Identity and Identifiers

Contents

V1: Why is identification important?

(A short introduction, referring to earlier discussions)

Identification Problems in Data Curation (again)

Identification problems

Archiving: Is this dataset already in the archive?

Preservation: Was the information preserved in the new file format?

Security: Has this dataset been tampered with?

Authentication: Is this the data we think it is?

Reproducibility: Does this XML file have the same information as that JSON file?

Provenance: Were these datasets derived from the same data?

Conversions: Does the converted file have the same data as the original?

Identifiers – what are they for?

Identifiers. . .

Enable *discovery and reuse* of relevant data sets

Support management of data sets
including *version control, correction, conversion*, etc.

Support workflow and provenance tracking

Promote transparency and reproducibility

Give credit to data produces

And more.

Identifiers, how they are used

An *identifier is often the one word answer to questions like:*

Which data set was the input for your analysis?

Which data set was the output of your analysis?

Is there a data set that has the temperatures by zip code?

Is this the JSON version of that XML data set??

Which version was corrected and anonymized?

Sure, you could say answer those questions by things like saying:

It’s this one, I think

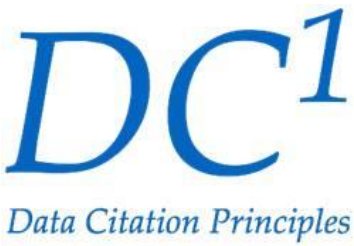
The one on the red USB drive

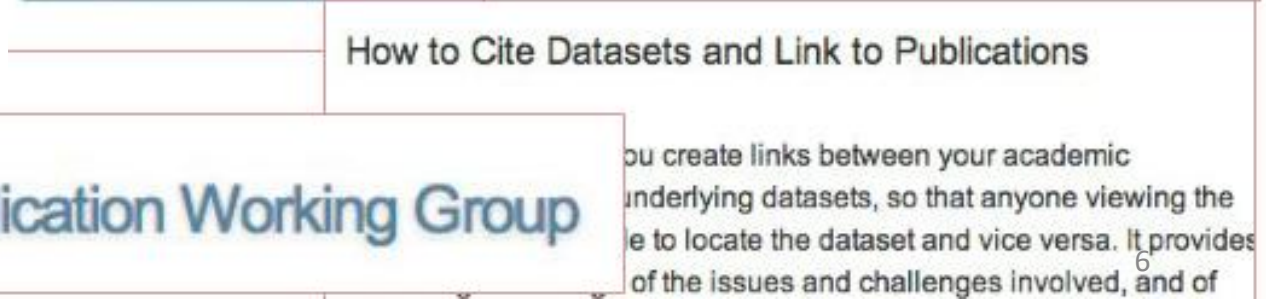
Bill has it, or maybe Daphne

NewMikeVersionSecondFinalV3.zip

But in the long run answers of this sort are neither reliable nor efficient.

Surf these up!





Readings

=> Persistent Identifiers, Fixity and Checksums, in *The Digital Preservation Handbook*, Digital Preservation Coalition, 2017.

=> On the utility of identification schemes for digital earth science data: An assessment and recommendations. Duerr, R. E., Downs, R. R., Tilmes, C., Barkstrom, B., Lenhardt, W. C., Glassy, J., Bermudez, L. E., & Slaughter, P. (2011). *Earth Science Informatics*.

Clifford A. Lynch, "Canonicalization: A Fundamental Tool To Facilitate Preservation and Management of Digital Information," *D-lib Magazine*, 5:9 (September 1999).

Canonical XML. Version 1.0. W3C Recommendation, John Boyer, March 2001. Latest version: http://www.w3.org/TR/xml-c14n.

V2: *What* are we identifying?

What exactly needs to be identified?

Lots of things to be identified

We need identifiers for lots of things:

Because without shared identifiers we cannot reliably communicate what we are taking about!

So: persons, properties, values, counties, automobiles, nations, proteins, events, etc. . .

(EVERYTHING!)

There are some shared standards in specialized domains,
and some good systems for developing identifiers, but much remains to done

Identifiers are at the heart of the semantic web:

Identifiers in RDF/OWL are URIs:

<http://example.org/#spiderman>
<http://www.perceive.net/schemas/relationship/enemyOf>
<http://example.org/#green-goblin>

Or literals:

<http://example.org/#spiderman>
<http://xmlns.com/foaf/0.1/name>
"Spiderman”

All these are important in data curation.

But in this video we will focus on identifiers for datasets.

Identity and Representation Levels (again)

Consider two files with the **same data**

*but **relational tables** in one case*

***and** **RDF triples** in another*

Same **data**, different **representations**

Identity and Representation Levels

Consider two files with

… **same** **data** and the ***same*** ***RDF triples***,

*but an **XML serialization** in one case,*

and *an **N3 serialization** in the other*

Identity and Representation Levels

Consider *two files*

with the ***same data***, ***same RDF triples***, ***same N3 serialization***,

*but an **ASCII** character encoding in one case*

***vs** an **EBCDIC** encoding in another*

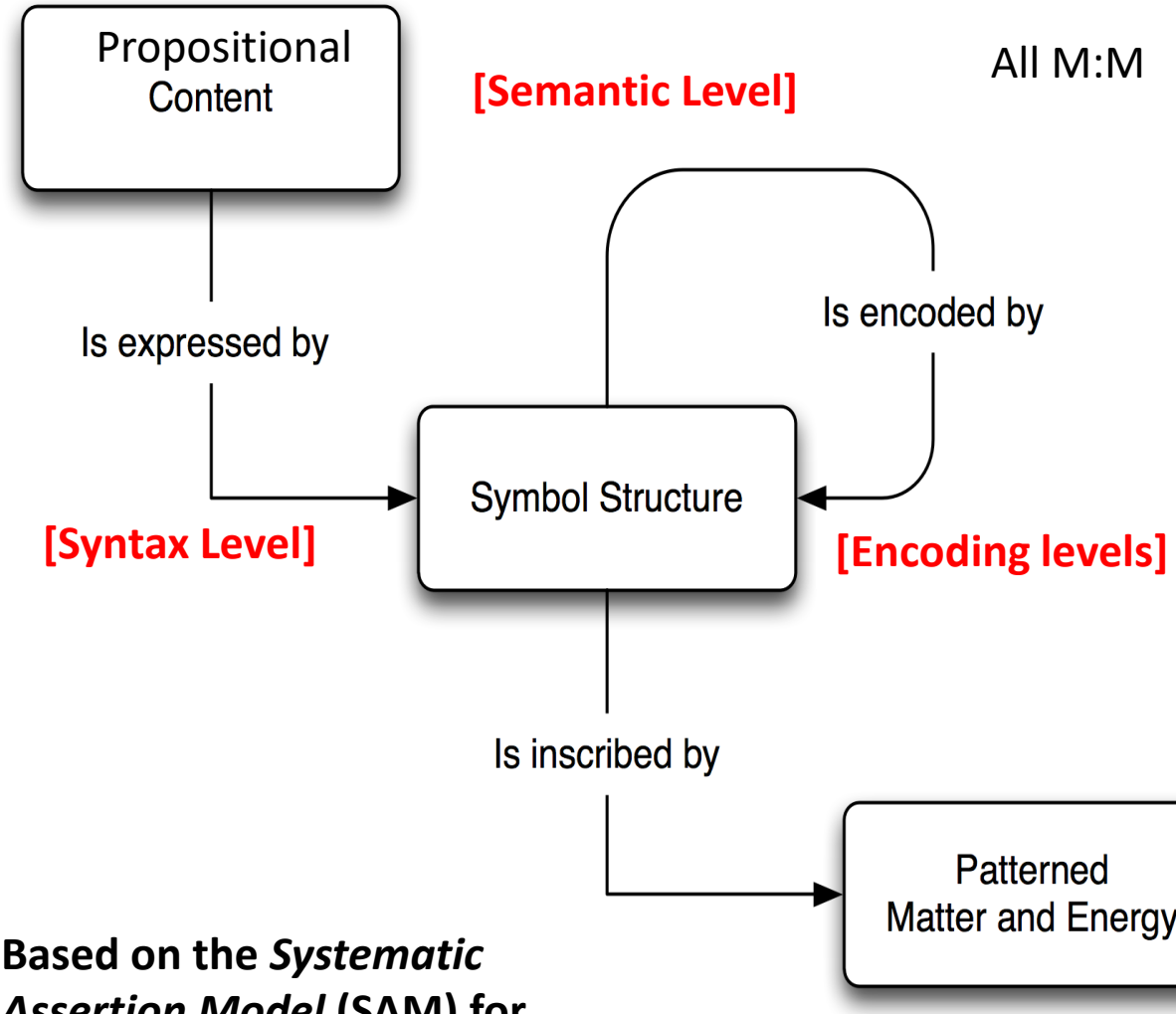
Identity and Representation Levels

How many of these levels are there ?

How do we name, define, and manage them?

How can they be identified and re-identified?

The Basic Representation Model (or FRBR refactored)



For example:

C1: propositions
expressed by…

S1: RDF triples
encoded by…

S2: RDF/XML
encoded by …

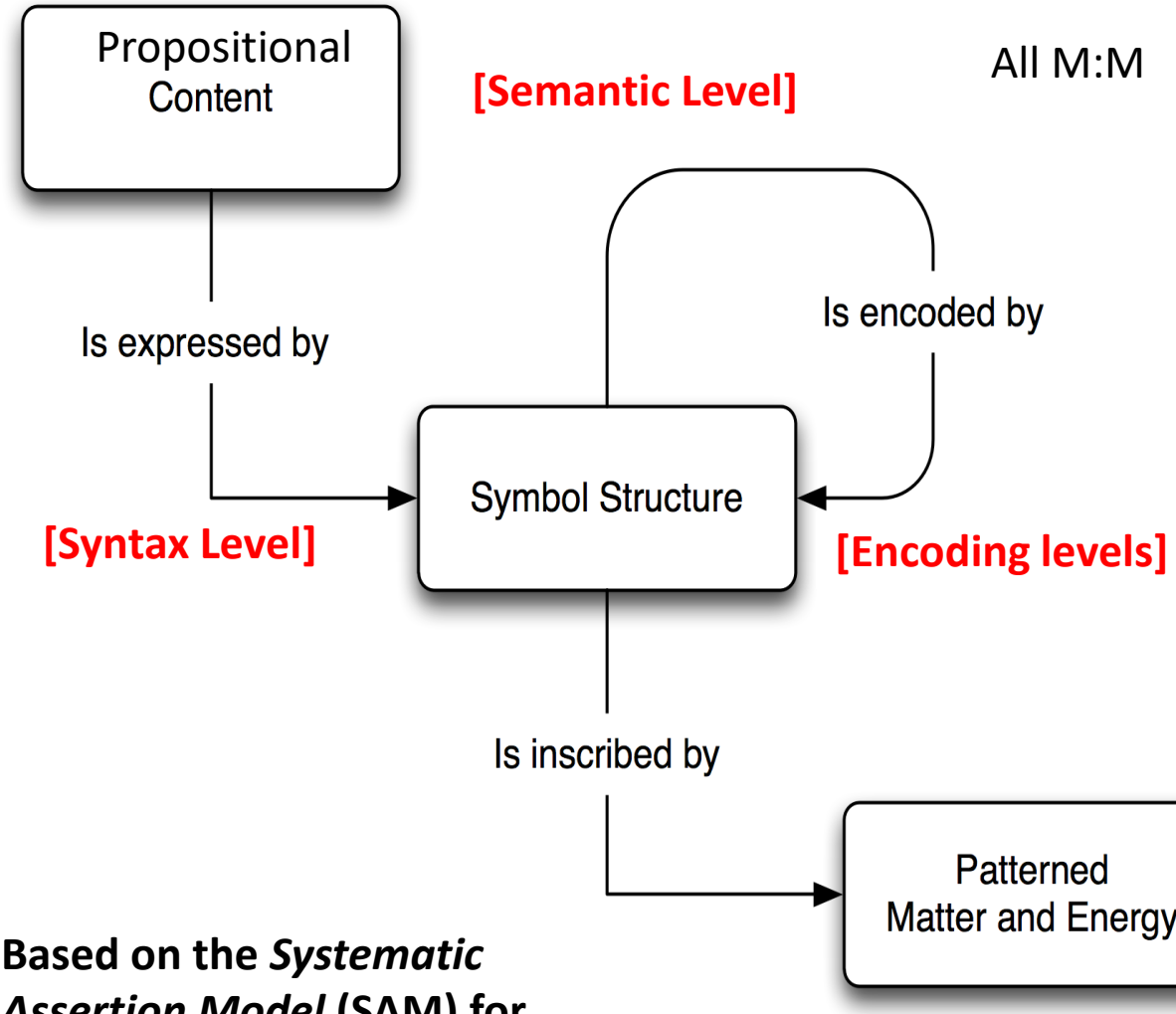
S3: Unicode characters
encoded by…

S4: UTF-8 bit streams
inscribed in…

M1: actual RAID array state

Based on the *Systematic Assertion Model* (SAM) for modeling datasets, developed by David Dubin et al.

So what are we identifying?



For example:

C1: **propositions**
expressed by…

S1: **RDF triples**
encoded by…

S2: **RDF/XML**
encoded by …

S3: **Unicode characters**
encoded by…

S4: **UTF-8 bit streams**
inscribed in…

M1: **actual RAID array state**

Based on the *Systematic Assertion Model* (SAM) for modeling datasets, developed by David Dubin et al.

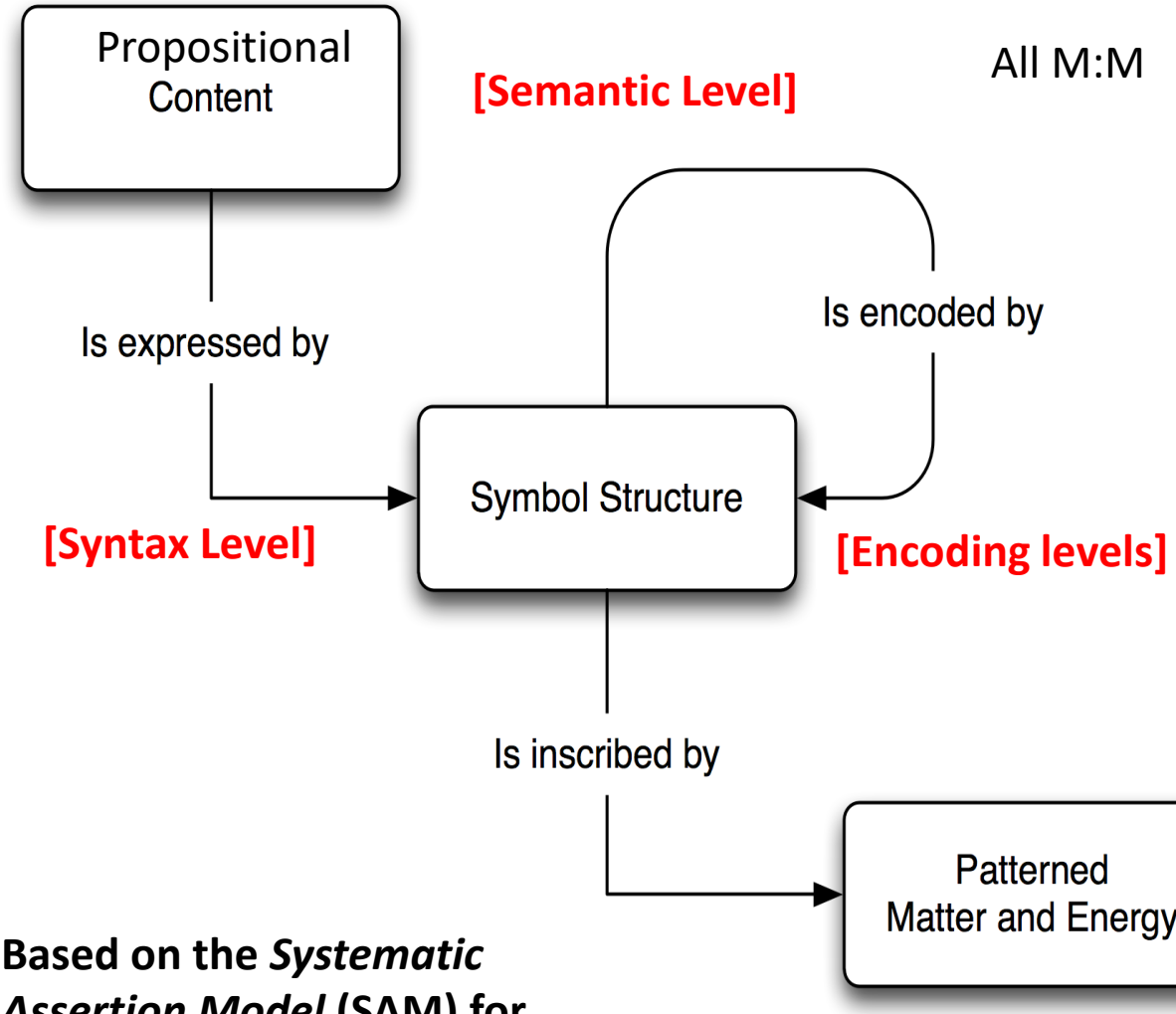
V3: *How* do we identify?

How do we identify sameness of propositions?

How do we identify sameness of syntax?

How do we identify sameness of encoding?

So what are we identifying?



For example:

C1: **propositions**
expressed by…

S1: **RDF triples**
encoded by…

S2: **RDF/XML**
encoded by …

S3: **Unicode characters**
encoded by…

S4: **UTF-8 bit streams**
inscribed in…

M1: **actual RAID array state**

Based on the *Systematic Assertion Model* (SAM) for modeling datasets, developed by David Dubin et al.

W*hat* are we identifying?

Propositions	Semantic level	
Symbols	Syntax level	
Symbols	Encoding level (1)	
Symbols	Encoding level (2)	
Symbols	Encoding level (3)	
. . .		
Symbols	Encoding level (n)	(e.g. 1s&0s, for inscription)

But operationally identification works from the bottom up.

We identify the bitstream or character sequence in a normal form in order to *indirectly* identify the higher level encodings, syntax, or propositional content.

Propositional Identity

Establishing that data sets have the same propositional content can be challenging.

Consider the case where data sets are using completely different modeling approaches.

In one case *relations* perhaps stored in a CSV file.
In the other case *RDF triples*, perhaps in N3.

Determining identity at the propositional level requires translating both data sets into a single data representation language,

[NB: *This has much in common with some of the strategies for data integration and data preservation we have seen.]*

. . . and then confirming that every proposition in one is also in the other.

This can be challenging, to say the least.

So before looking more closely at this hard case,
let's warm up by considering the simpler cases: encoding and syntax identity.

Encoding Identity

Establishing that two files contain the same sequence of bits is relatively straightforward in the current environment.

> We rely on standard tools that recognize the (intended) recorded sequence of 1s and 0s within the context on an operating system or storage medium and compare the two bitstreams.

Establishing that data sets contain the same sequence of meaningful octets or bytes is usually not particularly difficult either.

> Here we rely on tools and standards like UTF-8 and UTF-16 for recognizing the bytes within bitstreams, and again comparison is fairly straightforward

And for the most part interpreting those bytes as corresponding to integers or, directly or indirectly, to characters or other semantic tokens is fairly straightforward

> Here we rely on character encoding standards, such as EBCDIC, ASCII or Unicode to convert byte sequences into characters sequences that can be compared.

Syntax identity

Establishing that two data sets contain* the same *representation* of relations, triples, graphs, etc. is more challenging however.

> This is because so many variations may occur (tabs, spaces, abbreviations, etc.) that are considered irrelevant to syntactical identity, even when the same serialization language is being used.

And establishing that two data sets contain* the same relations, triples, graphs, etc. is of course even more challenging.

> This is because the same data structure can be represented in different serialization languages

*Now is as good as time as any to point out that the notion, used here and elsewhere, that data sets "contain" things is a useful idiom, but not only cannot be taken literally, but is not really used in the same sense throughout: data sets do not contain bits, bytes, characters, syntactical representations, and relations all in the same sense of contain in each case.

Generalizing

The general problem:

The things we want to identify can be expressed or encoded in different ways,

We are typically looking at a 1:M* relationship between

- Propositions and representations
- Representations and encodings
- Encodings and encodings and encodings . . . [lather, rinse, repeat]

and so we can't easily use the lower level instantiations to determine identity of the upper levels entities

[in other words: variation at the lower level does not necessarily entail variation at the upper levels]

But, there is a solution to this problem.

(in the next video)

*Actually the relationships are M:M because of the arbitrary nature of representation described in the Data Concepts videos, that is: depending on conventions (like standards) and other social circumstances (agreements, decisions, intentions, expectations) the same (e.g.) encoding can encode multiple representations. But most of the time in identity problems we can ignore this and rely on a shared understanding of the relevant concepts.

V4: An Example: Canonicalization

Here we give an example of *canonicalization*, a central strategy for determining identity of representation or propositional content.

Canonicalization

Canonicalization is a technique for determining representational identity

and is a reasonable proxy for propositional identity.

It is most directly applicable when the representation syntax of the two data sets is the same, but there are

-- variations in encoding

-- alternative synonymous variation within a single language

XML as an example

Imagine two XML files.

Assume they both define the same data structure
(the same labelled graph with attribute/value pairs).

But they could have

- different pretty printing conventions
  - end tags indented or not
  - spaces vs tabs
- Or different character encodings
  - [or different line end encodings (0x0A, 0x0D, 0x0D+0x0A. . .)]
- Or they may have multiple ways to define the same data structure, e.g,:
  - global defaults for attribute values vs local specification
  - random order for attribute/value pairs on element vs . . . Some other random order
  - (and so on).

These quite different files can specify the exact some data structure, but how can we tell?

- You can't use a checksum to determine whether they are logically equivalent
  - A checksum only determines bit, byte, or (sometimes) character "fixity"

This is the problem canonicalization goes after.

A canonicalization recipe

Here is a simple example of how XML canonicalization might work:

To determine whether or not two XML files determine the same data structure.

1. Convert to a single character encoding and normalize line ends.
2. Remove all comments, tabs, non-significant spaces, etc.
3. Propagate all attribute defaults indicated in the schema to the elements themselves
4. Put attribute/value pairs on elements in alpha order
5. Expand all character references
6. Remove any internal schema or declarations.
7. Now test to see if character sequences are identical.

Canonicalized normal forms as proxies for propositional content

[Ok, I see you have noticed we seem to have walked away from identifying propositional content]

Are we giving up?

No, but discretion is the better part of valor, so we are compromising:

When a single data description language is being used a fixity checksum on files in canonical normal form is for all practical purposes our best shot at sameness of propositional content.

Alternative approaches can be imagined, but they are difficult to implement and it isn't clear the results will be much better.

Readings

Required:

Persistent Identifiers, Fixity and Checksums, in *The Digital Preservation Handbook*, Digital Preservation Coalition, 2017.

On the utility of identification schemes for digital earth science data: An assessment and recommendations. Duerr, R. E., Downs, R. R., Tilmes, C., Barkstrom, B., Lenhardt, W. C., Glassy, J., Bermudez, L. E., & Slaughter, P. (2011). *Earth Science Informatics*.

=> Clifford A. Lynch, "Canonicalization: A Fundamental Tool To Facilitate Preservation and Management of Digital Information," *D-lib Magazine*, 5:9 (September 1999).

=> *Canonical XML*. Version 1.0. W3C Recommendation, John Boyer, March 2001. Latest version: http://www.w3.org/TR/xml-c14n.