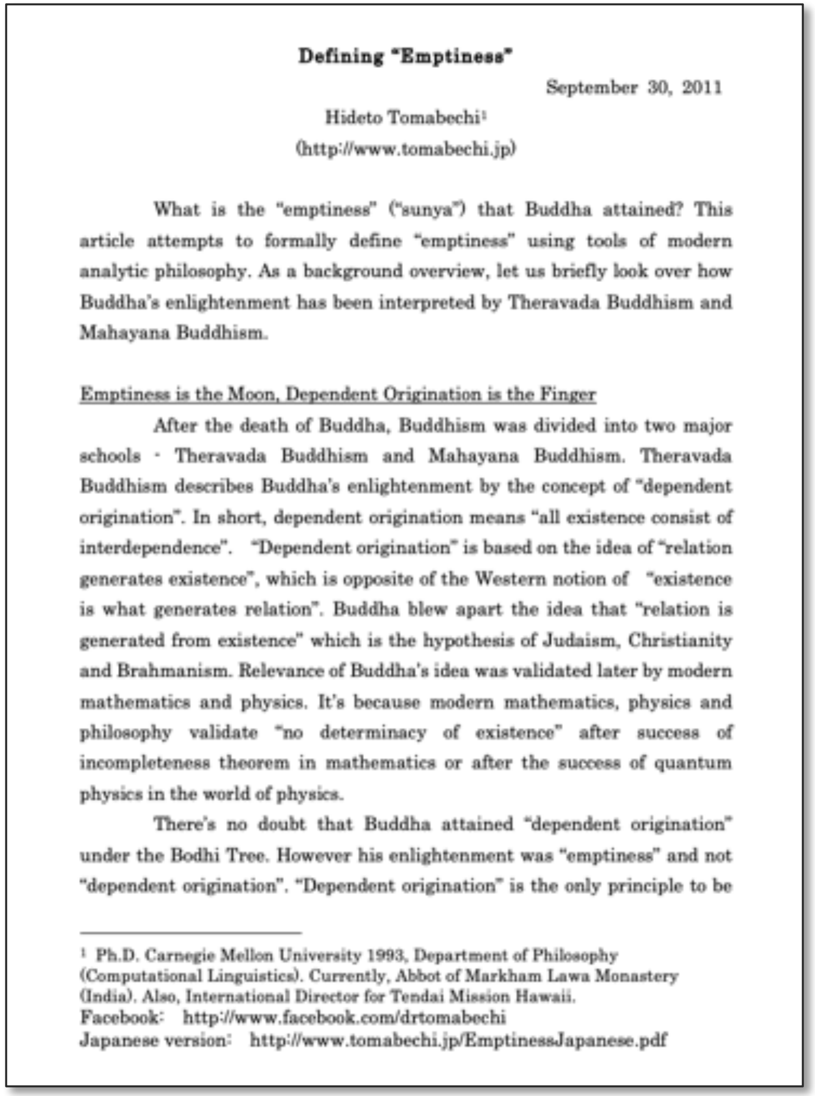
Everything in Universe (including concepts) are represented as partial functions



Combining East with West

Top “*emptiness*” （空）
Bottom *contradiction*

http://tomabechi.jp/EmptinessDrTomabechi.pdf
or
🔍 "Tomabechi emptiness”



Defining "Emptiness"

September 30, 2011

Hideto Tomabechi[1]
(http://www.tomabechi.jp)

What is the "emptiness" ("sunya") that Buddha attained? This article attempts to formally define "emptiness" using tools of modern analytic philosophy. As a background overview, let us briefly look over how Buddha's enlightenment has been interpreted by Theravada Buddhism and Mahayana Buddhism.

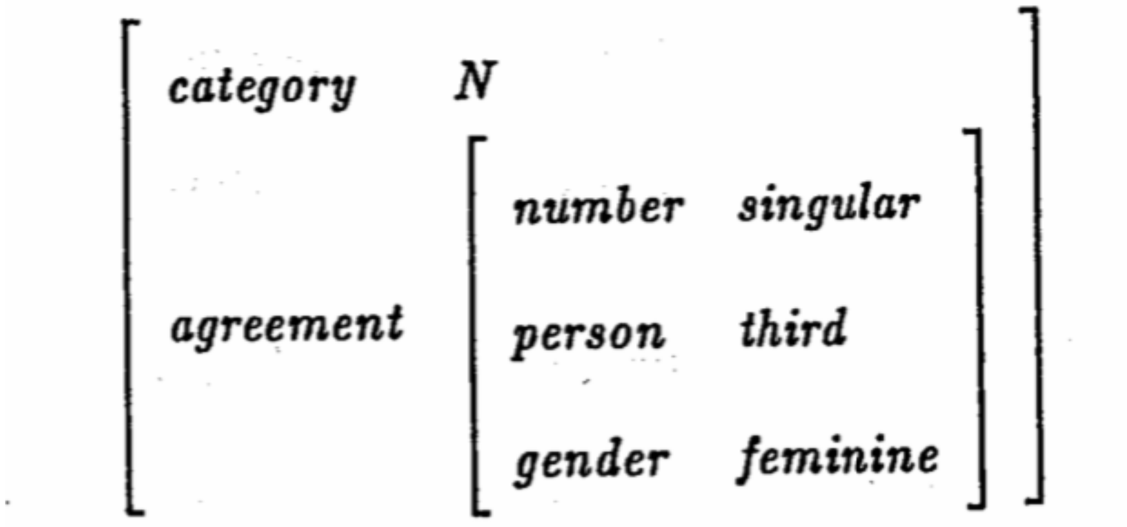
Emptiness is the Moon, Dependent Origination is the Finger

After the death of Buddha, Buddhism was divided into two major schools · Theravada Buddhism and Mahayana Buddhism. Theravada Buddhism describes Buddha's enlightenment by the concept of "dependent origination". In short, dependent origination means "all existence consist of interdependence". "Dependent origination" is based on the idea of "relation generates existence", which is opposite of the Western notion of "existence is what generates relation". Buddha blew apart the idea that "relation is generated from existence" which is the hypothesis of Judaism, Christianity and Brahmanism. Relevance of Buddha's idea was validated later by modern mathematics and physics. It's because modern mathematics, physics and philosophy validate "no determinacy of existence" after success of incompleteness theorem in mathematics or after the success of quantum physics in the world of physics.

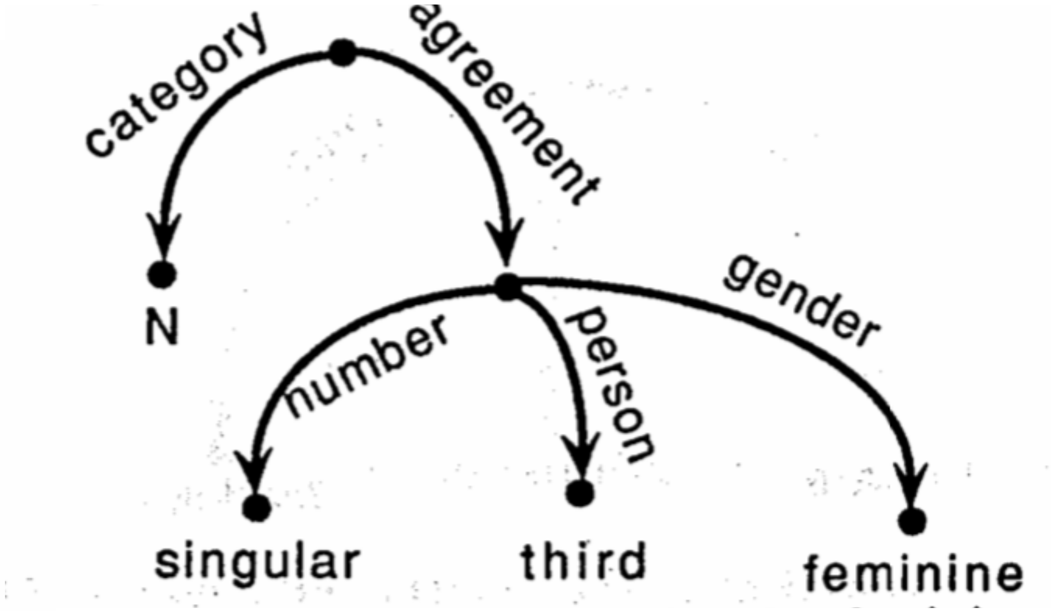
There's no doubt that Buddha attained "dependent origination" under the Bodhi Tree. However his enlightenment was "emptiness" and not "dependent origination". "Dependent origination" is the only principle to be

[1] Ph.D. Carnegie Mellon University 1993, Department of Philosophy (Computational Linguistics). Currently, Abbot of Markham Lawa Monastery (India). Also, International Director for Tendai Mission Hawaii.
Facebook: http://www.facebook.com/drtomabechi
Japanese version: http://www.tomabechi.jp/EmptinessJapanese.pdf

Feature Structure as Graph Theoretic Directed Graph



$$\begin{bmatrix} \textit{category} & N \\ \textit{agreement} & \begin{bmatrix} \textit{number} & \textit{singular} \\ \textit{person} & \textit{third} \\ \textit{gender} & \textit{feminine} \end{bmatrix} \end{bmatrix}$$



Representing a third person singular feminine noun entity

Reentrant feature structure

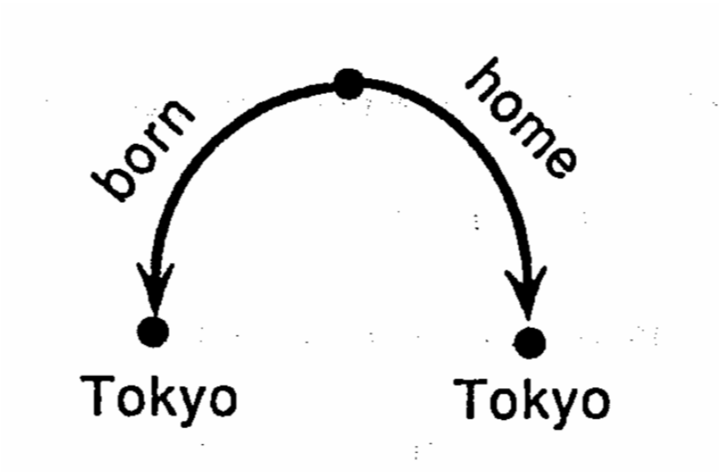
Non-reentrant （Tokyo maybe referring to different Tokyo）

[[born Tokyo]
  [home Tokyo]]

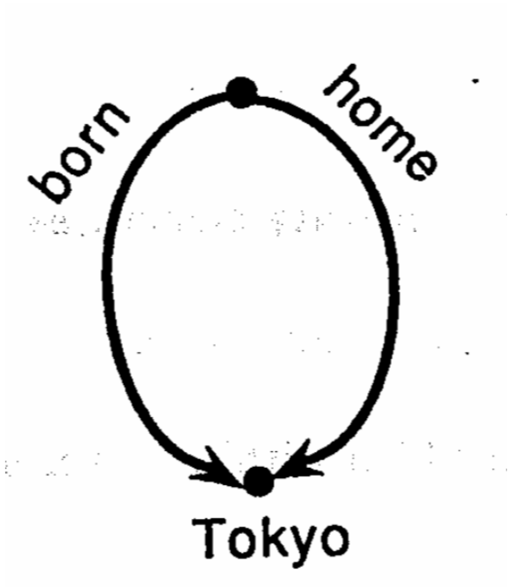
Reentrant feature structure （Tokyo referring to same Tokyo）

[[born X01 Tokyo]
  [home X01]]

Reentrant feature structure in graph notation



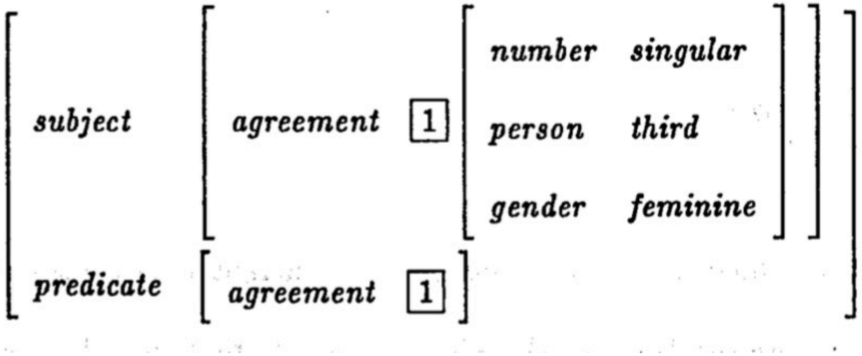
Non-reentrant feature structure



Reentrant feature structure

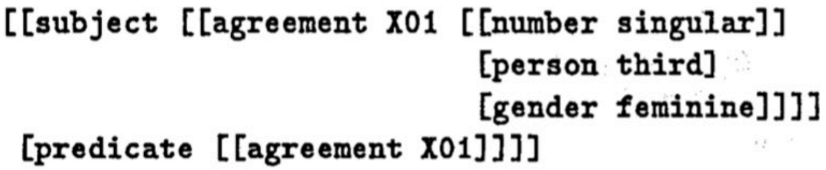
A reentrant structure graph with a complex value

Using common notation found in linguistic literature

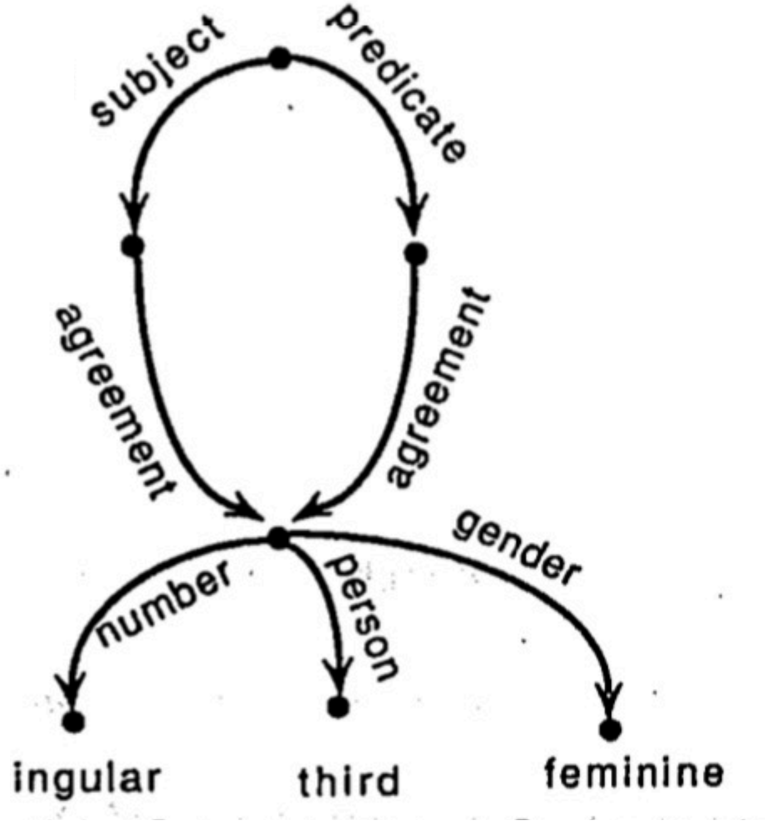


$$\begin{bmatrix} \textit{subject} & \begin{bmatrix} \textit{agreement} & \boxed{1}\begin{bmatrix} \textit{number} & \textit{singular} \\ \textit{person} & \textit{third} \\ \textit{gender} & \textit{feminine} \end{bmatrix} \end{bmatrix} \\ \textit{predicate} & \begin{bmatrix} \textit{agreement} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

In bracketed notation

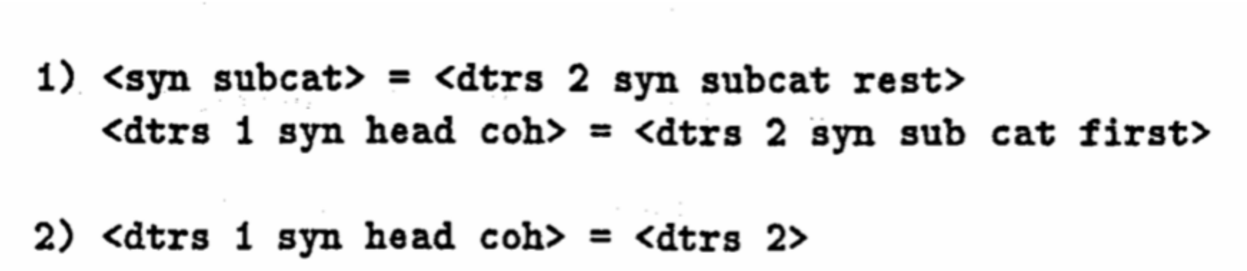


```
[[subject [[agreement X01 [[number singular]]
                           [person third]
                           [gender feminine]]]]
 [predicate [[agreement X01]]]]
```



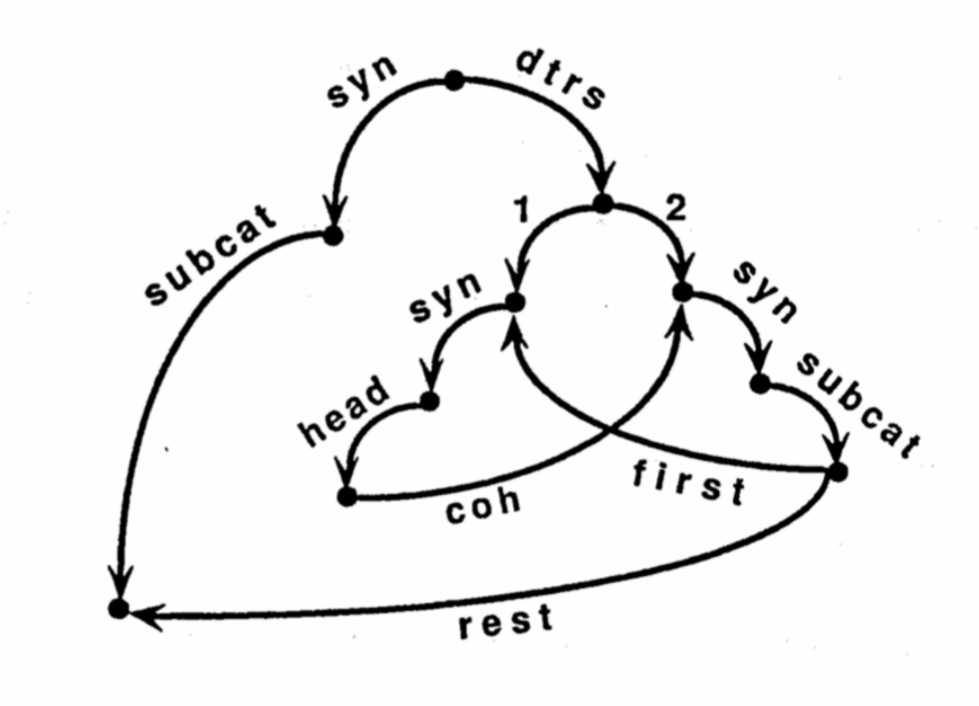
Feature structure with cycle (directed graph with a cycle)

Example of actual English grammar rules (using path equation notation)



```
1) <syn subcat> = <dtrs 2 syn subcat rest>
   <dtrs 1 syn head coh> = <dtrs 2 syn sub cat first>

2) <dtrs 1 syn head coh> = <dtrs 2>
```



Graph Unification on Feature Structure Graph

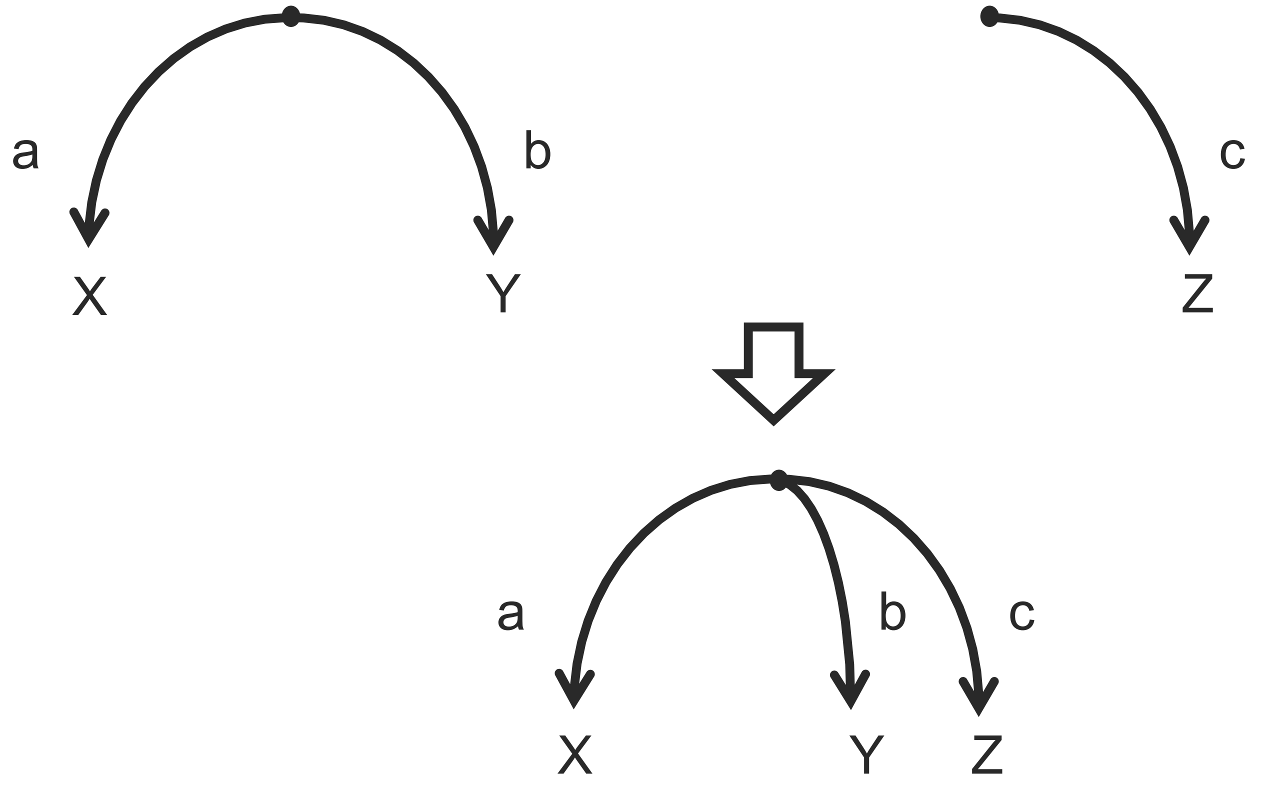
Monotonic

Efficient detection of contradictions

Non-destructive efficient data structure building

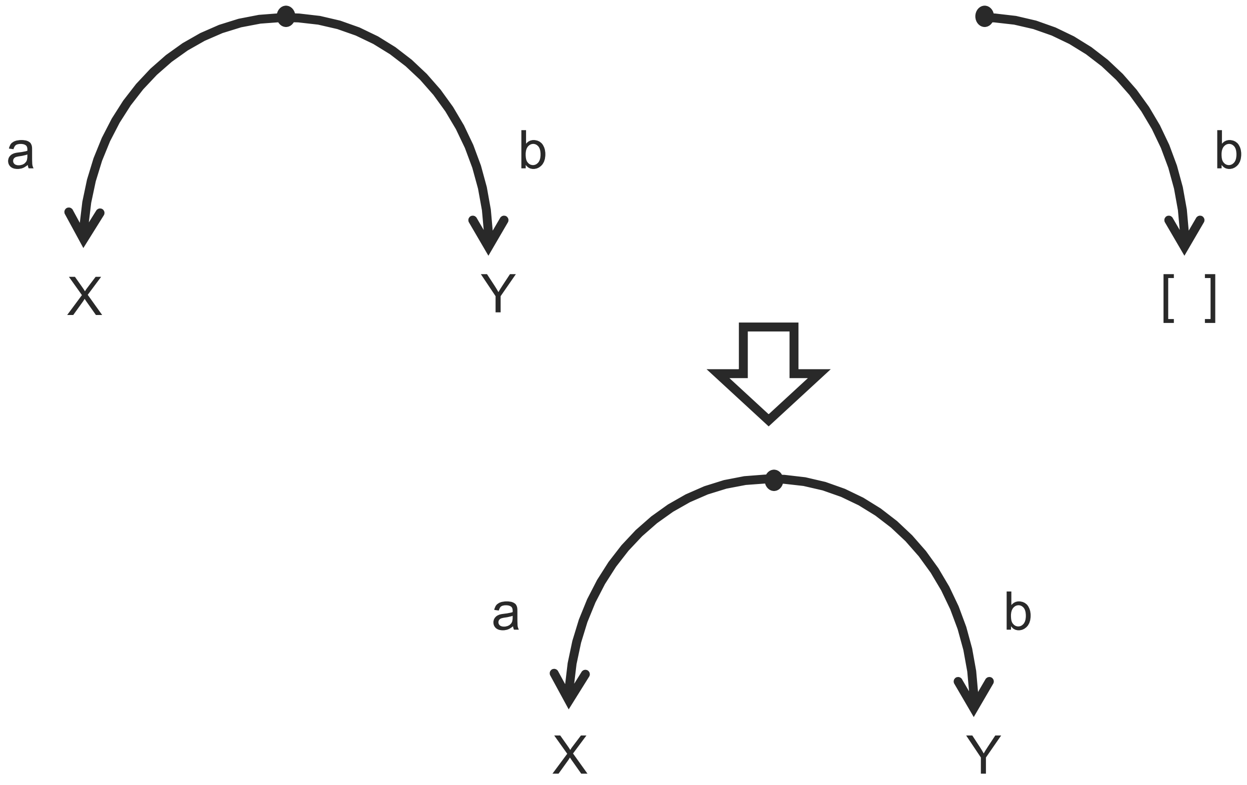
Unification as purely monotonic operation

Simple Example of unification two graphs



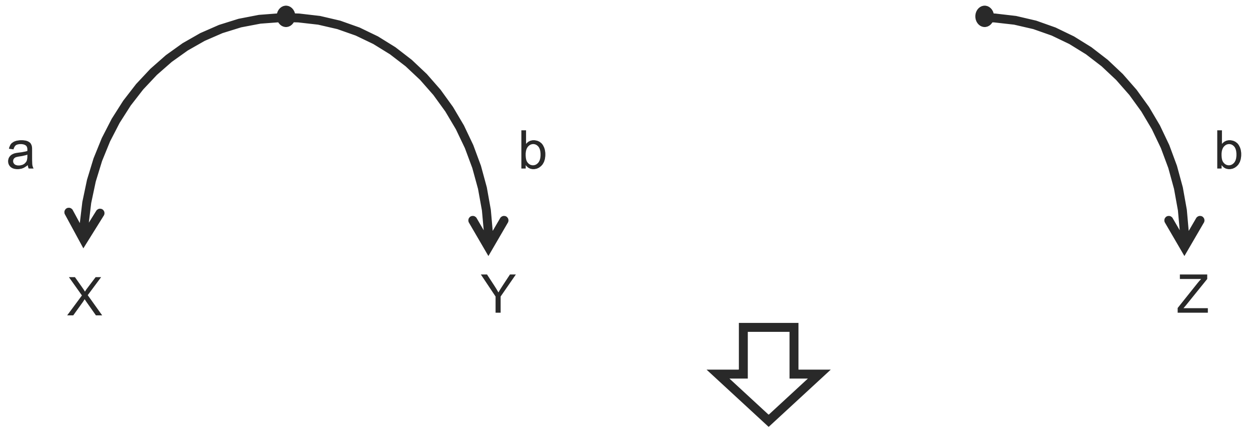
Unification as purely monotonic operation

Simple Example of unification two graphs



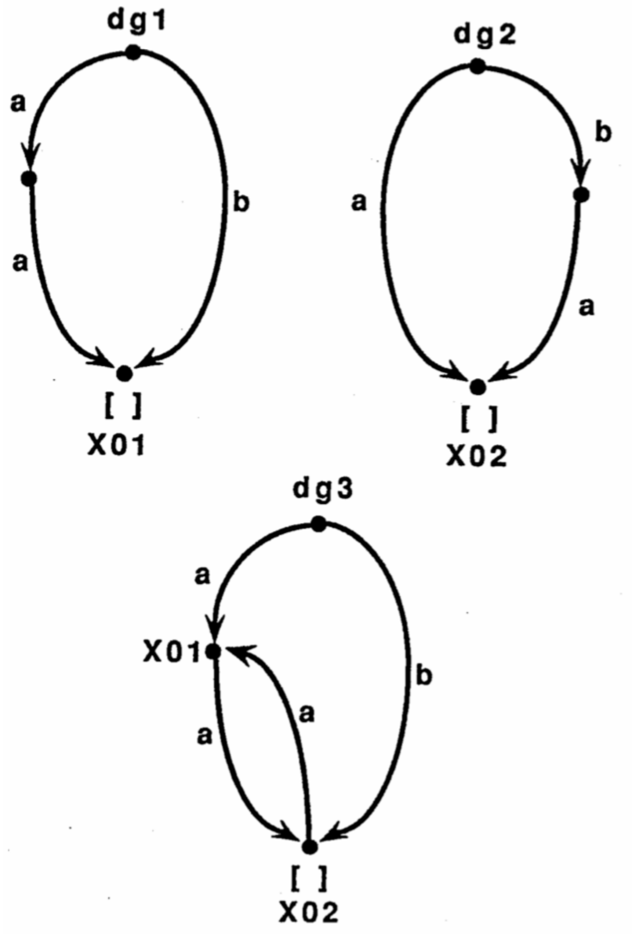
Unification with failure

Simple Example of unification two graphs



⊥ **Contradiction**

Unification resulting in a cycle



dg1

$$\begin{bmatrix} a & \begin{bmatrix} a & X01 \end{bmatrix} \\ b & X01 \end{bmatrix}$$

dg2

$$\begin{bmatrix} a & X02 \\ b & \begin{bmatrix} a & X02 \end{bmatrix} \end{bmatrix}$$

dg3

$$\begin{bmatrix} a & X01 \begin{bmatrix} a & X02 \end{bmatrix} \\ b & X02 \begin{bmatrix} a & X01 \end{bmatrix} \end{bmatrix}$$

Unification algorithm should not result in infinite loop even with acyclic graph input

True monotonicity at data structure and algorithmic level

Blockchain is a pseudo monotonic with hashing and signature

Non-Destructive Graph Unification is a true monotonic algorithm

Unification as monotonic algorithm

1. Algorithmic guarantee of monotonicity
2. Very fast detection of contradiction
3. Non-destructive result building
4. Very efficient (linear time) complex structure building

Monotonicity in Near Future Computations - Applicational

Under zero-trust paradigm,
we assume encryption/hashing do not work.

Resilience is achieved if monotonicity is assured.

Efficient detection and
constant/linear time (at least log N) original recovery.

Monotonicity in Near Future Computations
- Architectural (for quantum resilience)

Even linear time algorithms require
very fast CPU for low level monotonicity

Machines are 100 million times faster than 1990
(invention of linear time algorithms)

Current CPUs allow for low level monotonic data structures
at O/S level

→ My current research includes design of entire computer
and network with monotonicity

Preferred direction for monotonicity

All basic data structure for computation and network must be monotonic

Must be embedded in low level data structure in future operation systems

Dedicated monotonicity chips by micro code unification or through next generation material engineering