Satisfiability for SCULPT-Schemas for CSV-Like Data

Johannes Doleschal
Universität Bayreuth

Frank Neven
Hasselt University

Wim Martens
Universität Bayreuth

Adam Witkowski
University of Warsaw

CSV-Like Data
(Comma-Separated Values)

Top Locations in Vienna

[…]

MuseumsQuartier Wien;	museum;	Museumsplatz 1 ;	1070;	Wien;
Musikverein;	musicstage;	Musikvereinsplatz 1;	1010;	Wien;
MuTh Museum & Theater;	musicstage;	Obere Augartenstrafle 1E;	1020;	Wien;
Naschmarkt Deli;	gastronomy;	Naschmarkt 421-436 ;	1060;	Wien;
Naturhistorisches Museum;	museum;	Maria-Theresien-Platz;	1010;	Wien;

[…]

www.data.wien.gv.at/csv/top-locations-wien.csv

CSV-Like Data
(Comma-Separated Values)

Top Locations in Vienna

```
[…]
MuseumsQuartier Wien;   museum;      Museumsplatz 1 ;          1070;   Wien;
Musikverein;            musicstage;  Musikvereinsplatz 1;      1010;   Wien;
MuTh Museum & Theater;  musicstage;  Obere Augartenstrafle 1E; 1020;   Wien;
Naschmarkt Deli;        gastronomy;  Naschmarkt 421-436 ;      1060;   Wien;
Naturhistorisches Museum; museum;    Maria-Theresien-Platz;    1010;   Wien;
[…]
```

www.data.wien.gv.at/csv/top-locations-wien.csv

Brexit Voting Data:

```
Region,       Area,              Votes_Cast,  Remain,  Leave,  Voting_for_both_answers
London,       Wandsworth,        158018,      118463,  39421,  55
London,       Westminster,       78325,       53928,   24268,  47
North East,   Hartlepool,        46134,       14029,   32071,  12
North East,   Middlesbrough,     61393,       21181,   40177,  16
Scotland,     City of Edinburgh, 252481,      187796,  64498,  71
[…]
```

www.electoralcommission.org.uk

CSV-Like Data
(Comma-Separated Values)

- Spreadsheets (e.g. Excel files)

CSV-Like Data
(Comma-Separated Values)

- Spreadsheets (e.g. Excel files)
- Exchanging data using a “low-tech” format

CSV-Like Data
(Comma-Separated Values)

- Spreadsheets (e.g. Excel files)
- Exchanging data using a “low-tech” format
- Public open data (e.g. data.gov, data.worldbank.org,…)

CSV-Like Data
(Comma-Separated Values)

- Spreadsheets (e.g. Excel files)
- Exchanging data using a “low-tech” format
- Public open data (e.g. data.gov, data.worldbank.org,…)
- Relational database dumps

But…

- Tabular data / CSV has many irregularities

But…

- Tabular data / CSV has many irregularities
- Because there was no real standard until 2015
  - *Besides an informal memo (RFC4180 The Internet Society, 2005)*

But…

- Tabular data / CSV has many irregularities
- Because there was no real standard until 2015
  - *Besides an informal memo (RFC4180 The Internet Society, 2005)*

"2/3 of 'CSV' files on data.gov.uk are
not machine-readable [in an elegant way]"
Jeni Tennison (2014)
(Open Data Institute and W3C CSV on the Web WG)

But…

- Tabular data / CSV has many irregularities
- Because there was no real standard until 2015
  - *Besides an informal memo (RFC4180 The Internet Society, 2005)*

"2/3 of 'CSV' files on data.gov.uk are
not machine-readable [in an elegant way]"
Jeni Tennison (2014)
(Open Data Institute and W3C CSV on the Web WG)

In 2015, the W3C standardized

Tabular Data and Metadata on the Web!

Metadata Vocabulary for Tabular Data (W3C)

CSV Data

[…]
MuseumsQuartier Wien; museum; Museumsplatz 1; 1070; Wien;
Musikverein; musicstage; Musikvereinsplatz 1; 1010; Wien;
MuTh Museum & Theater; musicstage; Obere Augartenstrafle 1E; 1020; Wien;
Naschmarkt Deli; gastronomy; Naschmarkt 421-436; 1060; Wien;
Naturhistorisches Museum; museum; Maria-Theresien-Platz; 1010; Wien;
[…]

Metadata Vocabulary for Tabular Data (W3C)

CSV Data

[…]				
MuseumsQuartier Wien;	museum;	Museumsplatz 1;	1070;	Wien;
Musikverein;	musicstage;	Musikvereinsplatz 1;	1010;	Wien;
MuTh Museum & Theater;	musicstage;	Obere Augartenstrafle 1E;	1020;	Wien;
Naschmarkt Deli;	gastronomy;	Naschmarkt 421-436;	1060;	Wien;
Naturhistorisches Museum;	museum;	Maria-Theresien-Platz;	1010;	Wien;
[…]				

Datatypes

string	string	string	integer	string

Metadata Vocabulary for Tabular Data (W3C)

CSV Data

```
[…]
MuseumsQuartier Wien;  museum;      Museumsplatz 1;          1070;  Wien;
Musikverein;           musicstage;  Musikvereinsplatz 1;     1010;  Wien;
MuTh Museum & Theater; musicstage;  Obere Augartenstrafle 1E; 1020;  Wien;
Naschmarkt Deli;       gastronomy;  Naschmarkt 421-436;      1060;  Wien;
Naturhistorisches Museum; museum;  Maria-Theresien-Platz;    1010;  Wien;
[…]
```

Datatypes

string	string	string	integer	string

W3C-style Metadata (i.e., the Schema)

```
"columns": [
  {"datatype": "string“},
  {"datatype": "string"},
  {"datatype": "string"},
  {"datatype": "integer“},
  {"datatype": "string“}
]
```

Properties of the W3C Approach

CSV Data

[…]				
MuseumsQuartier Wien;	museum;	Museumsplatz 1;	1070;	Wien;
Musikverein;	musicstage;	Musikvereinsplatz 1;	1010;	Wien;
MuTh Museum & Theater;	musicstage;	Obere Augartenstrafle 1E;	1020;	Wien;
Naschmarkt Deli;	gastronomy;	Naschmarkt 421-436;	1060;	Wien;
Naturhistorisches Museum;	museum;	Maria-Theresien-Platz;	1010;	Wien;
[…]				

Datatypes

string	string	string	integer	string

Properties of the W3C Approach

CSV Data

```
[…]
MuseumsQuartier Wien;  museum;      Museumsplatz 1;          1070;  Wien;
Musikverein;           musicstage;  Musikvereinsplatz 1;      1010;  Wien;
MuTh Museum & Theater; musicstage;  Obere Augartenstrafle 1E; 1020;  Wien;
Naschmarkt Deli;       gastronomy;  Naschmarkt 421-436;       1060;  Wien;
Naturhistorisches Museum; museum;  Maria-Theresien-Platz;     1010;  Wien;
[…]
```

Datatypes

string	string	string	integer	string

(P1) All rows have equal length

Properties of the W3C Approach

CSV Data

```
[…]
MuseumsQuartier Wien;  museum;      Museumsplatz 1;          1070;  Wien;
Musikverein;           musicstage;  Musikvereinsplatz 1;      1010;  Wien;
MuTh Museum & Theater; musicstage;  Obere Augartenstrafle 1E; 1020;  Wien;
Naschmarkt Deli;       gastronomy;  Naschmarkt 421-436;       1060;  Wien;
Naturhistorisches Museum; museum;   Maria-Theresien-Platz;    1010;  Wien;
[…]
```

Datatypes

string	string	string	integer	string

(P1) All rows have equal length

(P2) Only one data type per column

Properties of the W3C Approach

CSV Data

```
[...]
MuseumsQuartier Wien;  museum;       Museumsplatz 1;          1070;  Wien;
Musikverein;           musicstage;   Musikvereinsplatz 1;      1010;  Wien;
MuTh Museum & Theater; musicstage;   Obere Augartenstrafle 1E; 1020;  Wien;
Naschmarkt Deli;       gastronomy;   Naschmarkt 421-436;       1060;  Wien;
Naturhistorisches Museum; museum;    Maria-Theresien-Platz;    1010;  Wien;
[...]
```

Datatypes

string	string	string	integer	string

(P1) All rows have equal length

(P2) Only one data type per column

(P3) Number of columns is bounded by the size of the schema

CSV on the Web: Use Case 13

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

CSV on the Web: Use Cases and Requirements, Use Case 13

CSV on the Web: Use Case 13

- “[…] a single row in the table comprises a triple (subject-predicate-object), one or more provenance references and an optional certainty measure“ (description of Use Case 13)

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

CSV on the Web: Use Cases and Requirements, Use Case 13

CSV on the Web: Use Case 13

- “[…] a single row in the table comprises a triple (subject-predicate-object), one or more provenance references and an optional certainty measure“ (description of Use Case 13)

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

CSV on the Web: Use Cases and Requirements, Use Case 13

CSV on the Web: Use Case 13

- Rows of different length

violates (P1)

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

CSV on the Web: Use Cases and Requirements, Use Case 13

CSV on the Web: Use Case 13

- Rows of different length — violates (P1)
- Columns can contain different data types — violates (P2)

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

CSV on the Web: Use Cases and Requirements, Use Case 13

CSV on the Web: Use Case 13

- Rows of different length — violates (P1)
- Columns can contain different data types — violates (P2)
- Row length is not limited — violates (P3)

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

CSV on the Web: Use Cases and Requirements, Use Case 13

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri rdf-lit certainty

doc-ID position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) →

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate | object

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate | object | provenance

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate | object | provenance

column(subject) →

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate | object | provenance

column(subject) → rdf-uri

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate | object | provenance

column(subject) → rdf-uri

column(predicate) →

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate | object | provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	"Bart"	D00124	283-286						
:e4	mention	"JoJo"	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	"10"	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate | object | provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	"Bart"	D00124	283-286						
:e4	mention	"JoJo"	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	"10"	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate | object | provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) →

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate | object | provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) → rdf-uri

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	"Bart"	D00124	283-286						
:e4	mention	"JoJo"	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	"10"	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate | object | provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) → rdf-uri + rdf-lit

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	"Bart"	D00124	283-286						
:e4	mention	"JoJo"	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	"10"	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate | object | provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) → rdf-uri + rdf-lit

rectangle(provenance) →

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate | object | provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) → rdf-uri + rdf-lit

rectangle(provenance) → doc-ID

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri | rdf-lit | certainty | doc-ID | position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject | predicate | object | provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) → rdf-uri + rdf-lit

rectangle(provenance) → doc-ID | position *

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri rdf-lit certainty

doc-ID position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject predicate object provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) → rdf-uri + rdf-lit

rectangle(provenance) → doc-ID position * certainty ?

SCULPT by Example

Martens, Neven, Vansummeren [WWW, 2015]

data-types

rdf-uri rdf-lit certainty

doc-ID position

Data:

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

SCULPT Schema:

row(1) → subject predicate object provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) → rdf-uri + rdf-lit

rectangle(provenance) → (doc-ID position * certainty ?)*

SCULPT

Rule Based Language:

‹ region selection › → ‹ content expression ›

SCULPT

Rule Based Language:

‹ region selection › → ‹ content expression ›

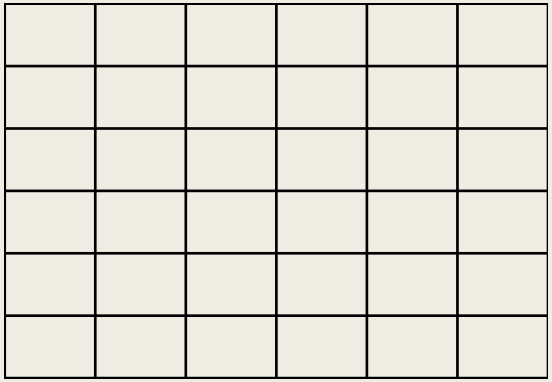
- The Region selections select sets of cells of the csv file
  - *based on Propositional Dynamic Logic*
  - *using the axes up, down, left, right*

SCULPT

Rule Based Language:

‹ region selection › ⟶ ‹ content expression ›

- The Region selections select sets of cells of the csv file
  - *based on Propositional Dynamic Logic*
  - *using the axes up, down, left, right*

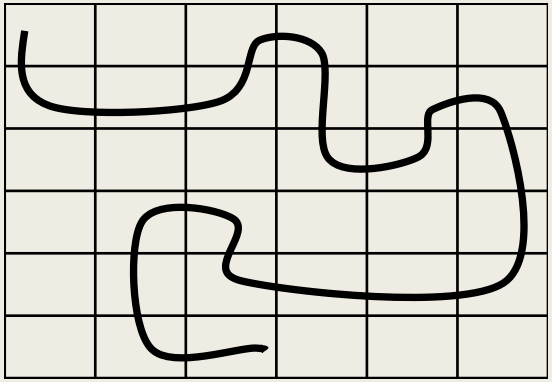


SCULPT

Rule Based Language:

‹ region selection › ⟶ ‹ content expression ›

- The Region selections select sets of cells of the csv file
  - *based on Propositional Dynamic Logic*
  - *using the axes up, down, left, right*

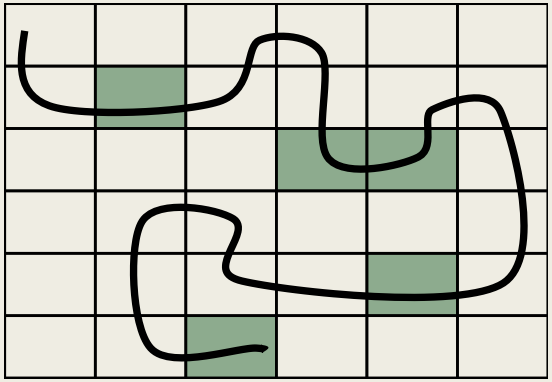


SCULPT

Rule Based Language:

‹ region selection › ⟶ ‹ content expression ›

- The Region selections select sets of cells of the csv file
  - *based on Propositional Dynamic Logic*
  - *using the axes up, down, left, right*

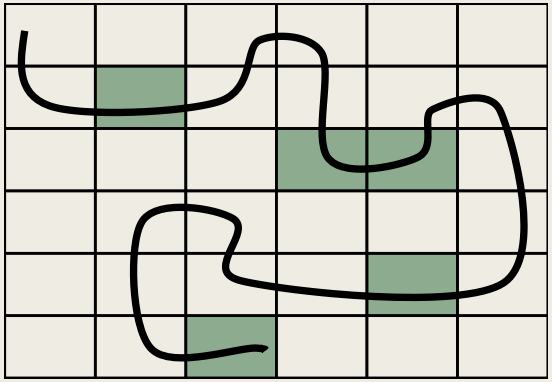


SCULPT

Rule Based Language:

‹ region selection › ⟶ ‹ content expression ›

- The Region selections select sets of cells of the csv file
  - *based on Propositional Dynamic Logic*
  - *using the axes up, down, left, right*
- The Content expressions are regular expressions over data-types



SCULPT

Rule Based Language:

‹ region selection › ⟶ ‹ content expression ›

- The Region selections select sets of cells of the csv file
  - *based on Propositional Dynamic Logic*
  - *using the axes up, down, left, right*
- The Content expressions are regular expressions over data-types
- A document satisfies a rule iff the selected region satisfy the content expression

SCULPT

Rule Based Language:

‹ region selection › ⟶ ‹ content expression ›

- The Region selections select sets of cells of the csv file
  - *based on Propositional Dynamic Logic*
  - *using the axes up, down, left, right*
- The Content expressions are regular expressions over data-types
- A document satisfies a rule iff the selected region satisfy the content expression
- A document should satisfy every rule

SCULPT

Rule Based Language:

‹ region selection › → ‹ content expression ›

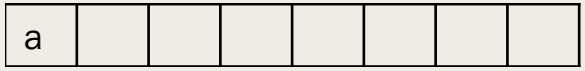
Schema Evaluation [Martens et al., WWW 2015]

Given a **document D** and a Sculpt **schema S**,
it can be tested in **linear time,** combines complexity, whether **D satisfies S**.

Static Analysis of CSV-Schemata

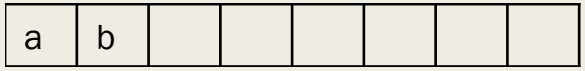
Schema Satisfiablity

(1,1) → a
right(a) → b + c
right(b) → a + c
right(c) → a



Schema Satisfiablity

(1,1) → a
right(a) → b + c
right(b) → a + c
right(c) → a



Schema Satisfiablity

(1,1) → a
right(a) → b + c
right(b) → a + c
right(c) → a

a	b	a					

Schema Satisfiablity

(1,1) → a
right(a) → b + c
right(b) → a + c
right(c) → a

a	b	a	c				

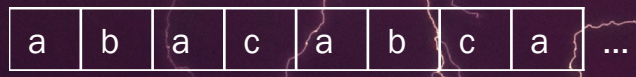
Schema Satisfiablity

(1,1) → a
right(a) → b + c
right(b) → a + c
right(c) → a

a	b	a	c	a			

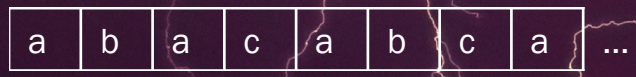
Schema Satisfiablity

(1,1) → a
right(a) → b + c
right(b) → a + c
right(c) → a



Schema Satisfiablity

(1,1) → a
right(a) → b + c
right(b) → a + c
right(c) → a



Schema Satisfiability

Given a schema S, is there a CSV-File that satisfies the schema?

Satisfiability of SCULPT is undecidable

The halting problem can be encoded easily (and in several ways)

q_0, ▷	0	1	0	1	1	0	1	0	␣
▷	q_1, 0	1	0	1	1	0	1	0	␣
▷	1	q_2, 1	0	1	1	0	1	0	␣
▷	1	q_1, 0	0	1	1	0	1	0	␣
▷	1	1	q_2, 0	1	1	0	1	0	␣
▷	1	1	1	q_2, 1	1	0	1	0	␣
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
▷	yes	␣	␣	␣	␣	␣	␣	␣	␣

Similar idea was used by Göller, Lohrey, Lutz [JSL 2009]

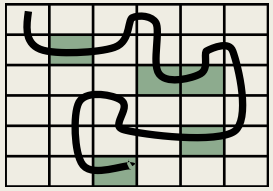
Satisfiability of SCULPT is undecidable

Reasons for undecidability:

Satisfiability of SCULPT is undecidable

Reasons for undecidability:

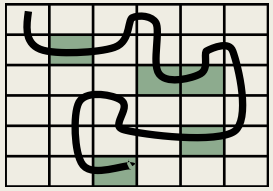
- “Complex” navigation in the table



Satisfiability of SCULPT is undecidable

Reasons for undecidability:

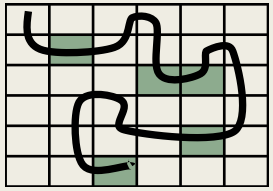
- “Complex” navigation in the table
- Cells are selected by multiple regions



Satisfiability of SCULPT is undecidable

Reasons for undecidability:

- “Complex” navigation in the table
- Cells are selected by multiple regions



Either one makes satisfiability undecidable

But do we really need this in practice?

Real-life cases are less complex

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

Used Regions Selections:

- row(1)
- column(subject)
- column(predicate)
- column(object)
- rectangle(provenance)

Real-life cases are less complex

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

Used Regions Selections:

- row(1)
- column(subject)
- column(predicate)
- column(object)
- rectangle(provenance)

  - Each cell is selected by at most one region

Real-life cases are less complex

subject	predicate	object	provenance							
:e4	mention	"Bart"	D00124	283-286						
:e4	mention	"JoJo"	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	"10"	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

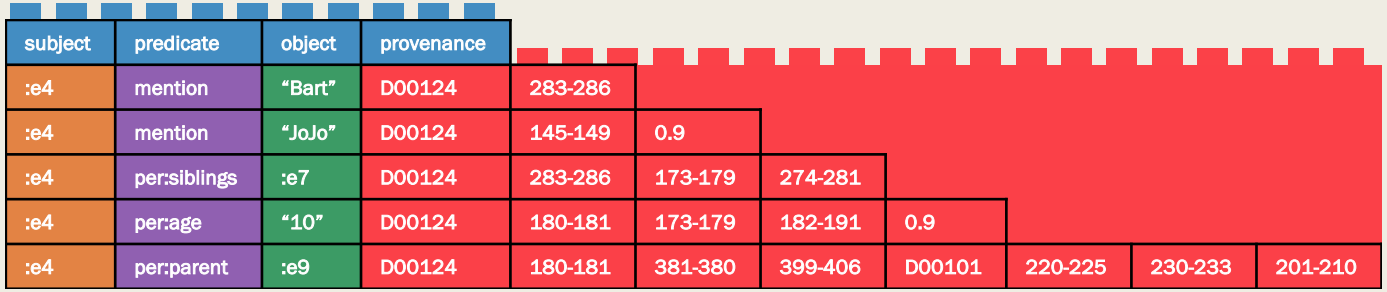
Used Regions Selections:

- row(1)
- column(subject)
- column(predicate)
- column(object)
- rectangle(provenance)

  - Each cell is selected by at most one region
  - Navigation is very simple

Lego-SCULPT

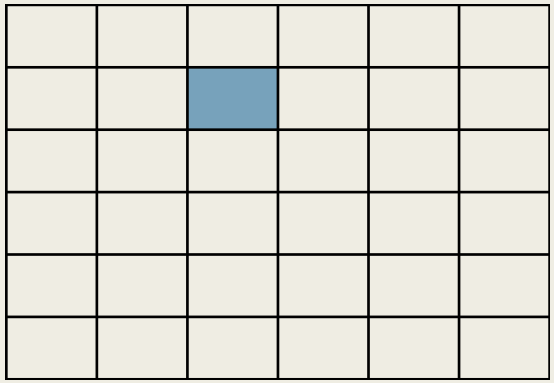
Finding a balance between usability and computational complexity



subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

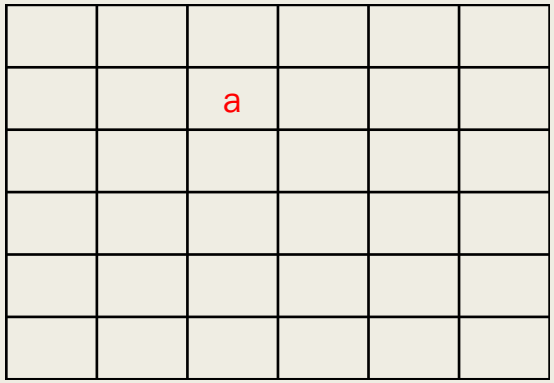
Lego-SCULPT Region Selections

- (2,3)



Lego-SCULPT Region Selections

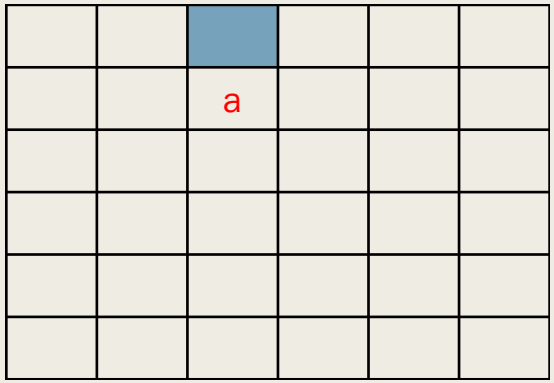
- (2,3)



a is
- data-type, or
- coordinate

Lego-SCULPT Region Selections

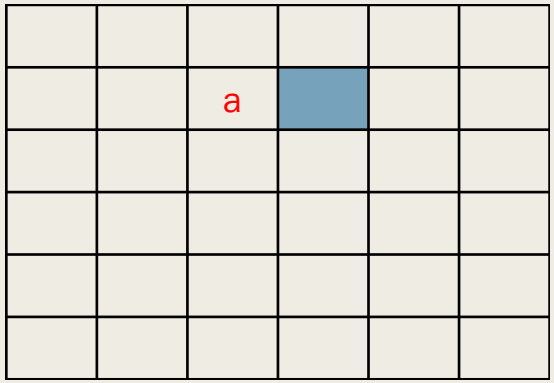
- (2,3)
- up(a)



a is
- data-type, or
- coordinate

Lego-SCULPT Region Selections

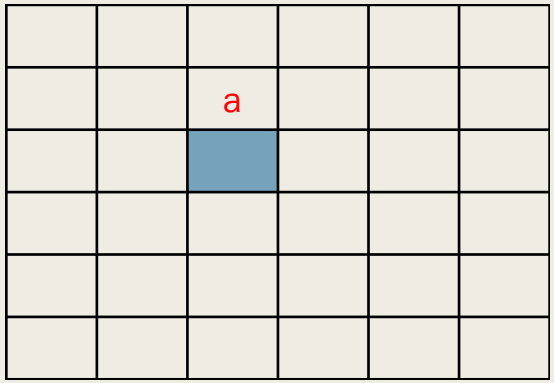
- (2,3)
- up(a)
- right(a)



a is
- data-type, or
- coordinate

Lego-SCULPT Region Selections

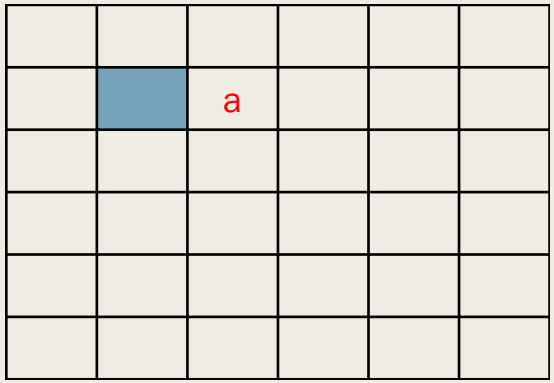
- (2,3)
- up(a)
- right(a)
- down(a)



a is
- data-type, or
- coordinate

Lego-SCULPT Region Selections

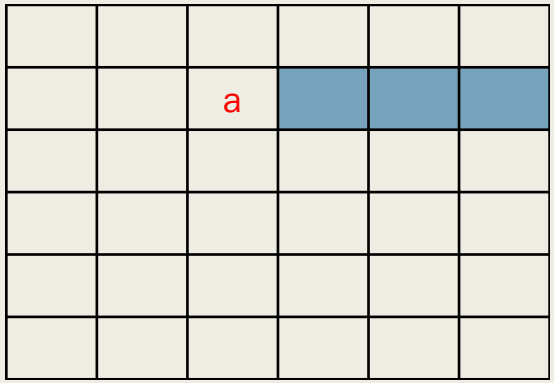
- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)



a is
- data-type, or
- coordinate

Lego-SCULPT Region Selections

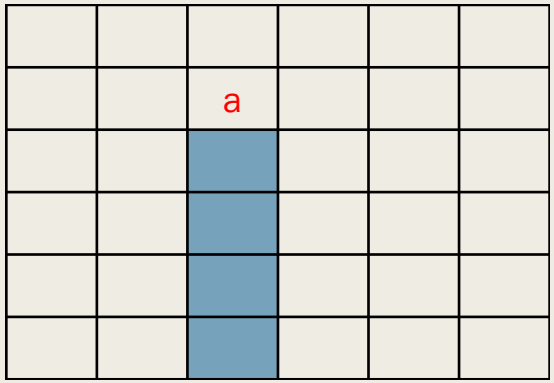
- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a)



a is
- data-type, or
- coordinate

Lego-SCULPT Region Selections

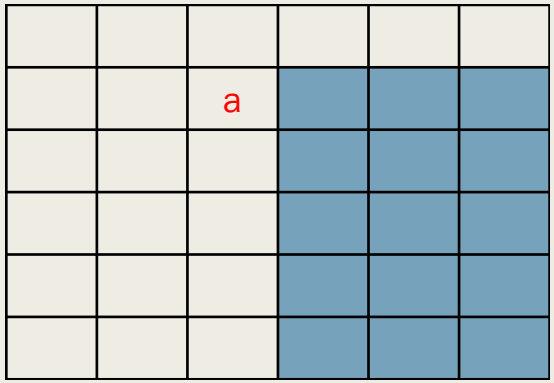
- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a)
- column(a)



a is
- data-type, or
- coordinate

Lego-SCULPT Region Selections

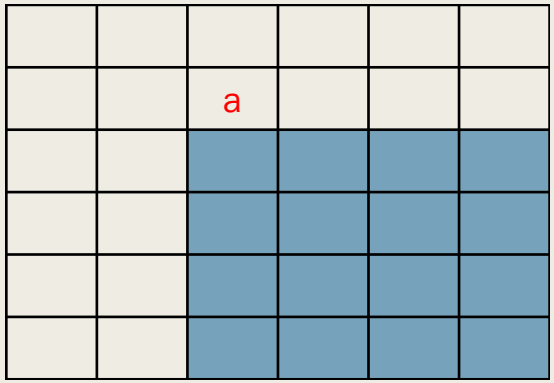
- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a)
- column(a)
- rectangle$^{\rightarrow}$(a)



a is
- data-type, or
- coordinate

Lego-SCULPT Region Selections

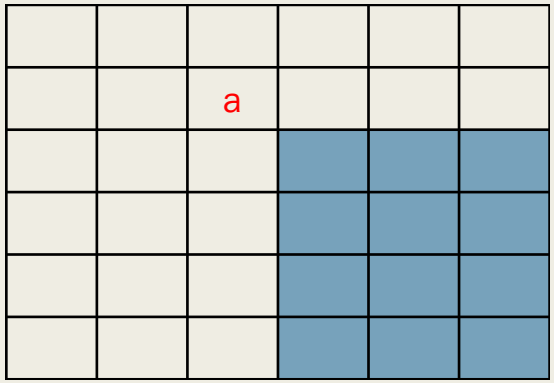
- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a)
- column(a)
- rectangle$^{\rightarrow}$(a)
- rectangle$^{\downarrow}$(a)



a is
- data-type, or
- coordinate

Lego-SCULPT Region Selections

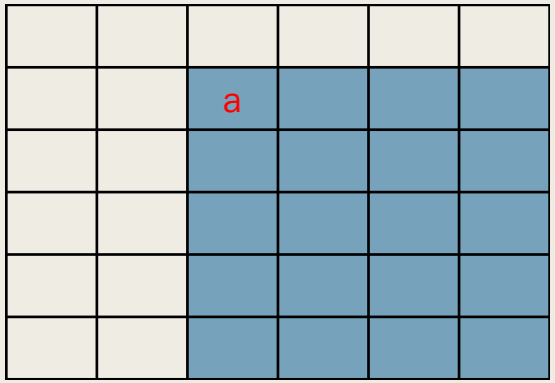
- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a)
- column(a)
- rectangle$^{\rightarrow}$(a)
- rectangle$^{\downarrow}$(a)
- rectangle$^{\searrow}$(a)



a is
- data-type, or
- coordinate

Lego-SCULPT Region Selections

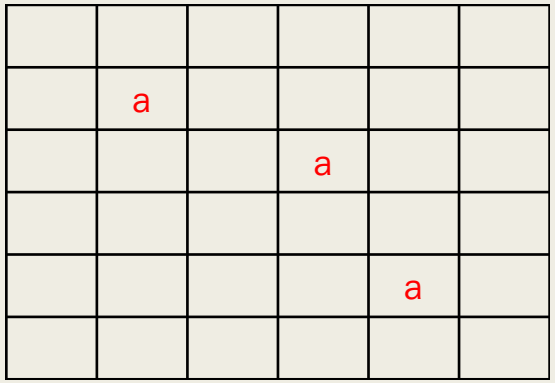
- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a)
- column(a)
- rectangle→(a)
- rectangle↓(a)
- rectangle↘(a)
- rectangle•(a)



a is
- data-type, or
- coordinate

Lego-SCULPT Region Selections

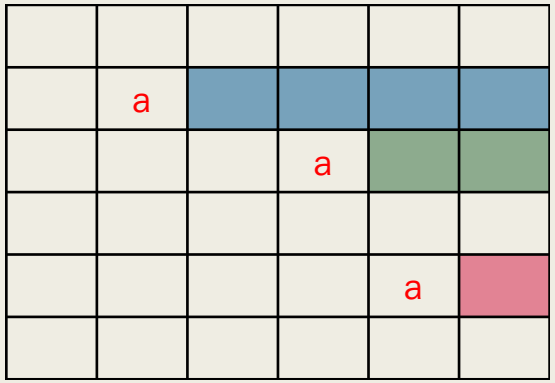
- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a)
- column(a)
- rectangle$^{\rightarrow}$(a)
- rectangle$^{\downarrow}$(a)
- rectangle$^{\searrow}$(a)
- rectangle$^{\cdot}$(a)



a is a data-type

Lego-SCULPT Region Selections

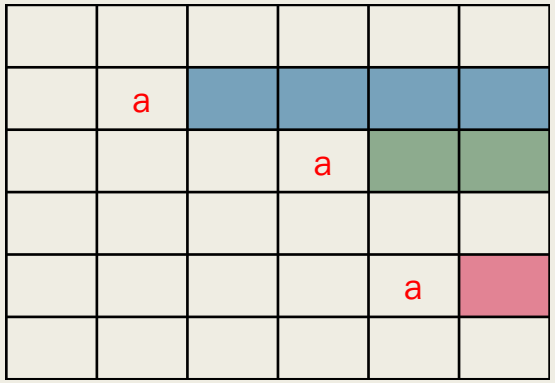
- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- **row(a)**
- column(a)
- rectangle$^{\rightarrow}$(a)
- rectangle$^{\downarrow}$(a)
- rectangle$^{\searrow}$(a)
- rectangle$^{\cdot}$(a)



a is a data-type

Lego-SCULPT Region Selections

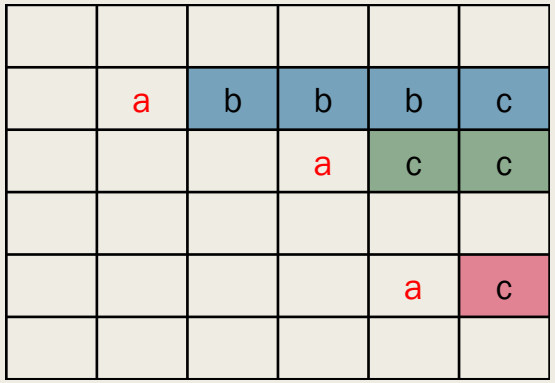
- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a) = {■■■■, ■■ , ■}
- column(a)
- rectangle→(a)
- rectangle↓(a)
- rectangle↘(a)
- rectangle•(a)



a is a data-type

Lego-SCULPT Region Selections

- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a) = {■■■, ■■ , ■}
- column(a)
- rectangle→(a)
- rectangle↓(a)
- rectangle↘(a)
- rectangle•(a)

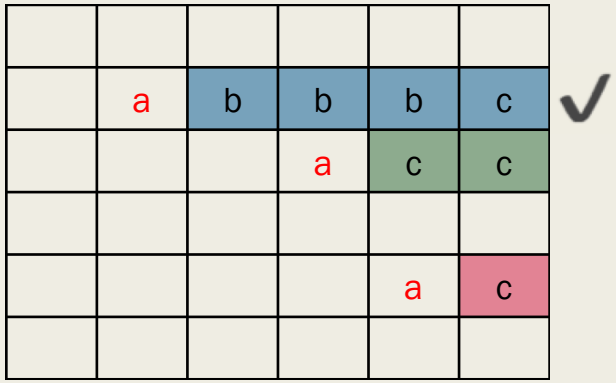


a is a data-type

row(a) ⟶ b*c*

Lego-SCULPT Region Selections

- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a) = {■■■■, ■■ , ■}
- column(a)
- rectangle→(a)
- rectangle↓(a)
- rectangle↘(a)
- rectangle•(a)

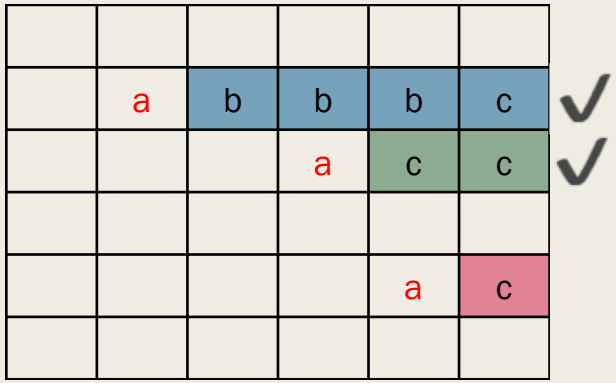


a is a data-type

row(a) ⟶ b*c*

Lego-SCULPT Region Selections

- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a) = {■■■, ■■ , ■}
- column(a)
- rectangle→(a)
- rectangle↓(a)
- rectangle↘(a)
- rectangle•(a)

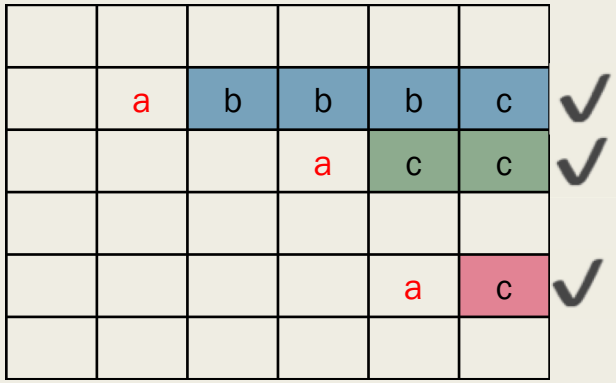


a is a data-type

row(a) ⟶ b*c*

Lego-SCULPT Region Selections

- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a) = {■■■, ■■ , ■}
- column(a)
- rectangle→(a)
- rectangle↓(a)
- rectangle↘(a)
- rectangle•(a)

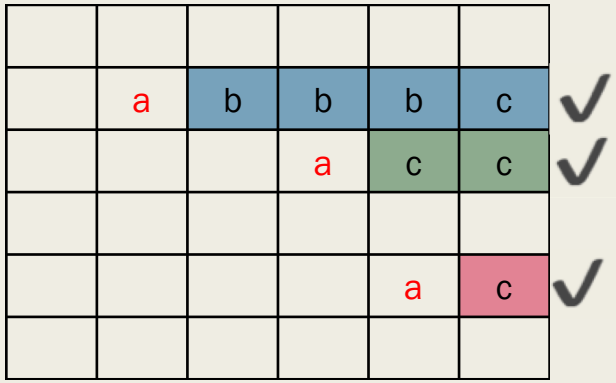


a is a data-type

row(a) ⟶ b*c*

Lego-SCULPT Region Selections

- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a) = {■■■■, ■■ , ■}
- column(a)
- rectangle→(a)
- rectangle↓(a)
- rectangle↘(a)
- rectangle•(a)



a is a data-type

row(a) ⟶ b*c*

Lego-SCULPT Region Selections

- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a) = {■■■■, ■■ , ■}
- column(a)
- rectangle→(a)
- rectangle↓(a)
- rectangle↘(a)
- rectangle•(a)

	a	b	b	b	c
			a	c	c
				a	c

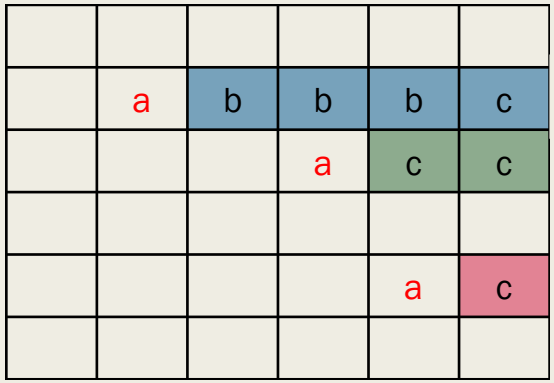
a is a data-type

row(a) → b*c* ✓

row(a) → c*

Lego-SCULPT Region Selections

- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a) = {■■■■, ■■ , ■}
- column(a)
- rectangle→(a)
- rectangle↓(a)
- rectangle↘(a)
- rectangle•(a)



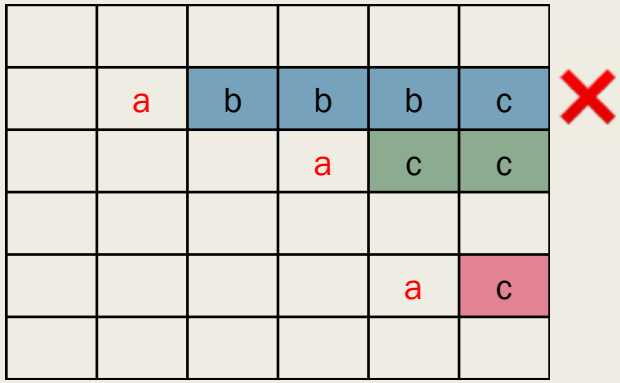
a is a data-type

row(a) → b*c*

row(a) → c*

Lego-SCULPT Region Selections

- (2,3)
- up(a)
- right(a)
- down(a)
- left(a)
- row(a) = {■■■■, ■■ , ■}
- column(a)
- rectangle→(a)
- rectangle↓(a)
- rectangle↘(a)
- rectangle•(a)



a is a data-type

row(a) → b*c* ✓

row(a) → c* ✗

Lego-SCULPT SAT

Satisfiability (SAT):

Input:	Lego-SCULPT **schema S.**
Question:	Is there a **table**, **satisfying all constraints**, such that **each cell is selected by at most one region**?

Lego-SCULPT SAT

Satisfiability (SAT):

Input: Lego-SCULPT **schema S.**
Question: Is there a **table**, **satisfying all constraints**, such that **each cell is selected by at most one region?**

Main Theorems:

SAT for **Lego-SCULPT(right, row, column, rectangle$^{\{\cdot,\rightarrow,\downarrow,\searrow\}}$)** is **PTIME-complete.**

Lego-SCULPT SAT

Satisfiability (SAT):

Input: Lego-SCULPT **schema S.**
Question: Is there a **table**, **satisfying all constraints**, such that **each cell is selected by at most one region?**

Main Theorems:

SAT for **Lego-SCULPT(right, row, column, rectangle$^{\{\cdot,\rightarrow,\downarrow,\searrow\}}$)** is **PTIME-complete.**

Adding any other axes (like left, up, …) results in **undecidability** for **SAT.**

Encoding Tables as Trees

subject	predicate	object	provenance						
:e4	mention	“Bart”	D00124	283-286					
:e4	mention	“JoJo”	D00124	145-149	0.9				
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281			
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9		
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

row(1) → subject predicate object provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) → rdf-uri + rdf-lit

rectangle(provenance) → (doc-ID position * certainty ?)*

Encoding Tables as Trees

subject	predicate	object	provenance						
:e4	mention	“Bart”	D00124	283-286					
:e4	mention	“JoJo”	D00124	145-149	0.9				
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281			
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9		
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

row(1) → subject predicate object provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) → rdf-uri + rdf-lit

rectangle(provenance) → (doc-ID position * certainty ?)*

Encoding Tables as Trees

subject	predicate	object	provenance						
:e4	mention	“Bart”	D00124	283-286					
:e4	mention	“JoJo”	D00124	145-149	0.9				
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281			
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9		
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

row(1) → subject predicate object provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) → rdf-uri + rdf-lit

rectangle(provenance) → (doc-ID position * certainty ?)*

Encoding Tables as Trees

subject	predicate	object	provenance							
:e4	mention	“Bart”	D00124	283-286						
:e4	mention	“JoJo”	D00124	145-149	0.9					
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281				
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9			
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

row(1) → subject predicate object provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) → rdf-uri + rdf-lit

rectangle(provenance) → (doc-ID position * certainty ?)*

Encoding Tables as Trees

subject	predicate	object	provenance						
:e4	mention	“Bart”	D00124	283-286					
:e4	mention	“JoJo”	D00124	145-149	0.9				
:e4	per:siblings	:e7	D00124	283-286	173-179	274-281			
:e4	per:age	“10”	D00124	180-181	173-179	182-191	0.9		
:e4	per:parent	:e9	D00124	180-181	381-380	399-406	D00101	220-225	230-233	201-210

row(1) → subject predicate object provenance

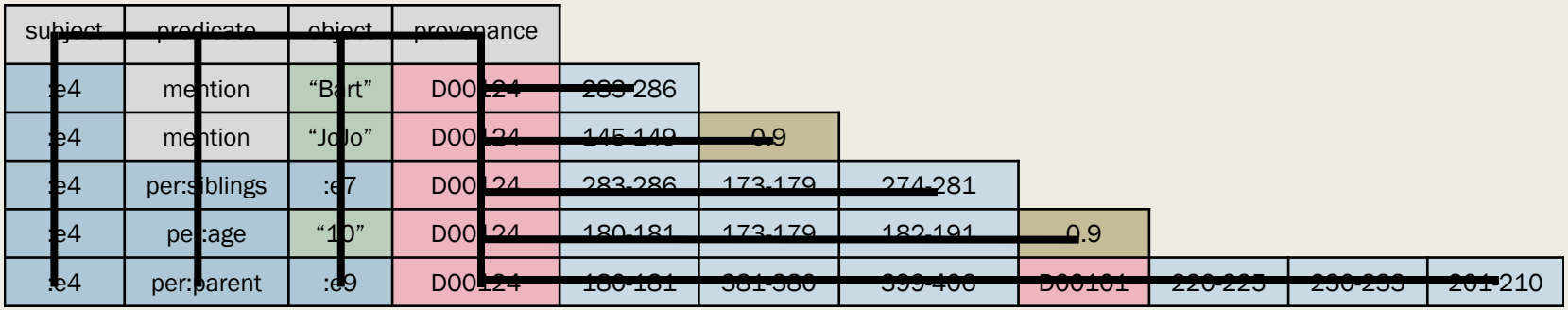
column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) → rdf-uri + rdf-lit

rectangle(provenance) → (doc-ID position * certainty ?)*

Encoding Tables as Trees



row(1) → subject predicate object provenance

column(subject) → rdf-uri

column(predicate) → rdf-uri + mention

column(object) → rdf-uri + rdf-lit

rectangle(provenance) → (doc-ID position * certainty ?)*

Encoding Tables as Trees

Table-Tree Encoding

For every **Lego-SCULPT(right, row, column, rectangle$^{\{\cdot,\rightarrow,\downarrow,\searrow\}}$)**
Schema S there is a **tree automaton A_S**, such that **S** is **satisfiable**
if and only if **$L(A_S)$** is **not empty.**

Encoding Tables as Trees

Table-Tree Encoding

For every **Lego-SCULPT(right, row, column, rectangle$^{\{\cdot,\rightarrow,\downarrow,\searrow\}}$)** **Schema S** there is a **tree automaton A_S**, such that **S** is **satisfiable** if and only if $L(A_S)$ is **not empty.**

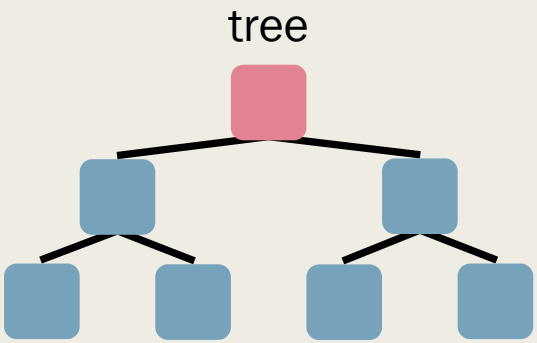
Challenge: **every node** in the **tree** has to have a **different coordinate** in the corresponding **table-model.**

Encoding Tables as Trees

Table-Tree Encoding

For every **Lego-SCULPT(right, row, column, rectangle$^{\{\cdot,\rightarrow,\downarrow,\searrow\}}$) Schema S** there is a **tree automaton A_S**, such that **S** is **satisfiable** if and only if $L(A_S)$ is **not empty.**

Challenge: **every node** in the **tree** has to have a **different coordinate** in the corresponding **table-model.**



Encoding Tables as Trees

Table-Tree Encoding

For every **Lego-SCULPT(right, row, column, rectangle$^{\{\cdot,\rightarrow,\downarrow,\searrow\}}$) Schema S** there is a **tree automaton A_S**, such that **S is satisfiable** if and only if $L(A_S)$ is **not empty.**

Challenge: **every node** in the **tree** has to have a **different coordinate** in the corresponding **table-model.**

tree

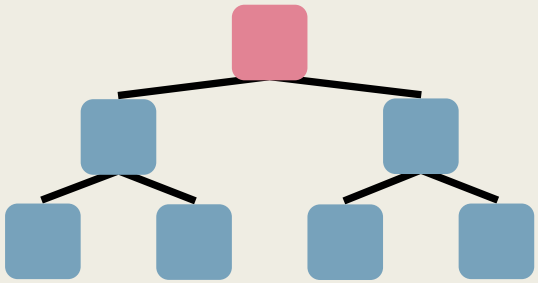


table-model:

Encoding Tables as Trees

Table-Tree Encoding

For every **Lego-SCULPT(right, row, column, rectangle$^{\{\cdot,\rightarrow,\downarrow,\searrow\}}$) Schema S** there is a **tree automaton A_S**, such that **S** is **satisfiable** if and only if $L(A_S)$ is **not empty.**

Challenge: **every node** in the **tree** has to have a **different coordinate** in the corresponding **table-model.**

tree

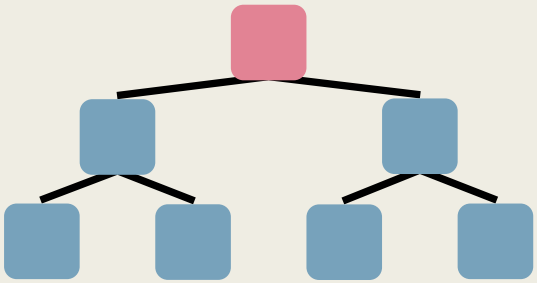


table-model:

Encoding Tables as Trees

Table-Tree Encoding

For every **Lego-SCULPT(right, row, column, rectangle$^{\{\cdot,\rightarrow,\downarrow,\searrow\}}$) Schema S** there is a **tree automaton A_S**, such that **S** is **satisfiable** if and only if $L(A_S)$ is **not empty.**

Challenge: **every node** in the **tree** has to have a **different coordinate** in the corresponding **table-model.**

tree

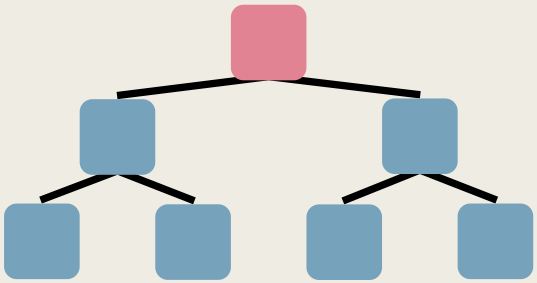
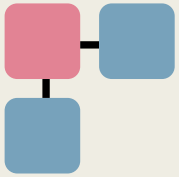


table-model:



Encoding Tables as Trees

Table-Tree Encoding

For every **Lego-SCULPT(right, row, column, rectangle$^{\{\cdot,\rightarrow,\downarrow,\searrow\}}$) Schema S** there is a **tree automaton A_S**, such that **S is satisfiable** if and only if $L(A_S)$ is **not empty.**

Challenge: **every node** in the **tree** has to have a **different coordinate** in the corresponding **table-model.**

tree

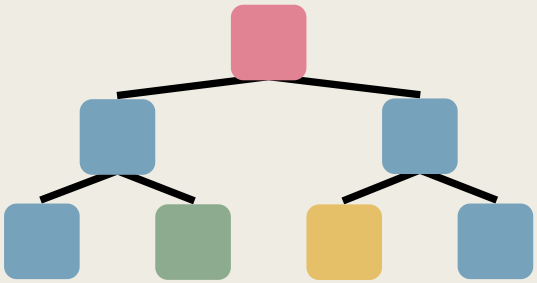
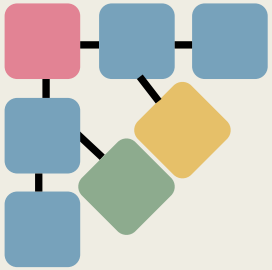


table-model:



Encoding Tables as Trees

Table-Tree Encoding

For every **Lego-SCULPT(right, row, column, rectangle$^{\{\cdot,\rightarrow,\downarrow,\searrow\}}$) Schema S** there is a **tree automaton A_S**, such that **S** is **satisfiable** if and only if $L(A_S)$ is **not empty.**

Challenge: **every node** in the **tree** has to have a **different coordinate** in the corresponding **table-model.**

tree

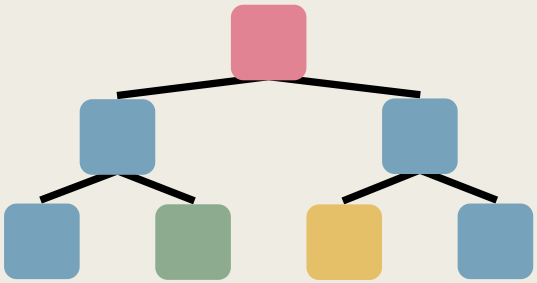
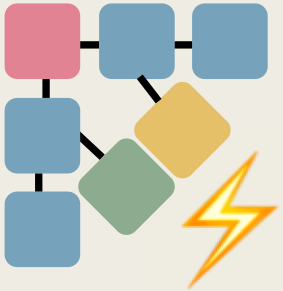


table-model:

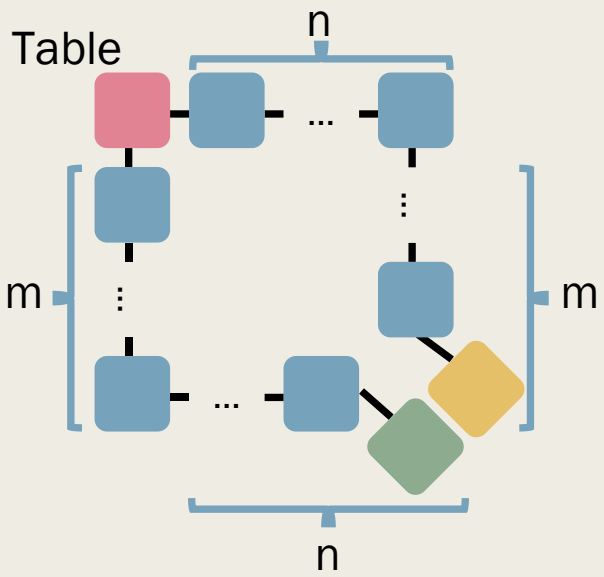


Encoding Tables as Trees

In general, this injectivity condition can **not** be **checked** by **Regular Tree Languages**!

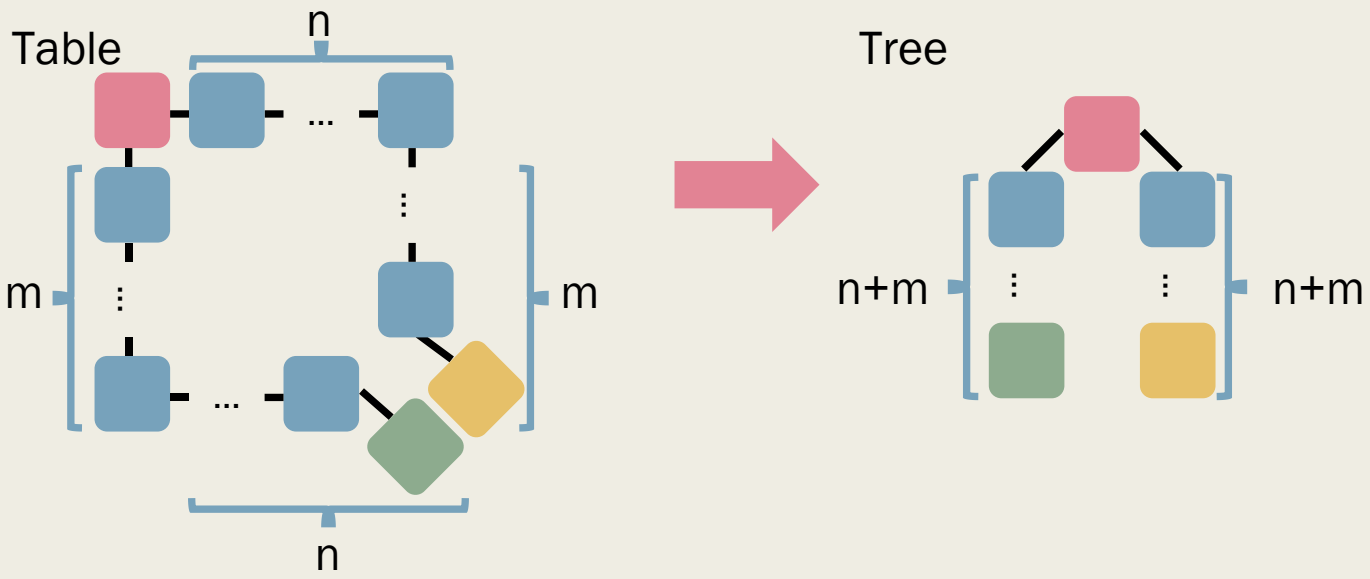
Encoding Tables as Trees

In general, this injectivity condition can **not** be **checked** by **Regular Tree Languages**!



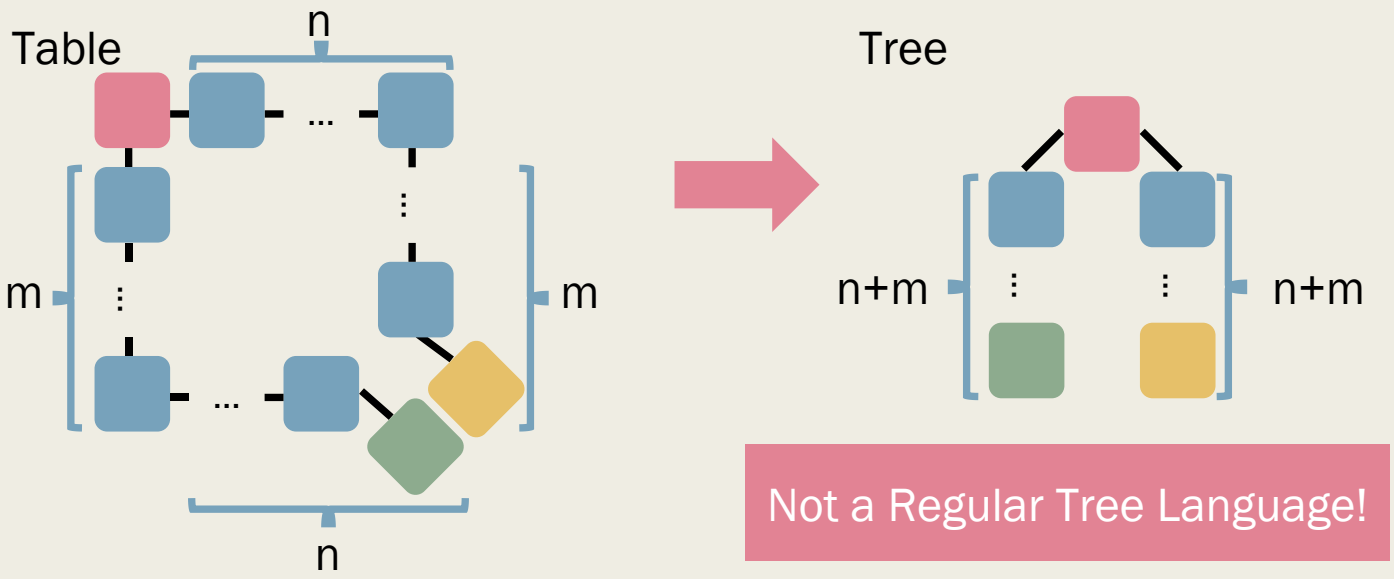
Encoding Tables as Trees

In general, this injectivity condition can **not** be **checked** by **Regular Tree Languages**!



Encoding Tables as Trees

In general, this injectivity condition can **not** be **checked** by **Regular Tree Languages**!



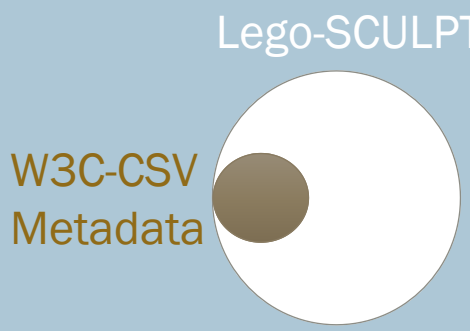
Expressiveness

Expressiveness


SCULPT
Expressiveness
W3C-CSV
Metadata
Satisfiability
• W3C-CSV Metadata: yes
• SCULPT: undecidable

SCULPT

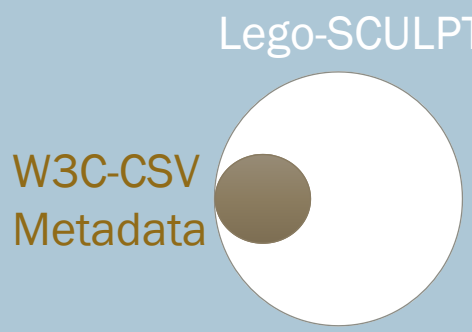
Expressiveness



Satisfiability

- W3C-CSV Metadata: yes
- SCULPT: undecidable

Expressiveness



Satisfiability

- W3C-CSV Metadata: yes
- SCULPT: undecidable
- Lego-SCULPT(right, row, col, rectangle): PTIME-complete

Related Work

- A framework for Annotating CSV-like Data, Arenas, Maturana, Riveros, Vrgoč [VLDB 2016]
  - *document-spanners to extract metadata*

Related Work

- A framework for Annotating CSV-like Data, Arenas, Maturana, Riveros, Vrgoč [VLDB 2016]
  - *document-spanners to extract metadata*
- Chisel: Sculpting Tabular and Non-Tabular Data on the Web, D., Höllerich, Martens, Neven [WWW 2018, Demo Paper]
  - *adding data transformation to SCULPT*
  - *generate W3C Metadata from SCULPT Schemata*

Thank You