Identity and Identifiers

Contents

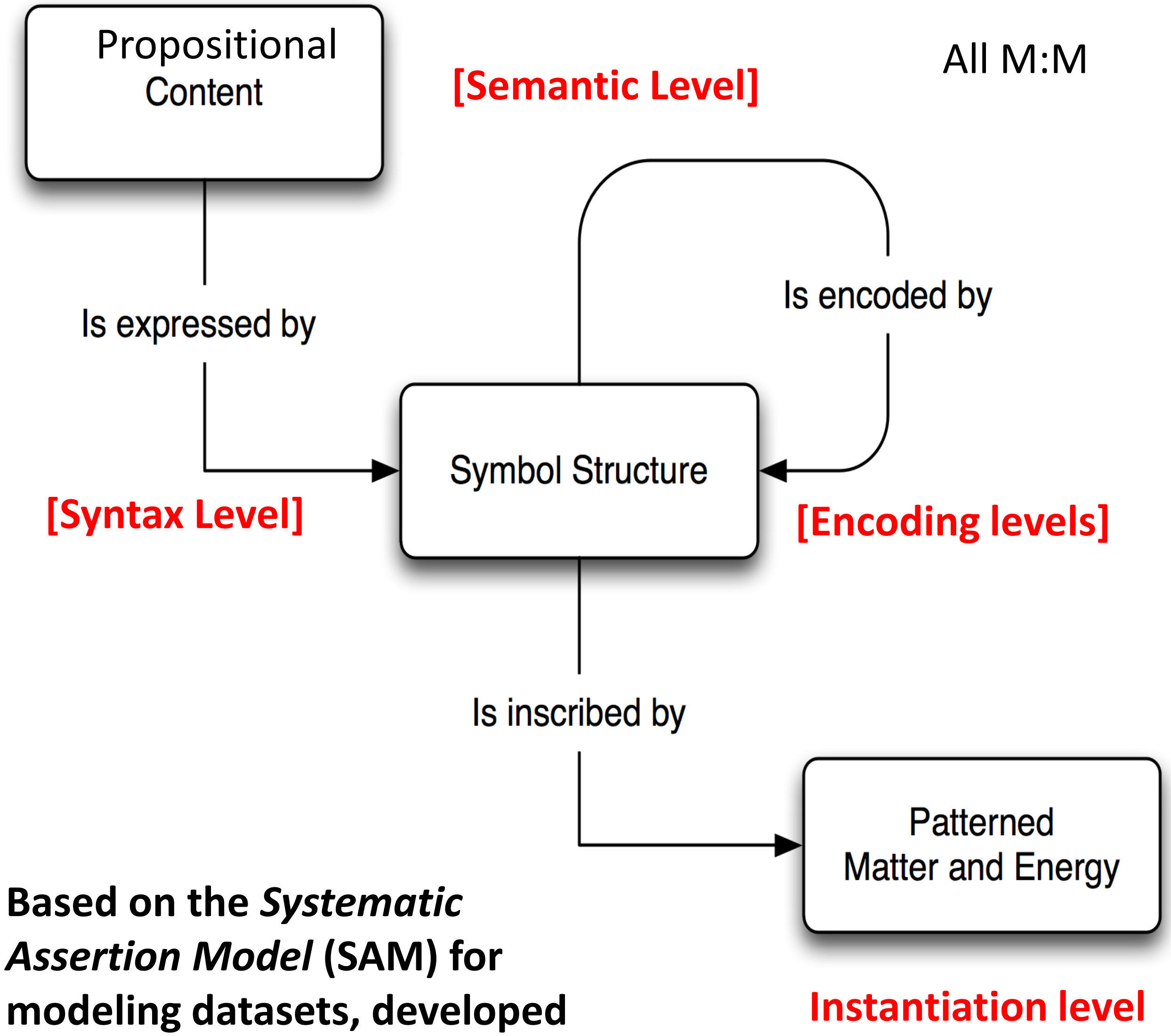
V3: *How* do we identify?

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.