解説／展望

知識情報構造の形式記述と解釈可能性についての展望
Formal Description of Knowledge Information Structure and Its Interpretability

天野晃 † 坂本浩一 †
Kou AMANO Koichi SAKAMOTO

† 物質・材料研究機構 統合型材料開発・情報基盤部門
Research and Services Division of Materials Data and Integrated System, National Institute for Materials Science
〒 305-0044 つくば市並木 1-1
EMail: AMANO.Kou@nims.go.jp

知識情報構造を表現可能な構造として均質化 2 部グラフが提案されて以来、その形式記述とパーサーの実装に関しての議論は殆どなされてこなかった。本論文では、当該構造の表現が従来の言語では困難であることを示し、均質化 2 部グラフを表現する言語とそのパーサーの実装について提案を行う。また、提案された仕組みを利用したいくつかの応用例を示し、これを元に知識の構造化と構造化（形式化）された知識に対する解釈に関しての展望を述べる。また、コンセプト実証のために作成されたツールについても解説する。当該ツールは、均質化 2 部グラフを形式記述可能な言語としては初めてのものである。なお、本論文は展望論文であるため、事実と考察が混在したスタイルを用いており、結果や考察の章は設けていない。

Since the homogenized bipartite graph was proposed as a structure that can represent the knowledge information structure, there has been little discussion on its formal description and parser implementation. In this paper, we show that the representation of the structure is difficult in a conventional language, and propose a new language to represent a homogenized bipartite graph structure and its parser implementation. In addition, we describe some applications using the proposed mechanism and the perspectives on structuring knowledge and interpreting structured (formalized) knowledge. We also describe the tools implemented for the proof of concept. This tool is the first language that can formally describe a homogenized bipartite graph structure. This paper uses a style in which facts and considerations are mixed.

キーワード: 木; グラフ; ハイパーグラフ; 均質化 2 部グラフ; 項参照システム; 木参照システム; 知識構造; 形式記述

1 背景

意味処理など高度な知的処理を計算機上で実現するには、意味処理のロジックとは別に、知識や概念の構造（本論文では知識情報構造と称する）を表現するためのモデル化手段と、モデル化された構造を計算機可読な形式で記述するための形式記述手段 (記述言語など) が必要である。

まずモデル化手段に関しては、筆者らは特にオントロジー構築における辞書記述への適用を念頭に自然言語の語彙の特徴的な性質である双対性[1]に着目した。双対性の概念は、述語論理や内包論理 [1] 等の古典的なロジックでは明確な扱いは見受けられず、状況意味論 (状況理論) において ”Everything is a first class citizen.” のスローガンのもと意識が見られる [2, 3, 4, 5]。具体的には、関係 (relation) や状況 (situation) など各種のオブジェクトがそれ自体で通常の個体 (individual) と同様に考察や言及の対象となり、関係のパラメータとなりうる。さらに藤原 [6] が提唱した均質化 2 部グラフにおいては、双対性の

[1]同じ語が relation にも entity にもなりうる性質。古典的なロジックにおいては relation と entity は数学的に全く別のオブジェクトであり (例:relation は entity の集合の直積の部分集合)、双対性の扱いは困難であった。

概念を入れ子構造とともに直接グラフの定義に取りこんでおり、明示的に双対性を考慮しているといえる。その結果、状況理論と同様 non-well-founded[2] な集合が導入されている。

一方で、知識情報構造に対応する形式記述については、現在に至っても具体的な設計は行われていない。

本論文では、藤原の提唱した均質化 2 部グラフが知識情報構造を表現し得ることを前提とし、均質化 2 部グラフの形式記述（すなわち言語）とパーサー（本論文では解釈器またはインタープリターと呼ぶ）の実装、およびそれに必要なデータ構造について議論する。したがって、本論文では意味処理のロジック、述語論理や数理解析等に関する具体的な議論は行わない。このことを明確にするために、本論文では、知識情報構造の解釈器をインタープリター、(意味) 解析器をソルバーと呼ぶ。

2 目的

知識情報構造を表現可能であるとされる均質化 2 部グラフを表現可能な言語を提案し、実際に知識情報構造を記述可能であることを、例示により検証する。

本論文では、均質化 2 部グラフで知識情報構造を表現可能であることを前提とし、以下の均質化 2 部グラフを表現可能な言語を設計、実装する。まず、均質化 2 部グラフを下式の定義とする [7, 8, 9]。

$$E \subseteq 2^V \tag{1}$$

$$V = V \cup E \tag{2}$$

$$E = E \cup V \tag{3}$$

$$\sigma : L \to E \cup V \tag{4}$$

ここで、E はグラフのエッジ、V はバーテックスであり、これらは、式 2 および式 3 により均質化されている。ここでの均質化とは、藤原も述べているように、常にオブジェクトが $V \wedge E$ であるわけではなく、場合によっては E のみ、V のみであることも有り得る。したがって、タイプ判定を必要とするものである。L はラベルであり、任意の E および V にラベル付与可能であることを示している。ハイパーグラフのような低次の表現については、前述の通り既報があるので [6, 10, 11]、そちらを参照されたい。以上に基き、さらに目的を具体化し、検証可能なタスクとして定義する：

- 多項性 : 一つの E により任意の数の V のリンクが可能（ハイパーエッジ）である。
- 双対性（均質化）: オブジェクトが V、E、$V \wedge E$ で有り得る、かつ、タイプを判定可能である。
- 入れ子（内部構造）: オブジェクトが $V \wedge E$ のとき、V に内部構造を持ち得る。（E の内部構造は自明。）

以上は、言語を形式的に精査することによって行う。

3 言語と内部データ構造

3.1 従来の言語

最初に、既存言語におけるグラフ様表現の現状について述べると、グラフを記述可能な言語は多い。Maple[12] や Wolfram Language[13] は標準でグラフ表現と解析の機能を持っている。しかし、ハイパーグラフや均質化 2 部グラフを言語的に表現するツールは無く、行列表現等を組み合わせてユーザーが定義する必要がある。特に内部構造を表現することは難しく[3]、結合隣接行列のみでは困難である。

ここでは、既存の言語実装の代表として Wolfram Language を用い、均質化 2 部グラフの表現が困難であることを示し、その要因を考察する。なお、他の言語においては、lisp[14] のように同様の挙動となるか、Maple のように再帰的なアサインが禁止されているかのどちらかであることを確認している。

まず、均質化 2 部グラフを構成するには、ハイパーグラフの表現機能が必要になるが、Wolfram Language には用意されていない。そこで、エッジを単なるリストで表現する。図 1 は、リスト構造と代入を利用し、均質化 2 部グラフを構成した例である。$In[1]$ では、右辺の $\{l, l, l\}$ という同一シンボル列を l に代入している。各 l が、自分自身を含む右辺の構造を参照する構造となっており、内部的には l に自身の構造が 3 つ内包される均質化 2 部グラフとして構成されているはずである（図 2）。これを確認

[2] 直感的には、x = {a, x} の解である x = {a, {a, {a, ...}}} のようなネストが無限に深い構造の集合 [4]。

[3] 技術的には難しくないはずであるが、多くの場合、表示量が厖大になるため実装が行われることは稀であり、例外はデバグ情報である。

するには、l を評価すればよいが、シンボル解決の段階で評価がホールドされており（その理由は再帰評価の回数制限による）、内部構造を確認できない。

以上の検証より、本論文の目的に対しては十分な考察が可能である。まず、均質化 2 部グラフが構成されていると仮定して、これと矛盾する挙動はない。一方でこれを確認する手段も見いだせない。本論文の目的からは、均質化 2 部グラフの構成とその表現の確認の双方が行われる必要があるが、既存言語においては、これを満たしておらず、その要因は幾つか考えられる：

- 構造を完全に表現できない：内部構造が表示されないために構造の確認ができない。
- 代入とは異なる仕組みで項を参照（表示）する必要がある：代入値を評価できない構造が構成されている可能性があり、これを回避する別の手段（参照系）が必要である。
- 言語から内部構造へのマップと内部構造の（意味）解析を分離する必要がある：これらを同時に行っているために、構造表現ができない可能性がある。
- サイクリックな構造を再帰構造と混同している：ここで構成されるのは有限なサイクリック構造であるが、これを再帰評価していることに問題がある。

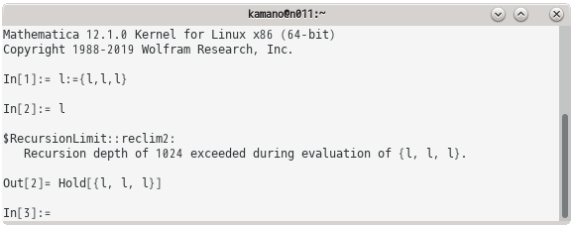

図 1: Wolfram Language による均質化 2 部グラフの構成

次節では以上の考察を元に、必要な言語機能について議論する。

3.2 言語の提案

前節において考察した 4 点すべて、または一部を満す言語があれば、均質化 2 部グラフの構成とその確認が可能であると考えられる。これについて、あらためて要件定義とその実装を提示する：

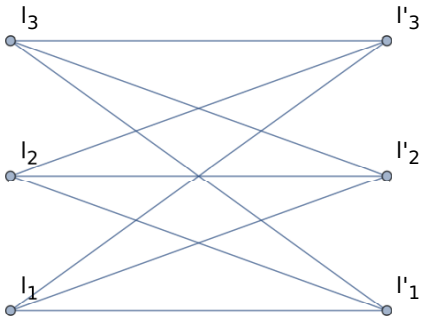

図 2: 均質化 2 部グラフの構造. l は自身の構造を 3 つ内包するが、それぞれの l は均質化されておりバーテックスとエッジ双方の側面を持つ。即ちバーテックス l_1〜l_3 であるとともにエッジ l'_1〜l'_3 でもある。さらにエッジ l'_1〜l'_3 はそれぞれ l_1〜l_3 をリンク（これにより、$l \times l$ の高位のグラフを形成しているとも言える）する。この構造がひとつのシンボル l にアサインされており全体で一つのオブジェクト（すなわち内部構造）と捉えることができる。

- 式だけでなく内部構造の情報を表示する必要がある: 項の構造体情報を表示する。
- 代入とは異なる仕組みにより項を参照する必要がある、あるいは代入よりも厳密な項参照の表現が必要である: 厳密な項参照の機能を実装する、すなわち、参照項・被参照項が明示されること、および、参照が成功したかが明示されること。
- インタープリターとソルバーを分離する必要がある: 当該言語ではインタープリターのみ実装する。
- 再帰的な構造とサイクリックな構造を分離する必要がある: 項参照機能の拡張として木参照機能を実装する。またその表現は項参照と木参照が明確に区別できる表現を用いる。項参照がサイクリック構造、木参照が再帰構造を表す。

本論文では以上のすべてを満す言語を定義、実装する。上記要件によれば、代入機能とソルバー機能は不要となる。

以上の議論に基き、言語設計を行う。言語はS式とする。特に準備すべき機能は参照である。この表現を次のように定義する:

- 項参照番号（同時に被参照項を表す）:
  #<数字>
- 項参照（同時に参照項を表す）:
  $#<数字>
- 木参照番号（同時に被参照木を表す）:
  ##<数字>
- 木参照（同時に参照木を表す）:
  $##<数字>
- 項の参照ルール：参照項は被参照項と同一オブジェクトと見なす。
- 木の参照ルール：参照木は被参照木の複製と見なす。
- 参照解決:参照と被参照のペアが確定することを参照解決という。参照解決された場合、参照項にマークが付与される（次節において詳述）。なお、
  - 被参照項に対して木参照を行なった場合、参照は失敗する。
  - 被参照木に対して項参照を行なった場合、参照は成功する。

また、式4を満たすために、参照または参照番号の後にアルファベットによるラベルを付与可能とする。

さて、前述したように均質化2部グラフのオブジェクトは2面性（タイプ）を持つので、その表現におけるタイプ判定が必要である。すなわち、そのタイプはバーテックス (V) のみ、エッジ (E) のみ、バーテックスかつエッジ ($V \wedge E$) の何れかであるが、この判定を次のように行う（参照は解決していると仮定）:

- V: リーフノード、参照項の場合は参照先もリーフノードである、被参照項の場合は参照元もリーフノードである。
- E: 子要素を持つ、参照項の場合は参照先も子要素を持つ、被参照項の場合は参照元も子要素を持つ。
- $V \wedge E$: (1) リーフノードの場合、参照項であり参照先が子要素を持つ、または被参照項であり参照元が子要素をもつ。(2) 子要素を持つ場合、参照項であり参照先がリーフノードである、または被参照項であり参照元がリーフノードである。
- ダミーノード: 参照表現、被参照表現、ラベルのいずれも付与がない（空文字列）ノードはオブジェクトとは見なさずダミーノードとする。

また、サイクリック構造と再帰構造を次のように定義する:

- 項参照の循環をサイクリック構造と見なす。
- 木参照の循環を再帰構造と見なす。

以上を構成可能な言語仕様を表15のように定義する。

最後に、この形式をT式（T-form）と名付ける。

3.3 インタープリターの提案

まず、内部構造を提案する。最小限必要となる内部構造は、たったひとつの構造体である（表1）[4]。T式のすべてのノードにこの構造体が作成される。

表 1: 構造体定義

```
struct tree{
        char* Head;
        struct tree* Ref;
        struct tree** Child;
}
```

インタープリターの動作も次のようにシンプルである：

1. T式がキャラクタのシークエンスとして読み込まれる
   (a) 構造体がノード毎に深さ優先で構成される

[4] 実際の実装では操作容易性のためメンバーが追加されている。

2. 構成された T 式が再度深さ優先でスキャンされ、参照処理を行う

インタープリターの出力は次の 3 形態とする：

- 解釈された T 式
- 項の構造体情報のリスト
- 参照関係判断を容易にするよう表現が拡張された隣接行列

以上、T 式とそのインタープリターにより前述した要件が満たされた：

- 内部構造の表示: 構造体情報の表示機能により満たされた。
- 項参照: 構造体メンバー Ref と参照処理により満たされた。
- インタープリターとソルバーの分離: インタープリターのみを実装することにより満たされた。
- 再起構造とサイクリック構造の分離: 明確な言語仕様により満たされた。

最後に、この T 式のインタープリターを tq（T-form Query language）と名付ける。

3.4 言語実装

以上の提案に基づき、以下の要件を満す言語 tq を実装した [15]。

- インタープリターにおいて参照解決が行われること（次節で検証）。
- インタープリターは前述の 3 形態の情報出力機能を持つこと（次節で検証）。
- 外部システムによりオブジェクトタイプ（V、E、$V \wedge E$）判定が可能な表現であること（次節で検証）。
- 外部システムにより再帰構造表現かサイクリック構造表現かの判別が容易となる表現であること（前述の説明より自明）。

3.5 動作検証

前述の要件定義の確認を動作検証において行う。

参照解決　参照には禁則事項があり、参照番号は（項/木あわせて）唯一とする。これはオブジェクトを特定するためである[5]。参照表現を表 2 に例示する。例のように参照解決が行われると、参照項に”@”が付与されさらに解決された表現が付与される。被参照側からは解決が行われたかは判別できず、出力式全体に対して検索を行う必要がある。

表 2: 参照表現の入出力

入力	出力
#1($#1)	#1($#1@#1)
$#1(#1)	$#1@#1(#1)
(#1,$#2,#2$#1)	(#1,$#2@#2$#1@#1 ,#2$#1@#1)
(#1,$##1)	(#1,$##1)
(##1,$#1)	(##1,$#1@##1)
(##1(#2),$##1)	(##1(#2),$##1@##1(#2))

3 形態の情報出力　前章において Wolfram Language により構成した均質化 2 部グラフ（図 2）を例に検証を行う。

式 5 の表現は、オブジェクト判定の定義により、自身の構造を内包する一つのオブジェクトであり、図 1 と同等の表現のであると見なせる。

$$\#1(\$\#1, \$\#1, \$\#1) \tag{5}$$

[T 式] 表 3 は、tq による式 5 の解釈後の表示である。1 行目は標準出力され、2 行目は標準エラー出力される。前述の通り、参照項が参照解決されていることが分かる。

[構造体情報] 表 4 は、tq による式 5 の構造体情報の表示である。表 1 に示す構造体情報をすべて表示すると Child メンバーの情報が煩雑になることから、Child 情報は子要素のカウントのみ表示し、親子関係の判別は Child 側に Parent メンバー情報を表示することで行う。Adr は項のアドレス、PaAdr は親項のアドレス（無い場合は 0）、Ref は参照先のアドレス（参照が無いまたは解決が行われなかった場合は 0）、H は Head、C は子要素のカウントである。:SN=＜数字＞:の数字部分はシステムにより採番された（処理された順の）項番号である。

[5] ただし、現実装では禁則処理は行っていない。

[隣接行列] 表 5 は、tq による式 5 の拡張された隣接行列の情報表示（2 部グラフの構造情報と同値）である。行と項が対応しており、各行にさらに各項との対応情報が記述されており、区切り子はカンマである。対角要素は自分自身の情報であり、特別に括弧 [] 内に記載されている。この行列には、ほとんどすべての情報が含まれており、たとえば、第 0 項の"->0->0->0"は、この項が、被参照項として 3 回参照解決されたことを示す。"=><数字>"というパターンは、親項の番号であり、参照解決された場合にのみ表示される。以上によれば、見た目からも明らかであるが、式の項数は 4 であり、前述した同様の構造、図 2 から伺える項数 3 とは異なる。これは、図に表現されているのは l の内部構造のみであり、オブジェクトそのものが l に内包されているという情報を欠いているからである。

表 3: tq による T 式の出力

行番号	出力
1	#1($#1@#1,$#1@#1,$#1@#1)
2	4 Nodes were operated.

表 4: tq による構造体情報表示

行番号	表示
0	:SN=0::Adr=30590592::PaAdr=0: :Ref=0: :RT=h::H=#1::C=3::
1	:SN=1::Adr=30595968: :PaAdr=30590592: :Ref=30590592::RT=::H=$#1::C=0:
2	:SN=2::Adr=30596128: :PaAdr=30590592: :Ref=30590592::RT=::H=$#1::C=0:
3	:SN=3::Adr=30596320: :PaAdr=30590592: :Ref=30590592::RT=::H=$#1::C=0:

オブジェクト判定　まず、V、E、$V \wedge E$ の基本入力表現を表 6 に例示する。これらは、式の一部ではなく、式全体である。表記例 "#1()" では、当該項がリーフノードとなることを防ぐために、括弧 () 内

表 5: tq による拡張隣接行列表示

行番号	表示
0	[:#1:0]->0->0->0,:1,:2,:3,
1	,[0=>:$#1:1->0],,,
2	,,[0=>:$#1:2->0],,
3	,,,[0=>:$#1:3->0],

隣接行列ではすべての項（$E \cup V$ のすべて）が出力されており（$n \times n$ の行列）、行がエッジオブジェクトとして、列がバーテックスオブジェクトとしての情報となる。

にダミーノードを構成している。Head が空である項はダミーノードであり、オブジェクトとは見なさない。tq は Head が空であっても括弧 () のみで言語の木構造を構成する仕様になっており、入力が空であったとしても、最小でも一つの項を構成する。なお、項とは言語の木構造を構成するノードであり、これにより表現される構造におけるオブジェクトとは異なる概念である。現仕様では、参照解決されない参照項をオブジェクトと見做さない。

表 6: tq による基本表現

タイプ	表記例	項数	オブジェクト数
V	#1	1	1
-	$#1	1	0
E	#1()	2	1
-	$#1()	2	0
V∧E	#1($#1)	2	1
V∧E	$#1(#1)	2	1
-	()	2	0
-	()()	3	0
-	(())	3	0

-：未定義

再帰構造、サイクリック構造の判別　前述したとおり、自明である。

4　コンセプト実証

本章では、いくつかのコンセプト実証を行う。

また、外部システムとの通信を考慮する際に必要となる概念、オーソライズを導入する。

4.1 知識情報構造（標準模型）の記述

知識情報構造の一つである標準模型の記述を例示する。

標準模型は Wikipedia に掲載されているもの [16] を利用する。

図 3 は Wikipedia より転載したものである。この構造を T 式で表すと表 7 の通りとなる[6]。tq が出力する各情報（T 式、構造体情報、拡張隣接行列）を検索することで、様々な知識情報構造の解析が可能となる。これらの出力は、grep 等の一般的なツールを利用して解析可能な表現になっている。このことをいくつかの例により示す。

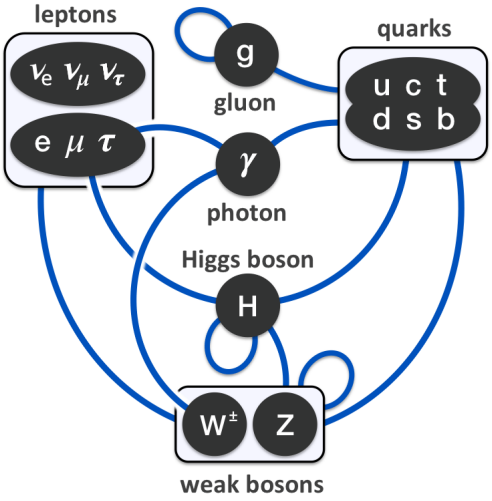

図 3: 標準模型の模式図

(1) 実体粒子の抽出：総称は実体と見なさない、即ち、leptons、quarks、weak bosons は対象としないとする。この条件下での実体粒子の抽出は、子ノードを持たず、参照項でないものを選択すれば良い。構造体情報を処理した例を表 8 に示す。0 行目は実行コマンドである。表 7 と同様の記述がファイル”standard-model-net.t”に記述してある。表には省略しているが、図 3 に示される実体粒子である、E（electron）、M（muon）、T（tau）、nE（electron neutrino）、nM（muon neutrino）、nT（tau neutrino）、u（up quark）、d（down quark）、c（charm quark）、s（strange quark）、t（top quark）、b（bottom quark）、W（W bosons）、Z（Z boson）、g（gluon）、p（photon）、H（Higgs boson）の 17 粒子が正しく抽出されている。[7][8][9]

表 7: tq による標準模型の形式表現

行番号	T 式
1	(
2	#1leptions(#7(E,M,T),nE,nM,nT),
3	#2quarks(u,d,c,s,t,b),
4	#3weak bosons(#8W,Z),
5	$#3($#1,$#2,$#3),
6	$#4(#4g,$#2),
7	#5p,
8	$#5($#2,$#7,$#8),
9	$#6(#6H,$#2,$#7,$#3)
10	)

(2) グルーピングに使われる項の抽出：子要素を持ち、参照項でない項を抽出すればよい。構造体情報を処理した例を表 9 に示す。粒子以外の項も含め抽出されており、木を構成するためのダミーノードである第 0 項（1 行目、ただし、前述のルールよりオブジェクトではない）、荷電レプトンのグルーピングに利用した”#7”、leptons、quarks、weak bosons、が抽出されている。

(3) 粒子グループの抽出：前述の抽出結果より、ラベル付与されているものを抽出すればよい。すなわち、leptons、quarks、weak bosons、を抽出可能である。

(4) 自己相互作用する粒子の抽出：親子関係が自分自身に構築されている項を抽出すればよい。隣接行列より、自身の項情報（括弧 [] で囲まれた情報）以外の要素に自身の項番号が表示されているものを抽出し、それが参照項であった場合は被参照項に置き換える。この処理を行った例を表 10 に示す。この抽出はルールは明確だが、操作はそれほど自明ではなく、短いスクリプト（数行の perl スクリプト）を書く必要があった。正しく、weak bosons、g、H が抽出されている。

[6] ただし、この解釈は筆者らが行ったものであるが、ここではその解釈が形式的に記述されている点が重要である。

[7] 図では W^+ と W^- が $W^\pm$ の 1 粒子で表されておりその通りに記述した。

[8] gluon は 8 色あるとされるが 1 粒子として表されている。

[9] 本論文執筆時点では、graviton の存在は確認されていない。

表 8: 実体粒子の抽出

行番号	表示
0	./tq.o in=standard-model-net.t -Pin -Fst | grep –text C=0 | grep –text Ref=0
1	:SN=3: ... 中略... :H=E:
2	:SN=4: ... 中略... :H=M:
-	... 中略...
17	:SN=29: ... 中略... :H=#6H:

表 9: グルーピング項の抽出

行番号	表示
0	./tq.o in=standard-model-net.t -Pin -Fst | grep –text SN | grep –text -v C=0 | grep –text -v '\$'
1	:SN=0: ... 中略... :H=::
2	:SN=1: ... 中略... :H=#1leptions:
3	:SN=2: ... 中略... :H=#7:
4	:SN=9: ... 中略... :H=#2quarks:
5	:SN=16: ... 中略... :H=#3weak bosons:

グルーピング項の中でも、ラベル付与されている、leptons、quarks、weak bosons は粒子グループである。

ここで、この例が、本論文の目的で述べた均質化 2 ブラフの特徴、多項性、双対性（均質化）、入れ子（内部構造）を表す例になっているか検証する。多項性は、ハイパーエッジ表現により自明である。双対性（均質化）の端的な例は、"$#4(#4)"により表現されており、オブジェクト"#4"は、$V \wedge E$ であり、均質化されている。同様の構造は、"#3","#5","#6"において見られる。この例では、均質化されているオブジェクトすべてが内部（入れ子）構造を持っている。ちなみに、内部構造を持たないが $V \wedge E$ であるオブジェクトの表現例は"(#1,$#1())"のように、ダミーノードを用いて表現できる。タイプが V（のみ）であるオブジェクトは、レプトン、クオークの各粒子である。タイプが E（のみ）であるオブジェクトはこの例には存在しない。

表 10: 自己相互作用する粒子の抽出

行番号	表示
0	./tq.o in=standard-model-net.t -Pin -FT -Fh | grep './tq.o in=standard-model-net.t -Pin -FMa | ./extract_self-ref.pl | ./extract_term-no.pl'
1	+16:#3weak bosons:(
2	-21:#4g:,
3	-29:#6H:,

4.2 構造化データの生成と外部ソルバーへのエクスポート

構造化データの生成　前節までにおいては、均質化 2 部グラフの構造の生成について述べた。この節では、tq の構造生成機能を利用した、データ構造の生成について述べる。

ここでの基本的な考えは、T 式を派生させ、純粋な木（テンソル）構造としてのデータ（構造）、データの外部知識として与えられる意味構造（均質化 2 部グラフ）に分離するものである。この節に該当するのは前者であり、前者においてはさらに構造とデータ要素とを分離し、構造にデータ要素をバインドするというアイデアに基く。後者については次次節で述べる。

表 11: データバインドのための構造体定義

```
struct tree{
        char* Head;
        struct tree* Ref;
        struct tree** Child;
        char* Dim;
        char* DataStr;
}
```

このための準備として、構造体を表 11 のように拡張する。ここで、構造体メンバー Dim には、データの次元情報がテキストとして保存される。構造体メンバー DataStr には、データ要素の並びがテキストとして保存される。

かつ、唯一の関数であるデータの内積を生成する PIを導入し（理由は後述する）、言語仕様を表 16

のように拡張する。

さらに、プログラムとしての tq を拡張し、次の 3 形式の入力を許す : (1) データ、(2) 入力データ構造定義式、(3) 出力データ構造定義式。データ構造化は (2)、(3) の T 式において行われ、これらはデータのメタデータとして必要である。なお、この時点で意味構造は与えられていない。データはすべて 1 行のデータ要素の並びとして解釈される。(3) においてのみ、唯一の関数PIの使用が許される。また、データバインドはリーフノードに対してのみ行えるものとする。

このコンセプトにより、データ次元情報までを含めた知識情報構造の概念と、それに対応する実データとを、明確に区別しつつリンクすることが可能となる。つまり、データ利用における概念表現とデータ表現の曖昧性を形式的には完全に排除可能であるが、このためにはデータ形式に対するオーソライズを明確にする必要がある。オーソライズとは、T 式のインタープリターにはない解釈の拡張を外部システムにおいて行うことであり、この場合はラベリングの解釈を追加する。このオーソライズを含めた出力のルールを、本節では仮に次のように決める : (1) 出力データ形式はサイクリック／再帰構造のない木構造とする、(2) オーソライズは出力構造定義にラベルとして記載する、(3) オーソライズラベル以降の木構造はオーソライズされたと見なす。なお、すでに説明してきたいくつかの T 式の判定基準もオーソライズであり、オーソライズは形式的に判定可能でなくてはいけない。

以上の定義を用いた CSV データの構造化の例を表 12 に示す。出力構造定義の”DataSet”ラベルにより、データ構造全体がオーソライズされていることを示している。また、”Quantity”のように、必要なラベルを挿入することもできるが、単なるラベルの追加とオーソライズを分けることはできない。

PIは、子要素の内積リストを、木構造を保持したまま生成する関数である。この場合、対象になる子要素は、”$#1”、”Quantity”、”$#4”、”$#2”である。子要素として最大のリスト長をもつものは、”$#4”（の参照解決）であり、他の子要素は最大リスト長に達するまで自身のリストでパディングされる。この機構により記述量を大きく削減できるため、特別に関数として導入した。

表では、出力構造定義 1 を用い、最終的に各セルには構造化されたデータを含むスプレッドシート形式のデータセットが生成されている。

表 12: データ構造化の例

タイプ	表現
入力構造定義	(#1[2],#2[2],#3[3](#4[2]))
出力構造定義 1	PIDataSet($#1,Quantity ($#4,$#2))
出力構造定義 2	PIDataSet($#1,Quantity ($#4,ToString($#2)))
入力データ	Length,Weight mm,kg 12,30 11,30 15,34
出力	DataSet(((Length,Quantity(12,mm)), (Weight,Quantity(30,kg))), ((Length,Quantity(11,mm)), (Weight,Quantity(30,kg))), ((Length,Quantity(15,mm)), (Weight,Quantity(34,kg))))

ソルバー形式への変換　前述の出力データは、ソルバーへの変換／エクスポートが容易に行える。データ変換とエクスポートのプロセスは、(1) データバインド、(2) 内積出力、(3) 変換とエクスポート、となる。出力構造定義 2 を用いて Wolfram Engine にインポートした例を表 13 に示す。問題なく解釈されていることが分かる。

T 式および Wolfram Language はどちらも S 式であるが、当然 Head の定義が異なる。T 式においては、括弧のみで木構造を記述／定義可能であるため、”DataSet”、”Quantity”、”ToString”のような、「Head の文字列」の挿入が可能であり、変換を著しく容易にしていることが分かる。

4.3　辞書記述

ここで、T 式を利用した辞書の記述を提案する。辞書の形式は、図 4 に示すような、(1) 語彙エントリ、(2) 語彙属性情報、を持つ、いわゆる、一般的な辞書形式である。当該の辞書例のページは生命科

表 13: 構造化データのエクスポートの例

タイプ	表現
コマンド	./tq.o in=mm-kg.ddf out=mm-kg.ddl data=mm-kg.csv -Pprod -FT -C | ./tq.o in=/dev/stdin -FWW -Pin -C | math
出力	In[1]:= Out[1]= DataSet[{{Length, 12 millimeters}, {Weight, 30 kilograms}}, {{Length, 11 millimeters}, {Weight, 30 kilograms}}, {{Length, 15 millimeters}, {Weight, 34 kilograms}}]

学の専門用語を中心に収集されたライフサイエンス辞書（”Life Science Dictionary”）[17] の 1 ページである。

これに対応する T 式の表現の例を表 22 に示す。1 行目と 7 行目は木の頂点を作るためのものである。2 行目は語彙エントリであり、一つのリストになっている。3 行目以降は、語彙属性情報の羅列である。語彙属性情報も語彙間の関係として記述される。関係を示す語彙そのもの（「概念ツリー」など）も語彙リストに登録されるので、参照項でも被参照項でもないものは即値であることが明らかである。

語彙間の関係とは、タイプ E または $E \wedge V$ の語彙によるタイプ V または $V \wedge E$ の語彙に対する説明である。ハイパーエッジを扱えるので、原則としてトリプル（*Subject-Predicate-Object*）で表現する必要はなくなる。たとえば、3 行目は「1-ホスファチジルイノシトール-3-キナーゼ」を含む「同義語群」が定義されている。仮にトリプルで記述された場合の表現は、$P(S,O)$ であり、S と O の木構造を妨げない。

以上の記述規則により、語彙の表示は単なるラベルとなり、完全に形式的に語彙を定義可能であり、仮に説明として自然言語記述があったとしても、その記述が即値であるのか、辞書記述による定義であるのかが明確となる。

さらには、このような言語特性により、異なる分野の知識も容易に辞書に追加可能である。たとえば、前述の標準模型を辞書に追加するには、(1) 標準模型に名前を付け、粒子や模型を表す語彙をリストにエントリする、(2) 参照番号が競合しないように付与し直す、(3) 模型全体を表す T 式を辞書に追加する、だけでよく、標準模型の木構造を書き直す必要はない。ただし、粒子の検索方法が前述とは異なる。以上の処理はオーソライズの書き換えと見なせる。

ここで、辞書におけるオーソライズ（のレベル）について整理しておく。前述ではラベル付与をオーソライズと見做した。一方、辞書の語彙登録も一種のオーソライズと見なせる。そこで、オーソライズのレベルを次のように定める：(1) 登録語彙、(2) 登録語彙の参照、(3) 登録されないラベル、(4) 登録されない項の参照、の順で強いオーソライズとなり、(3) は即値、(4) は変数と見なせる。なお、辞書構造全体をオーソライズするには、辞書を表す語彙を語彙リストに登録し、参照により木全体をオーソライズする。表 22 の例では、辞書構造全体が”$#0”によりオーソライズされている。”$#1”による語彙リストのオーソライズは必ずしも必要ではない。

この仕組みにより、粒子間相互作用の具体的な関係（ラグランジアン [18]）等も記述可能となる[10]。

4.4 データフェデレーションフレームワークの提案

ここでは、T 式の特性を利用したデータフェデレーションフレームワーク構築の見通しを述べる。

まず、データセットの形式をあらためて表 17 のように定義する[11]。あらためて述べるが、これは、tq による出力形式として期待されるものである。このとき、 <item>、 <type>、 <unit> は、辞書により定義されているとする（すなわち解決可能な項参照 がなされているとする）。さらに、tq を拡張し、(4) 辞書を読み込み可能とする。これにより、インタープリターによって、(1) 非構造化データ、(2) 入力形式定義、(3) 出力形式定義、によって構成された構造化データセットの各項目と登録語彙とが参照解決され、さらに辞書記述により語彙間の関係が項目間の関係として決定する。<item> 間に変換が可能な場合、パターンマッチにより形式的に変換ルールを記述可能である。表 14 は変換ルールの記述例である。”$#1002”によりパターンマッチ指示がなされており、このとき、”#2001”と”$#2001”の参

[10] ただし、辞書記述は人間（または何らかの知能）が行う必要がある。たとえば $\frac{d^2}{d\xi^2}$ という関係は、2 が即値、ξ が変数、式全体は微分を表す語彙としてオーソライズされるであろう。

[11] ただし、データバインドによる <value> の T 式表現には制限がある。

照関係は、2 行目の語彙エントリに被参照項が無いことから、3 行目の木における局所参照関係（変数）であることが分かる（ただし、参照番号は前述したとおりユニークである必要がある）[12]。ただし局所参照は外部システムに渡す前に、あるいは外部システム側で、当該システムが受け入れ可能なパターン参照表現に置き換える必要がある[13]。

表 14: 変換ルール記述の例

行番号	T 式
1	$#0(
2	$#1(... 中 略...,$#1001Convert, $#1002Pattern,... 中略...),
3	$#1001(($#501,$#502($#1002(#2001,), $#503)), ($#501,$#502(Times($#2001, 1000),$#504))),
-	... 中略...,
5	)

以上に示した仕組みにより、図 5 に示すような、機械可読性の高い標準的なデータセット構造とそのメタデータ参照によるデータフェデレーションシステムが構築可能である。当該システムはデータリポジトリとソルバーの組み合わせで動作し、リポジトリからは表 17 に示すスプレッドシート形式のデータセット、スプレッドシートに使われる項目間の関係や変換規則を含めた辞書、およびこれらの形式解釈器 tq が提供される。すでに示してきたように、スプレッドシートおよび辞書は T 式で記述されており、統合的解釈が可能である。tq と、ソルバー形式への変換器により、ソルバー側でデータの変換・統合・解析が可能となる。

5 まとめ

技術的展望 本論文で提案した言語について、技術的な展望をまとめる。

提案された言語とそのインタープリターは、均質化 2 部グラフを解釈・ストア・再表示できることが示された。この点において、既存言語にはない機能の実装に成功したと言える。すなわち、均質化 2 部グラフを構築・解釈する基盤技術が示されたものであり、今後のグラフ様構造の解析における実用的な発展が期待される。

この技術の応用範囲は広く、たとえば、項参照機能を応用すれば構文解析器における代名詞表示の曖昧性等を排除できる。

式の解釈を変更すれば、化合物構造表示にも対応可能であり、環とポリマーを、項参照と木参照により区別可能である（表 18）。表 19 にいくつかの化合物構造表示例を示す。この例の T 式の解釈は、本論文で提案してきた知識構造を表す T 式の解釈とは異なる。すなわち、化合物の構造は均質化 2 部グラフとしてではなくグラフとして表されるので、オブジェクトはすべて V であり、子要素と親要素の間がエッジとなり、これが化学結合（の情報）として解釈される。表の定義では、子要素の括弧 [] 内の記述が当該子要素とその親要素の結合情報となっている。SMILES や InChi と比較して、より直接的に化学構造を表現可能である。

知識情報構造の解析あるいは解釈についての展望 tq の表現に着目し、構造解釈や解析の容易性について展望する。

tq が出力する形式の一つである T 式の特徴として、項のタイプを明確に V、E、$V \wedge E$ に決定できることが挙げられる。その判定基準は項の子要素の存在をもとに判断しているため、ネットワーク解析に依らない文字列マッチによりタイプ判別が可能であり、情報モデルにおいて問題となる実体／属性判別も、実体は V、属性は E、実体かつ属性は $V \wedge E$ として記述されていれば、外部システムによる文字列の抽出と判別によりどのタイプかが決定する。

ここで注目すべきは、複雑なアルゴリズムを用いることなく、表現の工夫により処理コストを大幅に下げられる可能性である。同一の構造に対して複数の表現が存在することは、解析の緒を広げることであり、（内部）構造の解析を表現の解析に置き換えられる可能性を高めるものであるが、多くのアプリケーションはこのような思想で作成されていない。このことは、コンセプト実証で述べたオーソライズに関わる。ここでのオーソライズは形式的に行える

[12] #2001 の上位にパターンマッチを明示する項（$#1002）が必要な理由は、マッチ条件を指定可能とするためである。ここでは空となっており、無条件で対応する項が変数$#2001 によって参照される。

[13] この仕組みは当然ながら対応するシステムごとに異なり、T 式のインタープリター側でなく外部システム側に実装する必要がある。前述の表 13 の例ではコンセプト実証のために特別に tq の機能として実装した。

ことを前提としており、実装技術的にはAPI定義である。すなわち、システムの出力表現はすべてAPIであり、この観点からは、その差は逆説的に、形式的解釈がどの程度容易であるかでしかない。

以上の例と説明によれば、均質化2部グラフを構成するツールや、解析するツールが存在しなかったのではなく、従来の表現のバラエティが乏しかったことにより形式的解釈が困難であったことを示している。これは、知識情報構造の解釈／解析の要点は、その形式記述（表現論）にあるということである。

サービスへの展望　前章ですでに述べたように、tqはサービスフレームワークとしての可能性を持っている。提案したようなサービスはすでにWolfram Data Repository[19]等により行われており、新奇性のあるものではないが、フレームワークとしてのオープン性、可用性はより高い。

また、tqでは、意味解析（関数機能）を可能な限り排除したため[14]、ソルバーと知識情報構造との分離が明確であり、結果としてソルバーの選択性が向上する。さらに、T式の表現力により、データ構造だけでなく関数構造をも表現可能であり、従来のデータインテグレーションを予め決められた定義によるデータ統合だとすれば、動的に統合ルールを適応できるデータフェデレーションの実現が見えてくる。このことは、gitlab[20]等によるリポジトリサービスとdocker[21]等によるソルバーの統合運用をイメージすれば理解しやすい。このようなシステムは、ほとんど全ての機能のコントロールがユーザーにあるため、tqをベースとした解釈・解析系を自由に構築可能である。

スコープ　ここで、以上説明してきたコンセプトや技術について、スコープをまとめる。このためには、議論されていないテーマを列挙するのが良いであろう。

均質化2部グラフを表現可能な言語とその構造を解釈可能なインタープリターを実装した。ただし、

- ダイナミズム（時間軸）に対応していない、
- （コンセプトとしても）有限なオブジェクトしか扱えない、
- アクセシビリティ（アクセスコントロール）が考慮されていない、
- 差分記述が考慮されていない。

実用に向けて　前述のスコープのうち、ダイナミズム、アクセシビリティ、差分記述に関しては外部システムにより実現可能である。一方、無限のオブジェクトの取扱いについてはそもそも不可能なので、議論の範囲外とする（量子計算的には可能性がある）。なお、数学的な定義をもつ無限の概念は、一つの語彙として取り扱いが可能である。

最後に、コンセプトを実用化する上で問題になりがちなパフォーマンスについて述べておく。オリジナルtqはコンセプトを忠実に再現するようなコーディングがなされているため、構造体メンバーの整理やデータ／関数呼出に関しての改良を行ったブランチを作成し、オリジナルと改良版でパフォーマンスを測定した。測定はパーシングのみである。比較として、T式と完全に同構造のJSONを作成し、jqとpythonに対しても同じ測定を行った。

表20に示すように、パフォーマンス改良版では、既存技術にくらべても非常に高いパフォーマンスを示しており、実用化技術としてもすでに有力な選択肢となっている。参考として、表21には変換を含めた測定結果を示した。JSONに変換する時間を考慮しても、そのパフォーマンスは高い。

[14] 唯一の例外がPIである。

謝辞

本研究は、情報・システム研究機構 国立遺伝学研究所が有する遺伝研スーパーコンピュータシステムを利用しました。

本研究は、物質・材料研究機構が有するMDPF解析クラスタを利用しました。

参考文献

[1] 高橋要: 「内包論理の限界とその新たな構成可能性について」, 東北哲学会年報, Vol. 4, , 1988.

[2] Barwise, Jon: “Situations, Sets and the Axiom of Foundation”. “*The Situation in Logic, CSLI Lecture Notes Number 17*”. pp. 177–200. Stanford University, 1998.

[3] Devlin, Keith: “Logic and information”. Cambridge University Press, 1991.

[4] 向井国昭: 「談話理解とロジック」, 人工知能学会誌, Vol. 3, No. 3, pp. 43–54, 1988.

[5] 向井国昭: 「状況理論の数学的基礎」, 人工知能学会誌, Vol. 7, No. 3, pp. 26–33, 1992.

[6] 藤原譲: 「情報学基礎論の現状と展望」, 情報知識学会誌, Vol. 9, No. 1, pp. 13–37, 1999.

[7] Fujiwara, Yuzuru: "The Model for Self Strucured Semantic Relationship of Information and Its Advanced Utlization", *International Forum on Information and Documentation*, Vol. 19, No. 2, pp. 8–10, 1994.

[8] Fujiwara, Yuzuru; Liu, Ye: "The Homoginized Bipartite Model for Self Organization of Knowledge and Information", *The Proceedings of The 48th FID Conference and Congress*, Vol. 2, No. 1, pp. 13–17, 1998.

[9] Fujiwara, Yuzuru: "The Hyper Brain Knowledge Infrastructure Based on The Semantic Analysis of Terms", *2EafTerm Conference*, 1998.

[10] Nayes, Jonathan; Gutierrez, Claudio: "Bipartite Graphs as Intermediate Model for RDF". https://www.researchgate.net/publication/225265076_Bipartite_Graphs_as_Intermediate_Model_for_RDF （2020-01-20 参照）.

[11] Munshi, Shiladitya; Chakraborty, Ayan; Mukhopadhyay, Debajyoti: "Integrating RDF into Hypergraph-Graph (HG(2)) Data Structure", *arXiv.org*, 2013.

[12] MapleSoft: "Maple". https://www.maplesoft.com/products/Maple/（2020-01-27 参照）.

[13] Wolfram: "Wolfram Language". https://www.wolfram.com/language/（2020-01-27 参照）.

[14] CMUCL: "CMUCL". https://www.cons.org/cmucl/ （2020-01-27 参照）.

[15] kouamano: "RECURSIVE-SISTEM". https://github.com/kouamano/RECURSIVE-SYSTEM （2020-01-20 参照）.

[16] Wikipedia: "Standard model - Wikipedia". http://en.wikipedia.org/wiki/Standard_Model（2020-01-20 参照）.

[17] LSD プロジェクト: 「ライフサイエンス辞書」. https://lsd-project.jp/cgi-bin/lsdproj/draw_tree.pl （2020-01-20 参照）.

[18] FeynRules. https://feynrules.irmp.ucl.ac.be/（2020-01-31 参照）.

[19] Wolfram-Research: "Wolfram Data Repository". https://datarepository.wolframcloud.com/（2020-01-24 参照）.

[20] GitLab: "GitLab Instlattion". https://about.gitlab.com/install/ （2020-02-05 参照）.

[21] docker.com: "DockerHub". https://hub.docker.com/ （2020-02-05 参照）.

表 15: 言語定義

LANGUAGE :: <T-form>;
<head>::=<ref-label>?<reference>?<label>?;
<ref-label>::=('#'|'##')<num>+;
<reference>::='$'('#'|'##')<num>+;
<label>::=<char>+;
<T-form>::=<head>('('(<T-form>(','<T-form>)*)?')')*;

<char> : '(', ')' などいくつかの文字を除く任意文字。

表 16: データバインド用に拡張された言語定義

LANGUAGE :: <T-form>;
<head>::=<ref-label>?<reference>?<operator>?<label>?<bind>?;
<ref-label>::=('#'|'##')<num>+;
<reference>::='$'('#'|'##')<num>+;
<operator>::='$'<label>'$';
<label>::=<char>+;
<bind>::='['(<num>+(,<num>+)*)?']';
<T-form>::=<head>('('(<T-form>(','<T-form>)*)?')')*;

<operator> : PIのみ機能定義されているがそれ以外も acceptable である。

表 17: データセット形式の定義

<detaset>::=DataSet'('<line>(','<line>)*')';
<line>::='('<cell>(','<cell>)*')';
<cell>::='('<item>','<type>'('<value>(','<unit>)?')")';
<value>::=<T-form>;

<T-form> : 表 15 による定義。

表 18: 化合物記述用に拡張された言語定義

LANGUAGE :: <T-form>;
<head>::=<ref-label>?<reference>?<label>;
<ref-label>::=('#'|'##')<num>+;
<reference>::='$'('#'|'##')<num>+;
<label>::=<char>+;
<bond>::='['<property>(','<property>)*']';
<property>::=<num-alpha-ext>+;
<T-form>::=<head><bond>?('('<T-form>(','<T-form>)*')')?;

<num-alpha-ext> : 数字、アルファベットに加え、電荷や結合種類を表すいくつかの文字。

表 19: T 式による化合物表現

化合物名	タイプ	式
ion	T 式	H[+](O[.,-](H))
Benzene	SMILES	C1=CC=CC=C1
Benzene	T 式	#1C[:,2](C[:](C[:,2](C[:](C[:,2](C[:]($#1))))))
trans-1,2-Difluoroethylene	SMILES	F/C=C/F
trans-1,2-Difluoroethylene	T 式	C(F,H,C[2](H,F)) | C(F,H,C[2,trans](H,F))
poly-phenyl	T 式	##1C[:,2](C[:](C[:,2](C[:](C[:,2](C[:]($#1)),R($##1)))))

Ion charge : + / -
Ionic bond : .
Aromatic bond : :

表 20: tq のパーシングパフォーマンス

Program	Size (nodes)	Time (min:sec)	Memory (bytes)
tq (stable)	124,653,854	45	26G
tq (exptl.)	124,653,854	27	12G
jq	124,653,854	2:25	36G
python	124,653,854	51	12G
tq (stable)	498,615,417	3:30	104G
tq (exptl.)	498,615,417	1:55	51G
jq	498,615,417	10:50	144G
python	498,615,417	4:07	53G

表 21: tq のパーシングパフォーマンス（変換含む）

Output form	Size (nodes)	Time (min:sec)	Memory (bytes)
JSON	124,653,854	1:21	29G
JSON	498,615,417	5:35	136G

表 22: T 式によるライフサイエンス辞書の形式表現の例

行番号	T 式
1	$#0(
2	$#1(#0 The Dictionary,#1 The Dictionary Terms,... 中略... ,#10 同義語,#11 概念ツリー,#12 解説語,#13 関連語, ... 中略... ,#101 1-ホスファチジルイノシトール-3-キナーゼ, ... 中略...),
3	$#10($#101,$#102,$#103, ... 中略...),
4	$#11($#201($#202(... 中略... ($#101(... 中略...))))),
5	$#12($#101,(https://www.google.co.jp/search ... 中略...)),
6	$#13($#251,$#252, ... 中略...),
-	... 中略...
8	)

1-ホスファチジルイノシトール-3-キナーゼ　1-phosphatidylinositol 3-kinase

PubMed, Scholar, Google, Wikipedia

同義語（異表記）：

- 1-Phosphatidylinositol 3-Kinase
- 1-ホスファチジルイノシトール-3-キナーゼ
- Phosphatidylinositol 3-Kinase
- Phosphoinositide 3 Kinase

概念ツリー：

- 酵素および補酵素　enzyme and coenzyme
  - 酵素　enzyme
    - 転移酵素　transferase
      - リン酸転移酵素　phosphotransferase
        - アルコール基へのリン酸転移酵素　phosphotransferase (alcohol group acceptor)
          - ホスファチジルイノシトール-3-キナーゼ　phosphatidylinositol 3-kinase
            - 1-ホスファチジルイノシトール-3-キナーゼ　1-phosphatidylinositol 3-kinase
              - クラスIホスファチジルイノシトール-3-キナーゼ　class I phosphatidylinositol 3-kinase +
              - クラスIIホスファチジルイノシトール-3-キナーゼ　class II phosphatidylinositol 3-kinase
              - クラスIIIホスファチジルイノシトール-3-キナーゼ　class III phosphatidylinositol 3-kinase +

解説語：

- ホスファチジルイノシトール-3-キナーゼ (phosphatidylinositol 3-kinase)

関連語：

- 分裂促進因子 (mitogen)
- ホスホエノールピルビン酸カルボキシキナーゼ (GTP-dependent phosphoenolpyruvate carboxykinase)
- ホスファチジルイノシトール-3-キナーゼ (phosphatidylinositol 3-kinase)
- LY294002 (LY294002)
- アドレナリン受容体 (adrenergic receptor)
- インスリン受容体基質タンパク質 (insulin receptor substrate protein)
- インスリン受容体 (insulin receptor)
- ホルボールエステル (phorbol ester)

図 4: ライフサイエンス辞書の表示画面

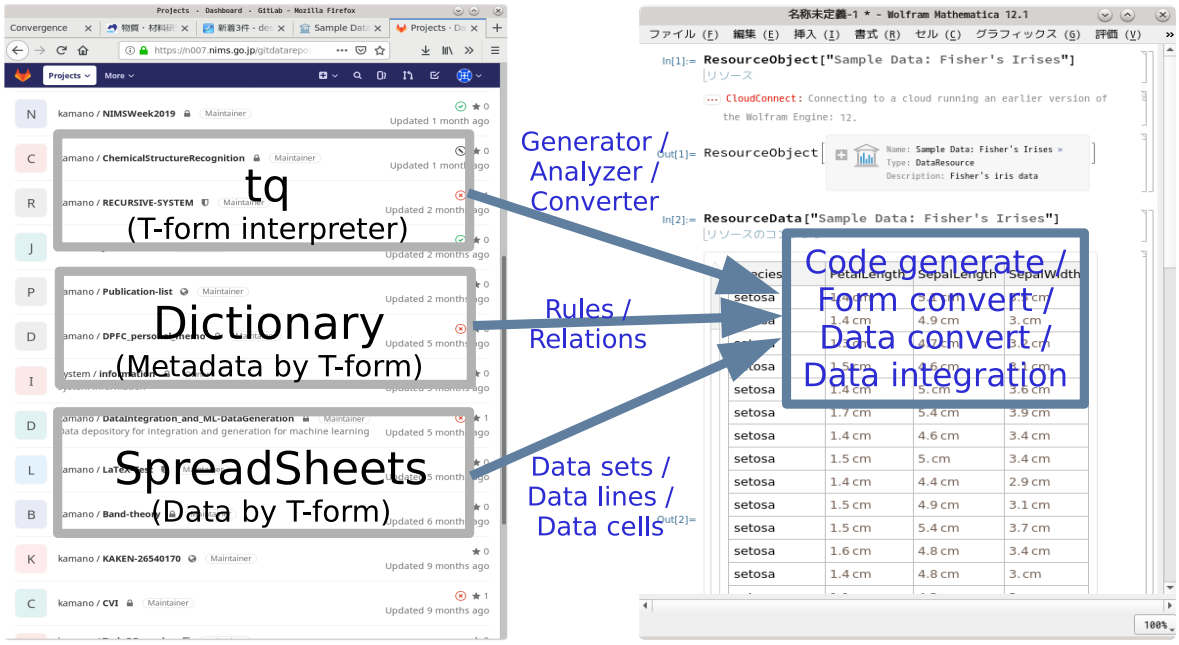

図 5: データフェデレーションの概念図. ここでは、辞書をメタデータとする。辞書とデータは T 式によって記述されている。データ項目間の関係は辞書に記述されており、これによりデータ（項目）間の変換規則を検索できる。変換規則もまた T 式として記述されていいるが、tq は実働する変換器を有しておらず、ソルバーが T 式による変換規則を解釈して変換を行う。ソルバーそのものがリポジトリより提供される（さらにリポジトリ上で動作する）場合もある。