tq :
A Comprehensive Disciplinary Language for Materials Science

Kou Amano† Koichi Sakamoto†
†Natinal Institute for Materials Science

Introduction

Objective

tq should satisfy

- parsing tree structure
- parsing graph structure
- searching dictionary
- matching terms using dictionary
- reforming from unstructured data to structured data
- conversion to other well-known formats such as JSON
- matching or searching tree or graph structure
- Term Rewriting by Network Similarity (TRNS)
- daemonizing dictionary system
- parallelizing.

The language

Short example

```
#1$Op$Name($#1[1])
↓ tq in=/dev/stdin -FT -Pin data=test.csv
#1$Op$Name($#1[1]@@#1$Op$Name(Length))
```

#1 : < *label* >
Op : < *operator* >
Name : < *name* >
$#1 : < *reference* >
[1] : < *data bind dimension* >
@@ : < *bind mark* >
#1OpName : < *binded object* >
Length : < *binded data* >

Data structure

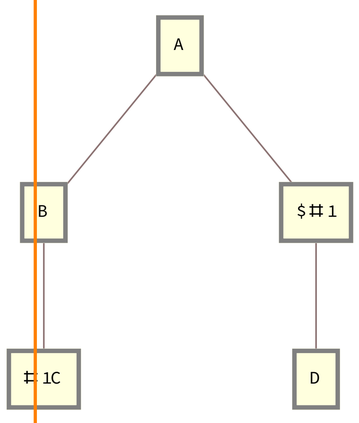
Table: Members of the data structure

	Lv	Adr	PAd	Ref	LT	LN	Hpt	H	D	VC	VSt	Cj	NC
0	0	14153344	0	0	h	1	2	#1OpName		0		0	1
1	1	14154608	14153344	14153344		-1	0	$#1[1]	[1	1	Length	0	0

Parsing

Parsing tree

Interpreted structure



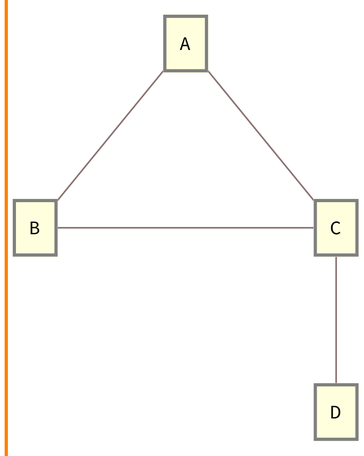
Statements

Input:
A(B(#1C),$#1(D))
Output:
A(B(#1C),$#1@#1C(D))

Parsing graph

Implicit graph

Interpreted structure

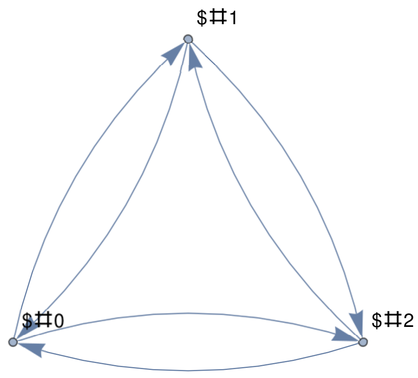


Statements

Input:
A(B(#1C),$#1(D))
Adjacency matrix:
[:A:0],1,2,3,,
,[:B:1],2,,,
,,[:#1C:2],,,
,,,[:$#1:3→2],4,
,,,,[:D:4],

Explicit graph

Interpreted structure

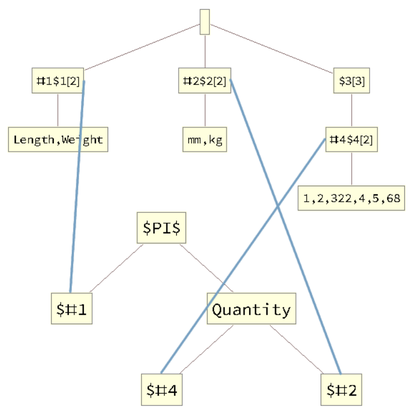


Statements

Input:
G(V(#0,#1,#2),
E($#0($#1,$#2),
$#1($#0,$#2), $#2($#0,$#1)))
Adjacency matrix:
,,,3,4,,[:$#0:6→2],7,8,,,,,,,
,,,,,,,[:$#1:7→3],,,,,,,,
,,,,,,,,[:$#2:8→4],,,,,,,
,,2,,4,,,,,[:$#1:9→3],10,11,,,,
,,,,,,,,,,[:$#0:10→2],,,,,
,,,,,,,,,,[:$#2:11→4],,,,
,,2,3,,,,,,,,[:$#2:12→4],13,14,
,,,,,,,,,,,,[:$#0:13→2],,
,,,,,,,,,,,,,[:$#1:14→3],

Binding and reforming data

Interpreted structure



Statements

Input:

(#1$1[2],#2$2[2],$3[3](#4$4[2]));
PI($#1,Quantity($#4,$#2))

Data:

Length,Weight, mm,kg, 1,2, 322,4, 5,68

Output:

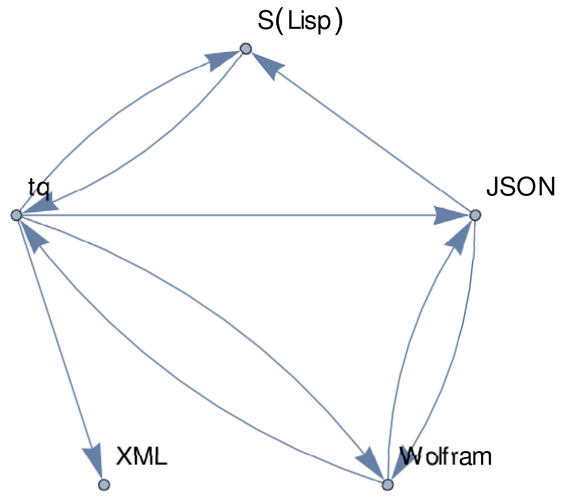
(((Length,Quantity(1,mm)),
(Weight,Quantity(2,kg))),
((Length,Quantity(322,mm)),
(Weight,Quantity(4,kg))),
((Length,Quantity(5,mm)),
(Weight,Quantity(68,kg))))

Development status

Program construction

- tq parser (designated as "tq") ... Done
- converter ... Developing
- analyzer ... TBD

Supported forms



Performance

Table: Parsing performance @ E5-2650

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
tq (stable)	498,615,417	3:30	104G
tq (exptl.)	498,615,417	1:55	51G
jq	498,615,417	10:50	144G

Table: Parsing and converting performance

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

Future plan

As a next step, we are restructuring the data structure of tq for parallelizing. In the current structure, the tree structure and node property are strongly related; therefore, parallelizing is difficult. We are attempting to divide the data structure into tree structure and node table.

Conclusion

tq can handle various types of data in a uniform manner, especially in the field of materials science. Adopting the syntax of S-expressions, tq incorporates the binding and node referencing mechanism to represent a graph structure that defines input and output data formats. Due to its expressive power, users can write a set of rules that reform unstructured data (e.g.CSV) into those of an arbitrary format as they need, such as a tensor format for machine learning.