事例／調査報告

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

[‡] 国立情報学研究所 オープンサイエンス基盤研究センター

[‡] Research Center for Open Science and Data Platform, National Institute of Informatics

〒 101-8430 千代田区一ツ橋 2-1-2

E-Mail: amano@nii.ac.jp

* 連絡先著者 Corresponding Author

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する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 部グラフ構造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、提案された仕組みによる応用例を示し、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する展望を述べる。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 部グラフ構造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、本論文は解説の要素が多いため、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、これらの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的特徴的な性質である双対性に着目した。双対性とは、同じ語が relation にも entity にもなりうる性質であり、本論文で後述するグラフ構造においてはオブジェクトがエッジにもバーテックスにもなりうる構造のことである。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、関係や状況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、より明確に双対性を意識しているといえる。その結果、状況理論と同様 non-well-founded^[4] (3.1 にて詳述する) 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うることを前提とし、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、パーサー機能を含む言語構造の解釈器をインタープリター、意味解釈器をソルバーと呼ぶ。

ここで、あらためて筆者らの主張をまとめると、主な主張は2点である：(1) 知識情報構造や意味処理の従来の議論においては、「解釈」の表す概念が広すぎるため、図1のように「解釈」を「分解」した (本章の第一段落)。(2) この、より詳細化された「解釈」の概念を用いた議論により、新しいタイプのインタープリターを提案する (本論文の以降の議論)。

新しく提案するインタープリターは、言語パーサーに相当し、後述する項 (木) 参照機能により、パース終了後の言語構造 (データ構造) に、均質化2部グラフの構成に必要な「多項性」「双対性」「入れ子」の要素を持たせることが可能である。以降、本論文での議論は均質化2部グラフの形式記述とインタープリターについてのもの (図1 グレー枠内) が中心となる。なお、図ではタスクを人間のものと計算機のものに分けてはいるが、異なる場合もありうる。また、均質化2部グラフの理解の鍵となる具体例は4.1章であらためて解説する。

また、本論文は、大型の図表 (表15以降および図3以降) を付録とする構成とした。

図1: 知識情報構造の解釈。実線矢印が本論文で想定するフローだが、波線矢印のフローもありうる。

2 目的

知識情報構造を構成可能であるとされる均質化2部グラフ構造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部グラフ構造で知識情報構造を構成可能であることを前提とし、以下の均質化2部グラフ構造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]。

$$E \subseteq 2^V \quad (1)$$

$$V = V \cup E \quad (2)$$

$$E = E \cup V \quad (3)$$

$$\sigma : L \rightarrow E \cup V \quad (4)$$

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うる。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 、 $V \wedge E$ で有りうる、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 に内部構造を持ちうる。（ E の内部構造は自明）

これらが満たされることにより均質化2部グラフの構成が可能となる。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構造の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 Engine^[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部グラフ構造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（技術的には難しくはないはずであるが、愚直にこれを行えば表示量が膨大になるために

弊害が生じる）、結合隣接行列のみでは困難である。また、前述の non-well-founded な集合の典型として、 $x = \{a, x\}$ のような再帰的な定義も扱える必要があるが、十分には扱えていない（再帰的な定義は可能であるが、その定義を確認できないなど）。

このように、均質化2部グラフを表現する上で、既存言語が十分に扱えていない要素には以下が考えられる：

- 言語の内部構造を表示できない：図1に照らせば、言語構造（内部構造）を表示できない。言語構造の確認には、式を構成する言語表現とは別に言語がパースされた後の単なる文字列ではない内部構造が表現される必要があるが、これが行えず言語構造の確認ができない。
- 代入とは異なる仕組みで項（の情報）を表示する必要があるが、これが行われていない：前述の再帰的な定義等により代入値を評価できない構造が構成されている場合は、これを確認する別の手段（本論文で参照と呼ぶもの）が必要である。
- 言語の内部構造へのマップと、内部構造に対する意味処理との分離が行われていない：図1に照らせば、言語構造が（表示できないのではなく）保持されていない（意味処理後の構造しか保持されていない）。形式記述の言語構造解析から意味処理までの工程において処理の分離が曖昧なために、言語構造や還元されたモデルを表現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：有限であるがサイクリックである構造と、実際に無限にコピーが再帰される構造の分離が必要である（たとえば化合物における環とポリマーの表記分離）。

これらの機能が満たされることにより、前述の「多項性」、「双対性」、「入れ子」が表現可能となる。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：

- 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機能をインタープリターに組み込む。
- 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たす言語を定義、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。前述の諸問題を解決する方法として、参照の機能と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。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、項参照は成功し、木参照は失敗する。

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。

加えて、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の定義を再掲する：

- 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付録表16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）。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;
}
```

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。

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ケ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
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（前節）とそのインタープリター（本節）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、以下の要件を満たす言語 tq を実装した^[14]。

- インタープリターにおいて参照解決が行われること（次節で検証する）。
- 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（次節で検証する）。
- T 式またはインタープリターの出力が、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（次節で検証する）。
- T 式またはインタープリターの出力が、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である）。

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（ただし、現実装では禁則処理は行っていない）、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。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ない。

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

式5の表現は、頭部のノードを子ノードが参照する、自身の構造を内包する一つのオブジェクトとして構成されている。以下はこの表現および構造を検証の対象とする。

$$\#1(\$ \#1, \$ \#1, \$ \#1) \quad (5)$$

[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。

[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。

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.

オブジェクト判定

いくつかのT式とオブジェクトタイプの判別を表6に例示する。1行目、3行目、6行目が、それぞれ

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:

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（ただし、オブジェクトでない項もある）。対角要素には項情報が表示される。

V、E、 $V \wedge E$ のタイプとして判別される典型的な表現である。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、カッコ()内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という定義を利用することにより、このような表現が可能となる。tqはHeadが空であってもカッコ()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なさない。

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。

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。

図 2: 標準模型の模式図

以上、均質化 2 部グラフを表現する機能を備えた言語を実装した。次章以降は、実際にこの言語により知識情報構造の表現と解析が可能かを実証する。

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^[15]を利用する。

図 2 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。なお、この解釈は筆者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。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。

(1) 実体粒子の抽出: 総称は実体と見做さない、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。

表 7: tq による標準模型の形式表現

行番号	T 式
1	(
2	#1leptons(#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	)

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。
gluon は 8 色あるとされるが 1 粒子として表されている。
本論文執筆時点では、graviton の存在は確認されていない。

[手作業による確認] 表 7 の各行より、リーフノードであり参照項でない「粒子」を抽出すれば良い。2 行目より、E、M、T、nE、nM、nT、3 行目より、u、d、c、s、t、b、4 行目より、#8W、Z、6 行目より、#4g、7 行目より、#5p、9 行目より、#6H が抽出される。

[コマンドによる確認] 実処理の例を表 8 に示す。0 行目がコマンドである。構造体情報を出力し (最初のパイプまでのコマンド)、子ノードを持たない条件に相当する文字列 "C=0" が含まれ (最初のパイ

ブ以降のコマンド)、かつ参照項ではない条件に相当する文字列"Ref=0"が含まれる(2段目のパイプ以降のコマンド)行を抽出し、頭部"H=" < 粒子名 >を確認することにより粒子の一覧が得られる。均質化2部グラフの構造に対応させれば、参照項ではない V または $V \wedge E$ であるオブジェクトの抽出である。表には省略しているが、図2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。

なお、"standard-model-net.t"の内容は表7と同等である。また、表8~11、および表14のコマンド行は一般的なUNIXコマンドとして実行可能であることを確認している。

(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。

[手作業による確認] 表7の各行より、当該条件にあてはまる「項」を選択すればよい。1行目より、ダミーノード、2行目より、#1leptons、#7、3行目より、#2quarks、4行目より、#3weak bosons、が抽出される。

[コマンドによる確認] 実処理の例を表9に示す。0行目がコマンドである。構造体情報を出力し(最初のパイプまでのコマンド)、子要素を持たない条件に相当する文字列"C=0"を含まず(最初のパイプ以降のコマンド)、参照項でない条件に相当する文字列"Ref=0"を含む(2段目のパイプ以降のコマンド)、行を抽出することにより項の一覧が得られる。均質化2部グラフの構造に対応させれば、参照項ではない E であるオブジェクトの抽出である。非オブジェクト項も抽出される可能性があり、例では実際に抽出されている。木を構成するためのダミーノードである第0項がこれにあたる。粒子としてはleptons、quarks、weak bosons、が抽出されている。なお、荷電レプトンのグルーピングに利用した"#7"は命名されていないがオブジェクトである。

(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。

[手作業による確認] (2)で抽出された項のうちラベル付与されているものは、#1leptons、#2quarks、#3weak bosonsである。

[コマンドによる確認] (2)のコマンドにさらに"`grep -E 'H=.*[A-Za-z]:'`"を追加すればよい。この例の場合、ラベルはアルファベットで構成されるため、参照表現の有無にかかわらず、頭部にアルファベットを含むものがラベルされたオブジェクトとなる。均質化2部グラフの構造に対応させれば、ラベル付与された参照項ではない E であるオブジェクトの抽出である。実処理の例を表10に示す。正しく、leptons、quarks、weak bosons、が抽出されている。

(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(自身の直下に自身のオブジェクトを持つ)項を抽出すればよい。

[手作業による確認] 表7の各行より、自身へのまたは自身よりの参照を子要素に持つ項を選択すればよい。5行目より、#3、6行目より、#4(=#4g)、9行目より、#6(=#6H)であるが、#3は#3weak bosonsであるので、粒子としては、#3weak bosons、#4g、#6Hである。

[コマンドによる確認] 実処理の例を表11に示す。0行目がコマンドである。隣接行列より、自身の項の子要素に自身の項番号が表示されているものをラベル付与されているものを選択すればよい。この抽出はルールは明確だが、ラベル付与された項を抽出するには参照を辿る必要があり、操作はそれほど自明ではなく、短いスクリプト^[16, 17]を書く必要があった。スクリプトの記述は自明ではないので、以下に処理の解説を行う。まず、最初のパイプのgrep直後"`./tq.o in=standard-model-net.t -Pin -FMa`"により隣接行列を出力する。次に、スクリプト"`extract_self-ref.pl`"により自分自身の項番号が同一行にある行を抽出する。この時点で参照項から被参照項の番号が得られる(表11 1~3行目)。次に、スクリプト"`extract_term-no.pl`"により当該項番号のリストを作成する。この時点ではまだ粒子名が得られていないため、あらためて項番号により式(最初のパイプまでのコマンド：`./tq.o in=standard-model-net.t -Pin -FT -Fh`)の項を検索(最初のパイプの後のgrep)する。均質化2部グラフの構造に対応させれば、ラベル付与された $E \wedge V$ が自身のみで構成されるオブジェクトの抽出である。正しく、weak bosons、g、Hが抽出されている(表11 4~6行目)。

ここで、以上の例が、本論文の目的で述べた均質化2部グラフの特徴、多項性、双対性(均質化)、入

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 C=0 grep Ref=0
1	:SN=3: ... 中略... :H=E:
2	:SN=4: ... 中略... :H=M:
-	... 中略...
17	:SN=32: ... 中略... :H=#6H:

standard-model-net.t : 表 7 と同等のテキスト。

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 -v C=0 grep Ref=0
1	:SN=0: ... 中略... :H=::
2	:SN=1: ... 中略... :H=#1leptons:
3	:SN=2: ... 中略... :H=#7:
4	:SN=9: ... 中略... :H=#2quarks:
5	:SN=16: ... 中略... :H=#3weak bosons:

standard-model-net.t : 表 7 と同等のテキスト。

れ子（内部構造）の構造を、知識情報に照らしてどのように表されているかを俯瞰する。

[多項性] 多項性は、ハイパーエッジ表現により自明である。 E を親とする構造、たとえば、表 7:3 行目”#2quarks(u,d,c,s,t,b)”が端的な例である。また、同時に当該行は、同表全体が表す式内の子要素でもある。

[双対性] 双対性による均質化の端的な例は、同表:5 行目”\$#3(\$#1,\$#2,\$#3)”により表現されており、オブジェクト”\$#3”は、自身で親子関係を構築し、子要素がリーフノードであることにより $V \wedge E$ のオブジェクトとなり均質化されている。”\$#3”がオブジェクトであることは、同表:4 行目の”#3weak bosons”の存在による。本論文での双対性の定義によれば、均質化が表現できるだけでなく、 V のみ、 E のみのオブジェクトも表現可能である必要がある。 V のみのオブジェクトについては、実体粒子のうちで媒介粒子ではない、 E 、 M 、 T 、 nE 、 nM 、 nT 、 u 、 b 、 c 、 s 、 t および媒介粒子の W 、 Z がこれにあたる。weak boson である W 、 Z が、 V のみの

表 10: 粒子グループの抽出

行番号	表示
0	./tq.o in=standard-model-net.t -Pin - Fst grep -v C=0 grep Ref=0 grep -E 'H=.*[A-Za-z]:'
1	:SN=1: ... 中略... :H=#1leptons:
2	:SN=9: ... 中略... :H=#2quarks:
3	:SN=16: ... 中略... :H=#3weak bosons:

standard-model-net.t : 表 7 と同等のテキスト。

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。

表 11: 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	[0=>:\$#3:19->16],:>16<
2	[0=>:\$#4:23->24],:>24<
3	[0=>:\$#6:31->32],:>32<
4	+16:#3weak bosons:(
5	-24:#4g:,
6	-32:#6H:,

standard-model-net.t : 表 7 と同等のテキスト。

オブジェクトとして表現されているのは、図 2 には W 、 Z による直接の相互作用が表されておらず、 T 式にはこれを忠実に反映したからである (p から W への連結は p の作用とした)。 E のみのオブジェクトについては、グルーピング項がこれにあたる。

[入れ子] 本論文における入れ子の定義を再掲すると、 $V \wedge E$ のオブジェクトが子要素を持つことである。前述の”\$#3(\$#1,\$#2,\$#3)”の例のように均質化されているオブジェクトがこれに当たるが（ただし、藤原はこのような自己参照構造を入れ子とは別に扱っている）、自身の参照による均質化構造を持つ必要はなく、たとえば photon は同表:7 行目”#5p”と、8 行目”\$#5(\$#2,\$#7,\$#8)”により、子要素を持つ $V \wedge E$ のオブジェクトとなっている。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。

以上、提案した言語において、知識情報構造の「多項性」、「双対性」、「入れ子」を表す記述が確認できた。

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

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、「オーソライズ」の概念を説明する。

ここでの基本的な考えは、データをデータのシークエンスとその構造に分離するものである。具体的には、データのシークエンスをファイルとして T 式に与えてバインドする仕組みを提案する。

表 12: 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 12 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部積（この処理は本論文で言う意味処理にあたる）を生成する \$PI\$ を導入し（理由は後述する）、言語仕様を付録表 17 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\$ の使用が許されこの関数の実行が意味処理に該当する。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解釈をさらに明確にする必要があり、オーソライズの概念を導入する。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、この場合はラベルに対しての解釈を追加する。つまり、T 式においてラベルは単なるラベルであるが、外部システムにおいてはラベルが意味を持つ（場合がある）。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、オーソライズは形式的に判定可能であることを前提とする。

以上の定義を用いた CSV データの構造化の例を表 13 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。

ソルバー形式への変換

表 13: 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	\$PI\$DataSet(\$#1,Quantity(\$#4,\$#2))
出力構造定義 2	\$PI\$DataSet(\$#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))))

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 14 に示す。問題なく処理されていることが分かる。

T 式および Wolfram Engine の言語はどちらも 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、付録図 3 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")^[18] の 1 ページである。

表 14: 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-2.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}}]

これに対応する T 式の表現の例を付録表 23 に示す。1 行目と 8 行目のカッコにより全体でひとつの木構造を作っている。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。

ここで、「辞書におけるオーソライズ」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す例を紹介したが、辞書の語彙登録も別のオーソライズの例であり、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。付録表 23 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^[19]）等も記述可能となる。ただし、これらの記述は人間（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解釈し、そのようにオーソライズされるように記述する必要がある。

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付録表 18 のように定義する。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 15 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）。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。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 14 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。

表 15: 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	)

#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。

この仕組みを用いることにより、推論や類推の規則も記述可能かもしれないが、本論文では形式の記述や変換に関する議論に留める。

以上に示した仕組みにより、付録図 4 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、リポジトリからは付録表 18 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

提案された言語とそのインタープリターは、均質化2部グラフ構造を解釈・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(付録表19)。付録表20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□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

T式の特徴として、オブジェクトのタイプを明確にV、E、 $V \wedge E$ に決定できる形式を持つことが挙げられる。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。

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、表現のバラエティが乏しいために、解析容易な表現が存在しなかったことを示している。こ

れは、知識情報構造の解釈や解析の要点は、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^[20]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。たとえば、tqでは、意味処理(関数機能)を可能な限り排除したため(唯一の例外が\$PI\$である)、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^[21]等によるリポジトリサービスとdocker^[22]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(厳密には式1を満たしていない)、
- 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

前述の本論文では実現されていない要素のうち、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困難であるが、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、パフォーマンスは考慮されていない。そのため、実用に向け構造体メンバーの整理やデータ／関数呼出に関して調整しパフォーマンスの改善を行った。付録表 21 はオリジナルおよび改良版 tq、python、jq のパフォーマンス計測結果である。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、付録表 22 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。

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, pp. 14–30, 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 Structured Semantic Relationship of Information and Its Advanced Utilization”, *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] Naves, 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 Engine”. <https://www.wolfram.com/engine/> (2020-01-27 参照) .
- [14] kouamano: “RECURSIVE-SISTEM”. <https://github.com/kouamano/RECURSIVE-SYSTEM> (2020-01-20 参照) .
- [15] Wikipedia: “Standard model - Wikipedia”. http://en.wikipedia.org/wiki/Standard_Model (2020-01-20 参照) .
- [16] kouamano. https://github.com/kouamano/RECURSIVE-SYSTEM/blob/jsik/extract_self-ref.pl (2020-02-05 参照) .
- [17] kouamano. https://github.com/kouamano/RECURSIVE-SYSTEM/blob/jsik/extract_term-no.pl (2020-02-05 参照) .

- [18] 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) .
- [19] FeynRules. <https://feynrules.irmp.ucl.ac.be/> (2020-01-31 参照) .
- [20] Wolfram-Research: “Wolfram Data Repository”. <https://datarepository.wolframcloud.com/> (2020-01-24 参照) .
- [21] GitLab: “GitLab Installation”. <https://about.gitlab.com/install/> (2020-02-05 参照) .
- [22] docker.com: “DockerHub”. <https://hub.docker.com/> (2020-02-05 参照) .

(2020年 2月20日 受付)

(2021年 1月12日 採択)

付録

表 16: 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 17: 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 18: データセット形式の定義

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

<T-form> : 表 16 による定義。

表 19: 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 20: 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 21: 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 22: 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 23: 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 3: ライフサイエンス辞書の表示画面

図 4: 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。