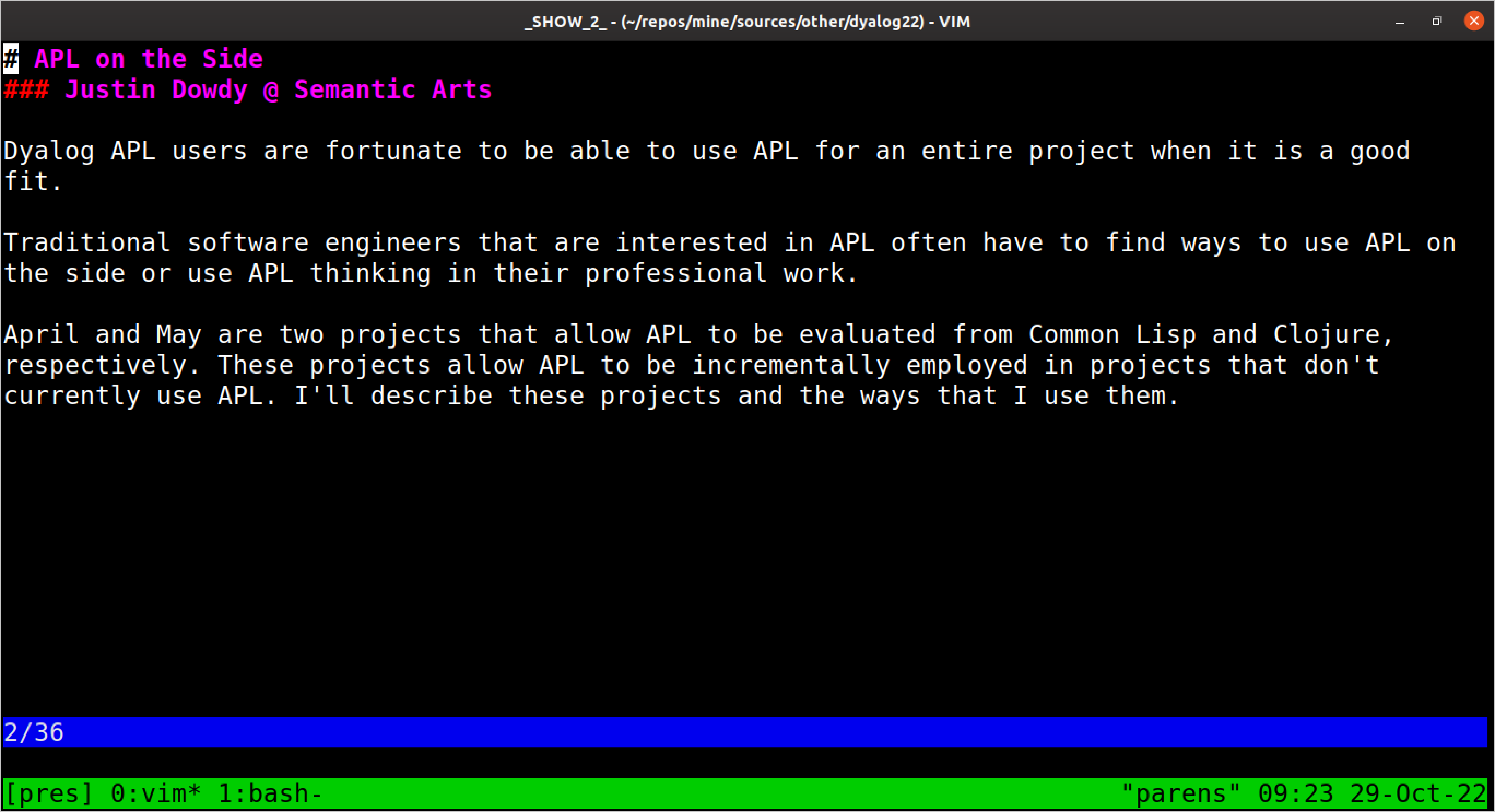
```
_SHOW_2_ - (~/repos/mine/sources/other/dyalog22) - VIM

# APL on the Side
### Justin Dowdy @ Semantic Arts

1/36

[pres] 0:vim* 1:bash-                                    "parens" 09:23 29-Oct-22
```

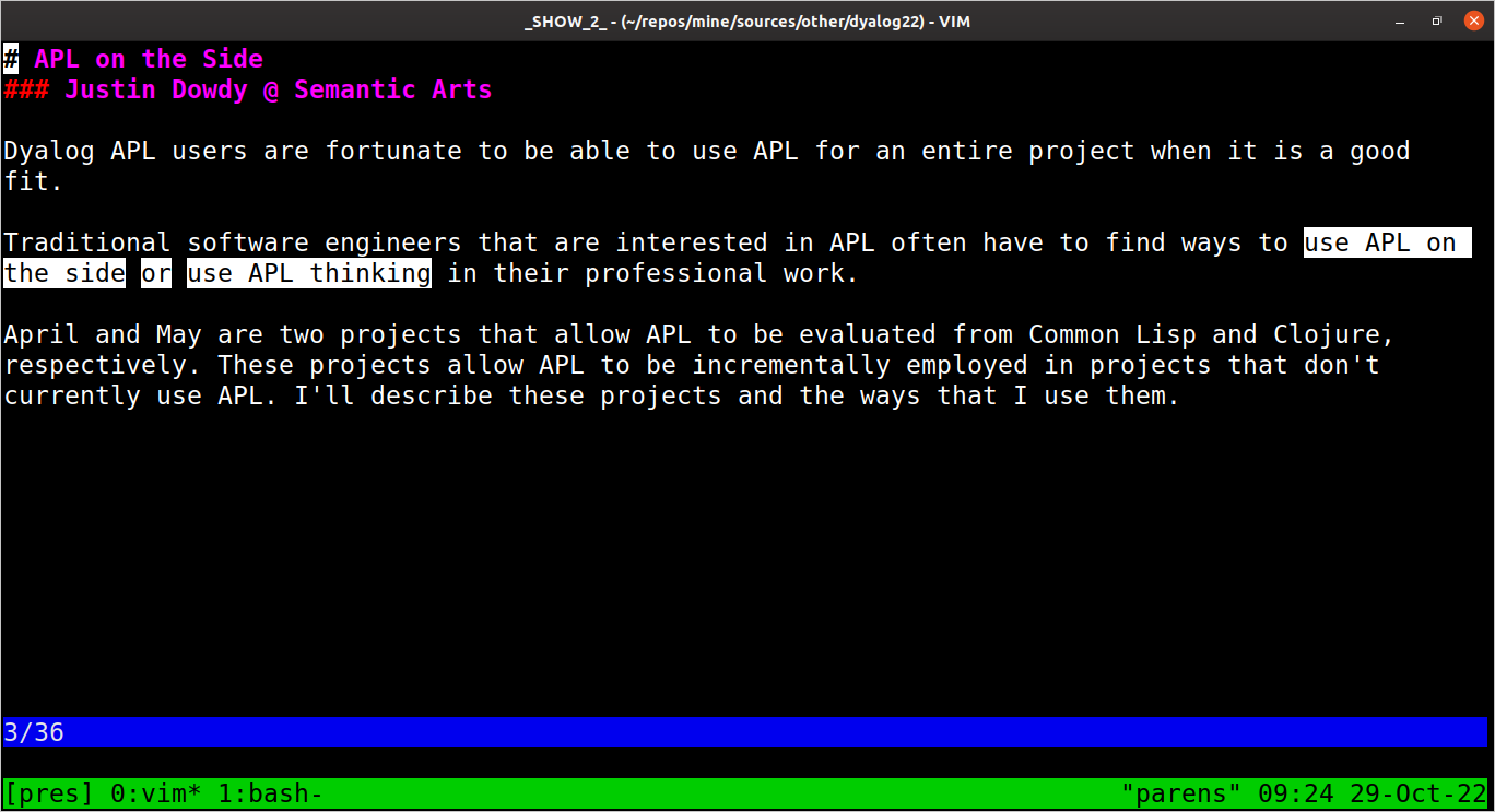


APL on the Side
Justin Dowdy @ Semantic Arts

Dyalog APL users are fortunate to be able to use APL for an entire project when it is a good fit.

Traditional software engineers that are interested in APL often have to find ways to use APL on the side or use APL thinking in their professional work.

April and May are two projects that allow APL to be evaluated from Common Lisp and Clojure, respectively. These projects allow APL to be incrementally employed in projects that don't currently use APL. I'll describe these projects and the ways that I use them.



APL on the Side
Justin Dowdy @ Semantic Arts

Dyalog APL users are fortunate to be able to use APL for an entire project when it is a good fit.

Traditional software engineers that are interested in APL often have to find ways to use APL on the side or use APL thinking in their professional work.

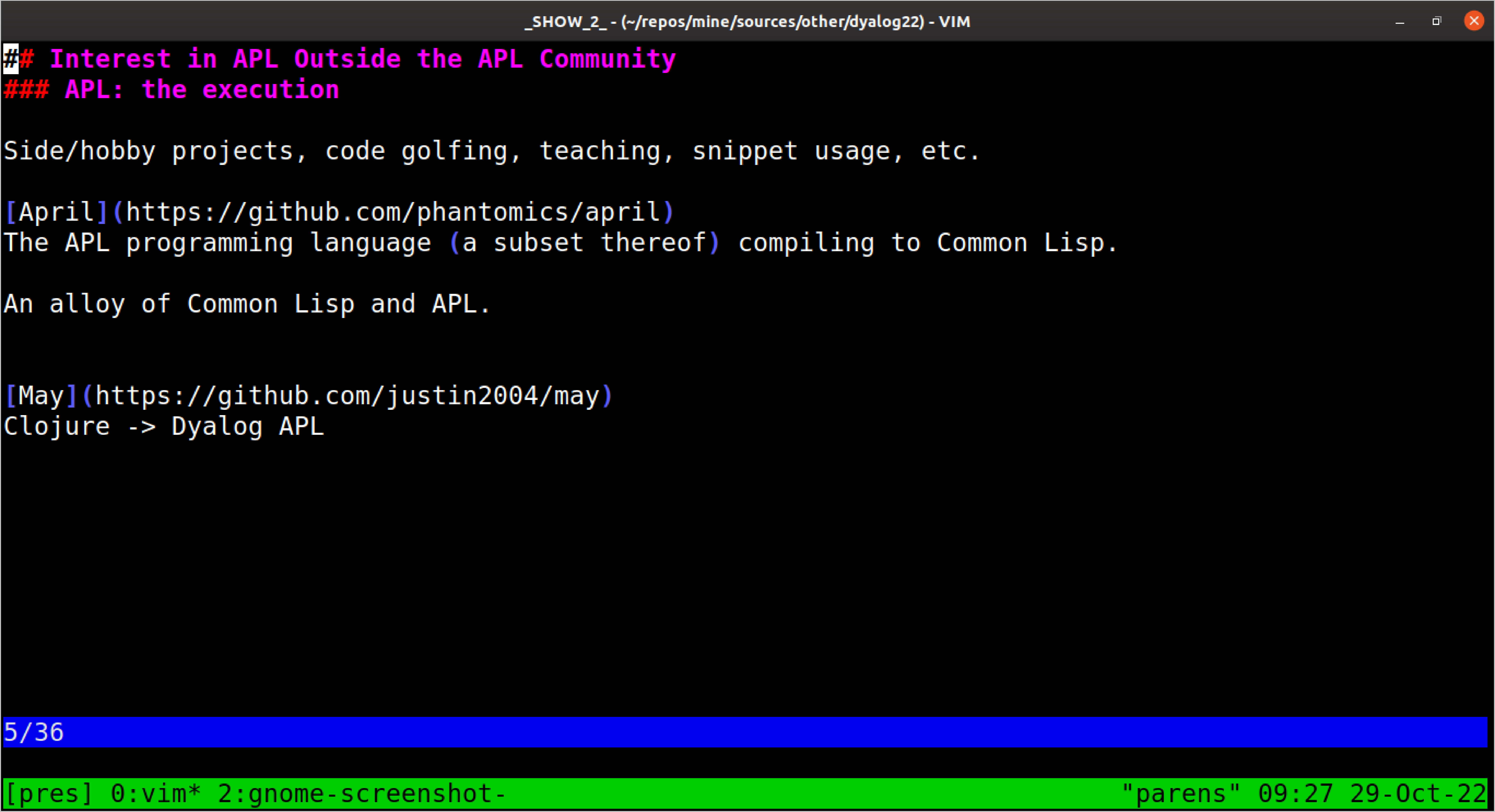
April and May are two projects that allow APL to be evaluated from Common Lisp and Clojure, respectively. These projects allow APL to be incrementally employed in projects that don't currently use APL. I'll describe these projects and the ways that I use them.

Interest in APL Outside the APL Community

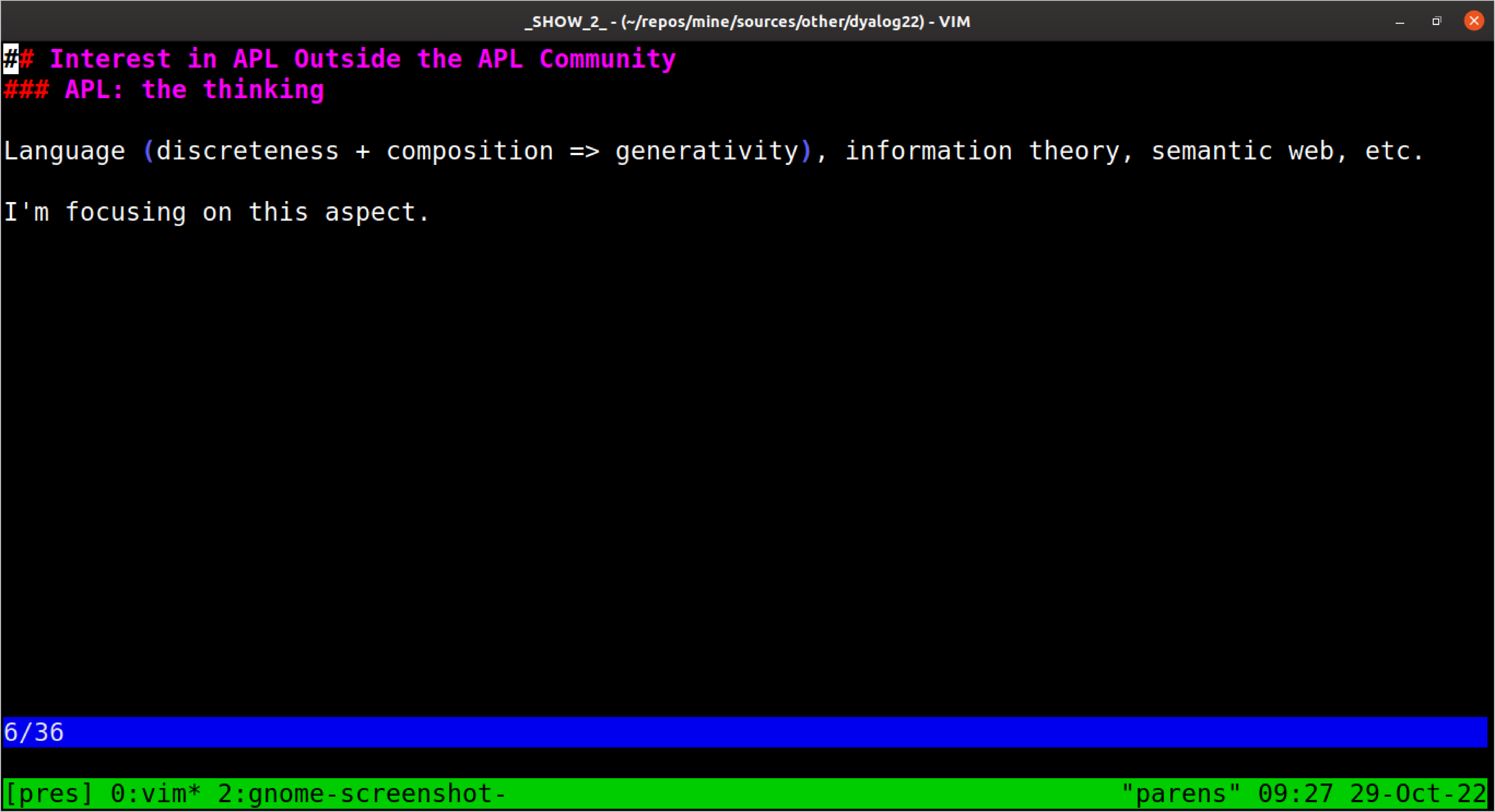
APL: the execution

APL: the thinking

```
4/36
[pres] 0:vim* 2:bash-                                            "parens" 09:26 29-Oct-22
```



```
## Interest in APL Outside the APL Community
### APL: the execution

Side/hobby projects, code golfing, teaching, snippet usage, etc.

[April](https://github.com/phantomics/april)
The APL programming language (a subset thereof) compiling to Common Lisp.

An alloy of Common Lisp and APL.


[May](https://github.com/justin2004/may)
Clojure -> Dyalog APL









5/36

[pres] 0:vim* 2:gnome-screenshot-                              "parens" 09:27 29-Oct-22
```


_SHOW_2_ - (~/repos/mine/sources/other/dyalog22) - VIM

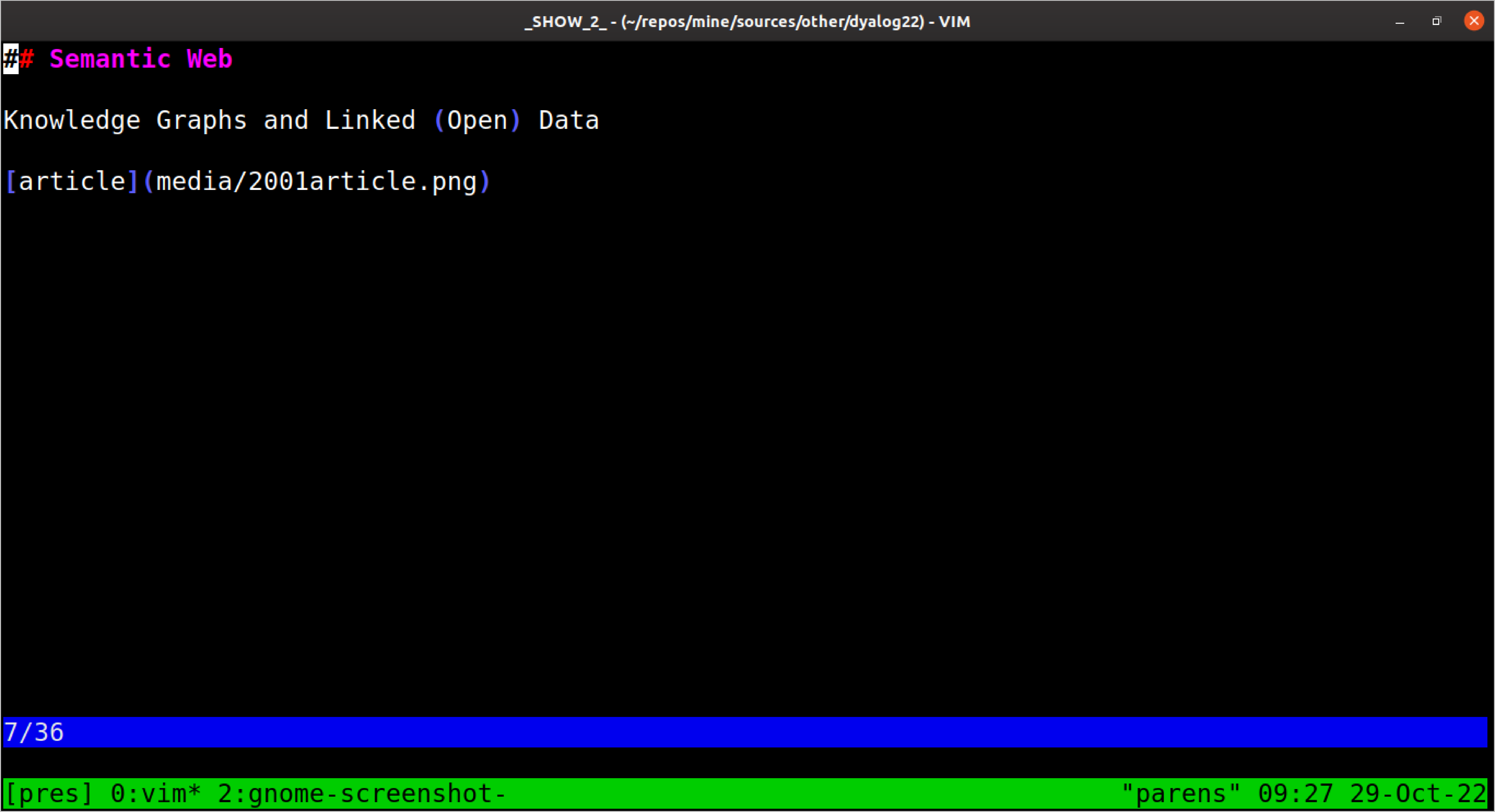
Interest in APL Outside the APL Community

APL: the thinking

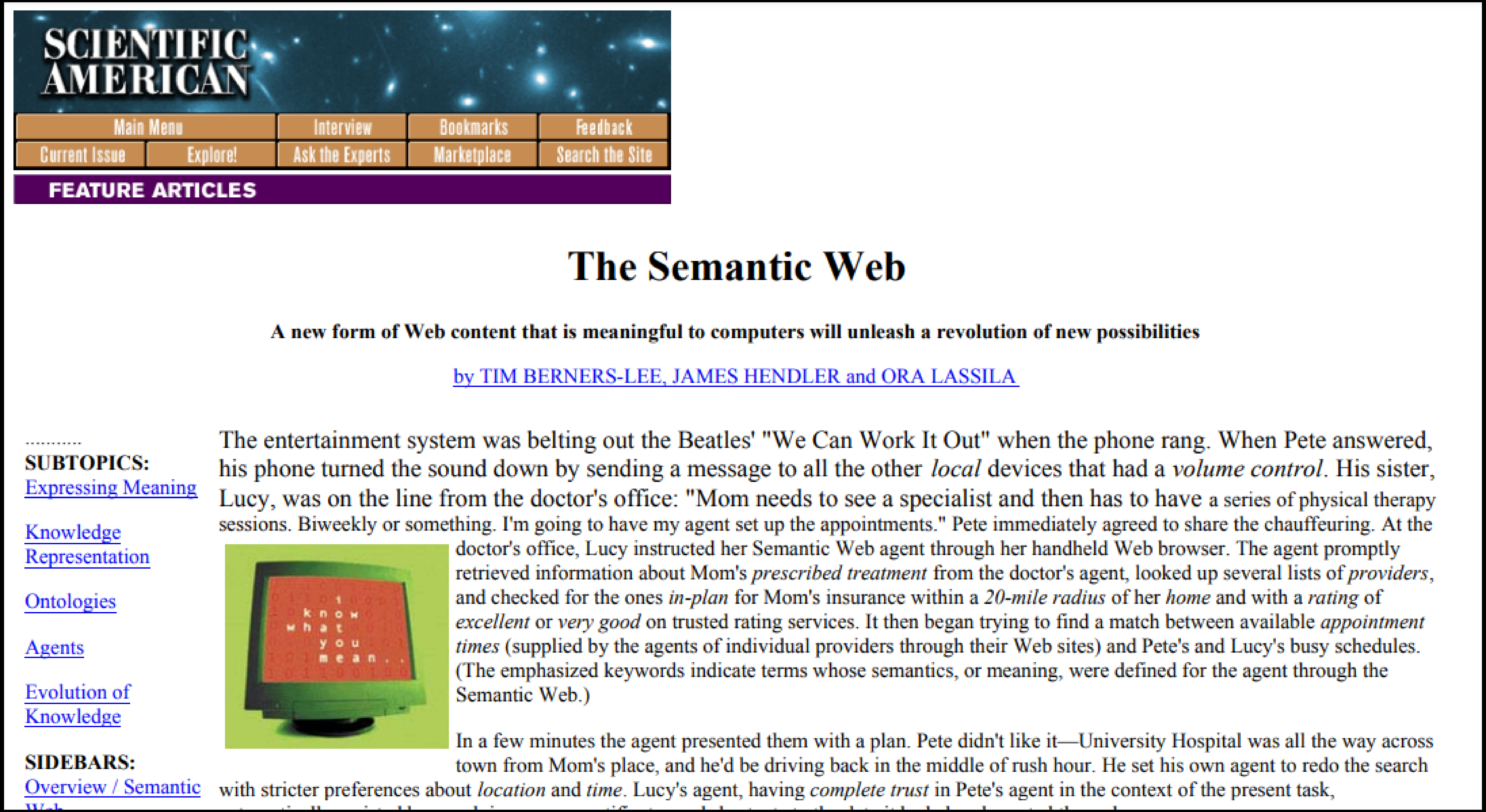
Language (discreteness + composition => generativity), information theory, semantic web, etc.

I'm focusing on this aspect.

6/36
[pres] 0:vim* 2:gnome-screenshot-
"parens" 09:27 29-Oct-22



```
## Semantic Web

Knowledge Graphs and Linked (Open) Data

[article](media/2001article.png)

7/36

[pres] 0:vim* 2:gnome-screenshot-                    "parens" 09:27 29-Oct-22
```



SCIENTIFIC AMERICAN

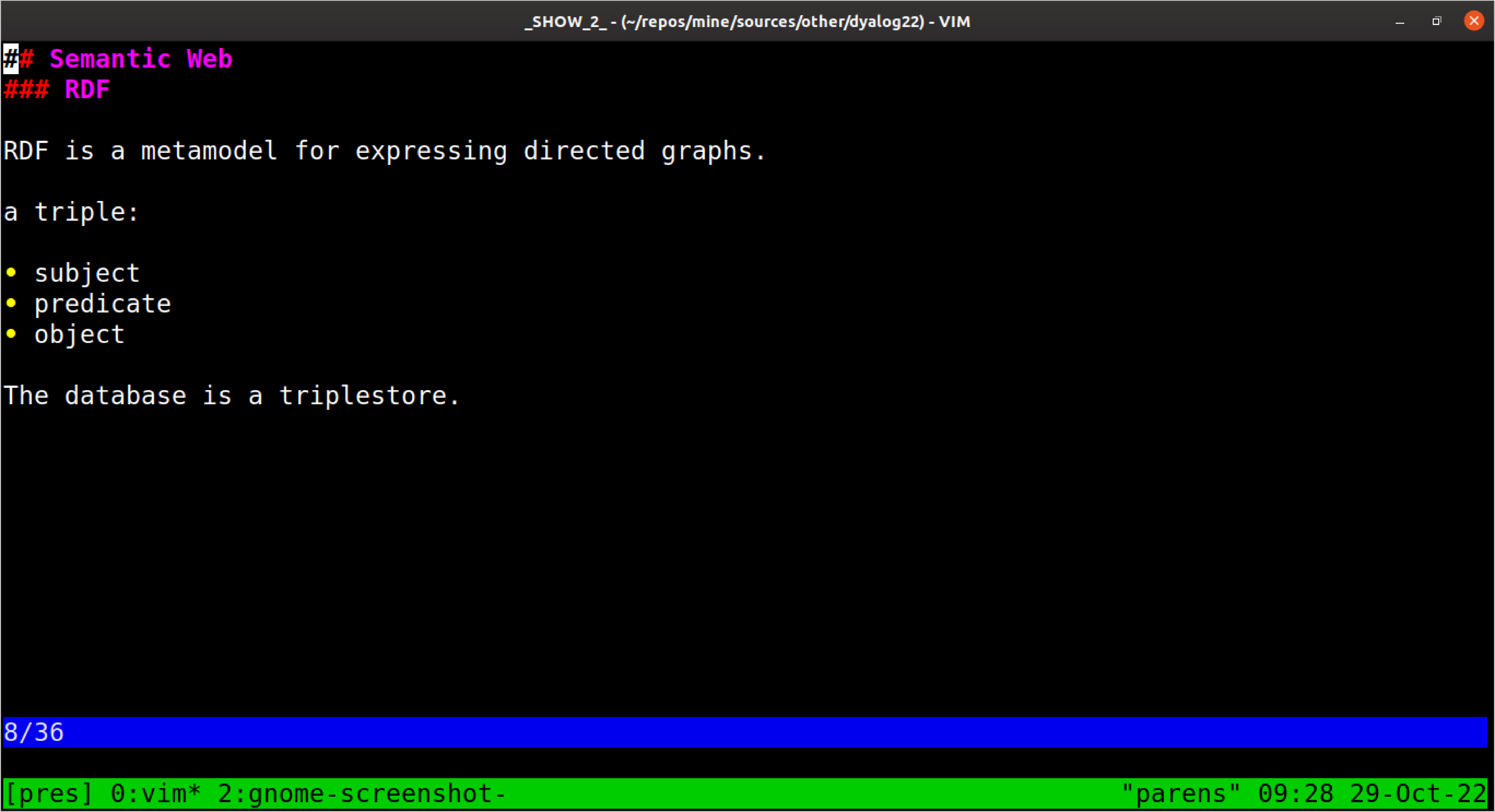
The Semantic Web

A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities

by TIM BERNERS-LEE, JAMES HENDLER and ORA LASSILA

The entertainment system was belting out the Beatles' "We Can Work It Out" when the phone rang. When Pete answered, his phone turned the sound down by sending a message to all the other *local* devices that had a *volume control*. His sister, Lucy, was on the line from the doctor's office: "Mom needs to see a specialist and then has to have a series of physical therapy sessions. Biweekly or something. I'm going to have my agent set up the appointments." Pete immediately agreed to share the chauffeuring. At the doctor's office, Lucy instructed her Semantic Web agent through her handheld Web browser. The agent promptly retrieved information about Mom's *prescribed treatment* from the doctor's agent, looked up several lists of *providers*, and checked for the ones *in-plan* for Mom's insurance within a *20-mile radius* of her *home* and with a *rating* of *excellent* or *very good* on trusted rating services. It then began trying to find a match between available *appointment times* (supplied by the agents of individual providers through their Web sites) and Pete's and Lucy's busy schedules. (The emphasized keywords indicate terms whose semantics, or meaning, were defined for the agent through the Semantic Web.)

In a few minutes the agent presented them with a plan. Pete didn't like it—University Hospital was all the way across town from Mom's place, and he'd be driving back in the middle of rush hour. He set his own agent to redo the search with stricter preferences about *location* and *time*. Lucy's agent, having *complete trust* in Pete's agent in the context of the present task,


_SHOW_2_ - (~/repos/mine/sources/other/dyalog22) - VIM

Semantic Web
RDF

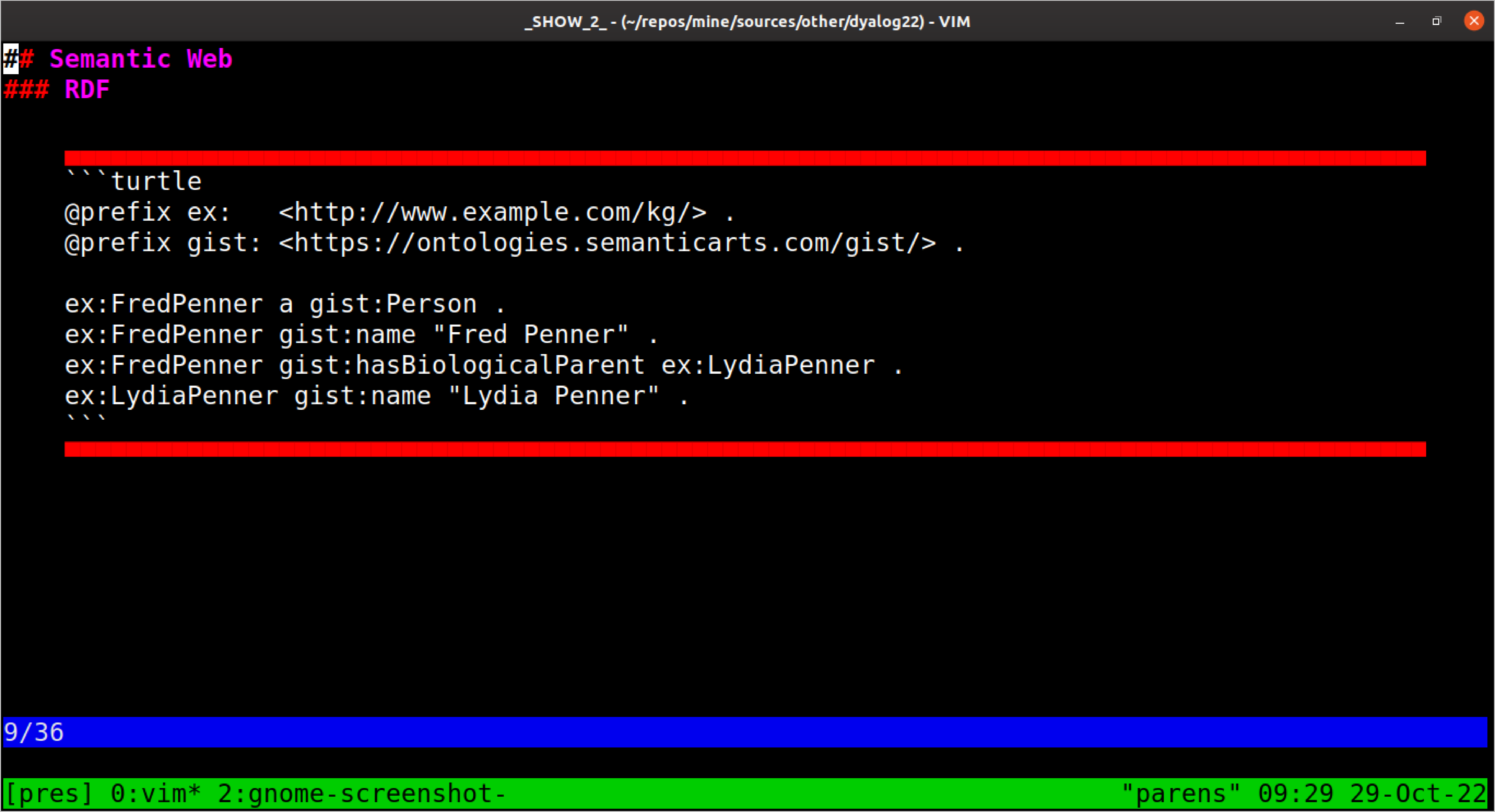
RDF is a metamodel for expressing directed graphs.

a triple:

- subject
- predicate
- object

The database is a triplestore.

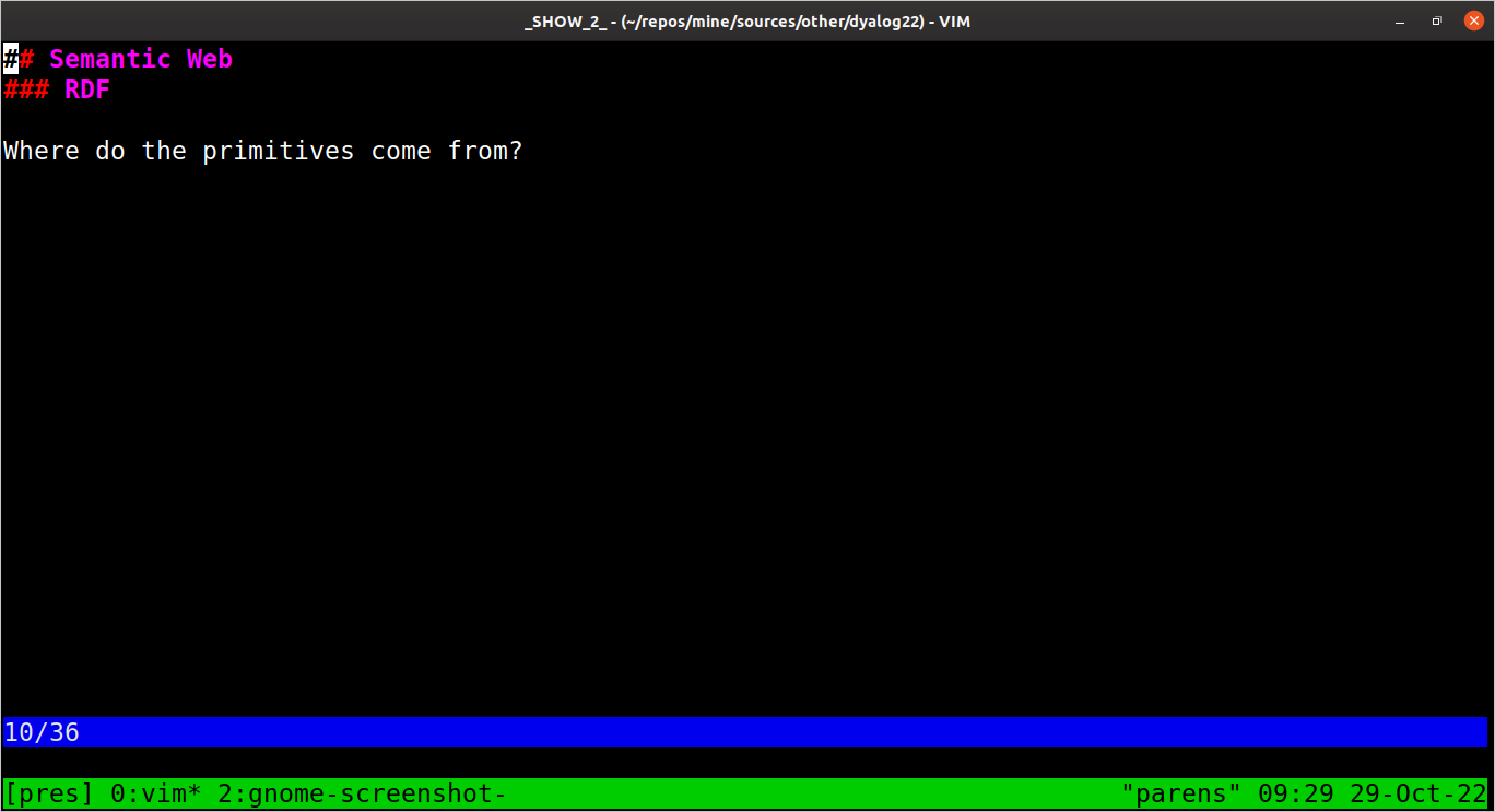
8/36
[pres] 0:vim* 2:gnome-screenshot-
"parens" 09:28 29-Oct-22



~~~
## Semantic Web
### RDF


    ```turtle
    @prefix ex:   <http://www.example.com/kg/> .
    @prefix gist: <https://ontologies.semanticarts.com/gist/> .

    ex:FredPenner a gist:Person .
    ex:FredPenner gist:name "Fred Penner" .
    ex:FredPenner gist:hasBiologicalParent ex:LydiaPenner .
    ex:LydiaPenner gist:name "Lydia Penner" .
    ```
~~~

```
9/36
```

```
[pres] 0:vim* 2:gnome-screenshot-                                    "parens" 09:29 29-Oct-22
```

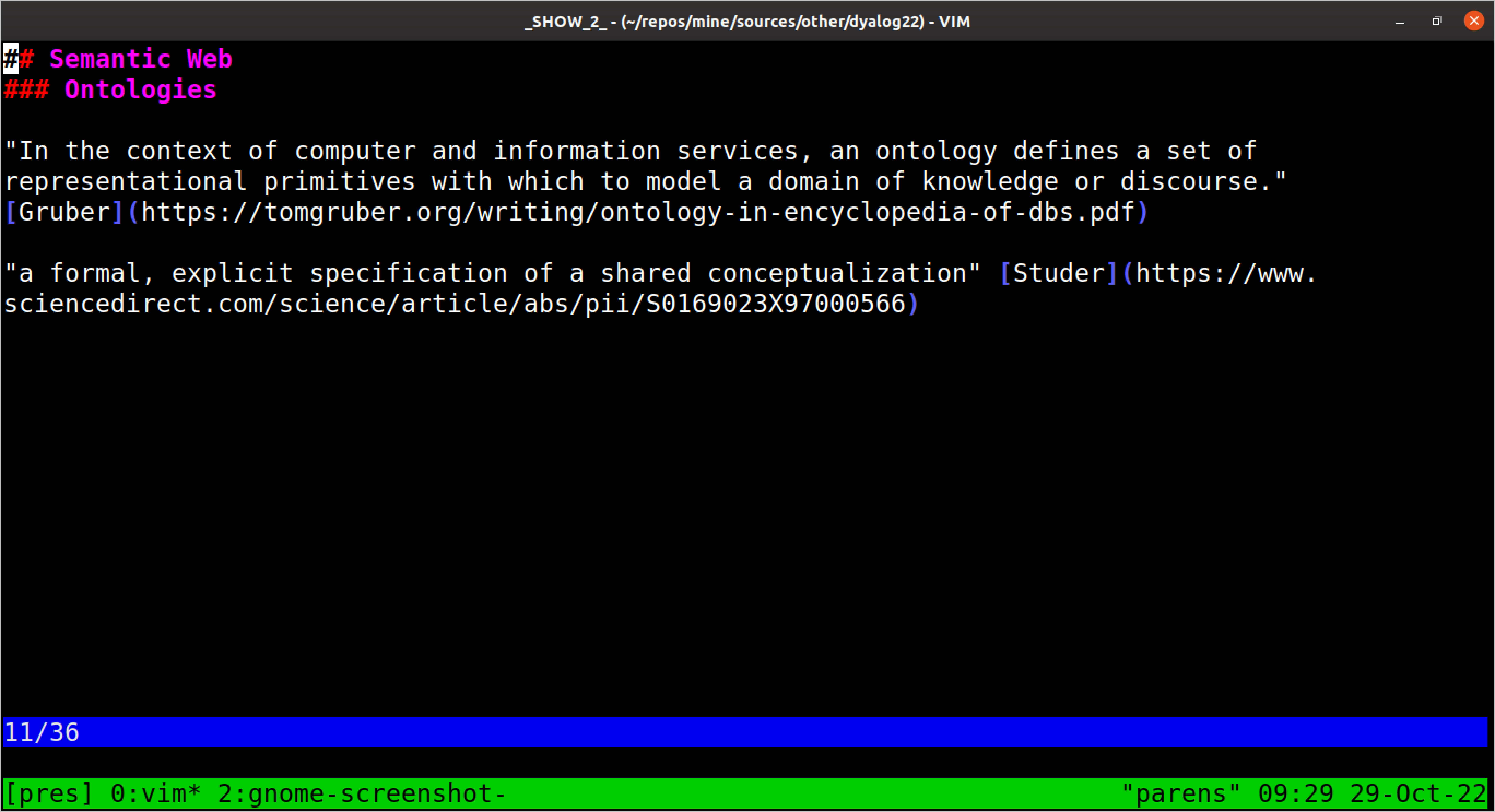


Semantic Web
RDF

Where do the primitives come from?

```
10/36
```

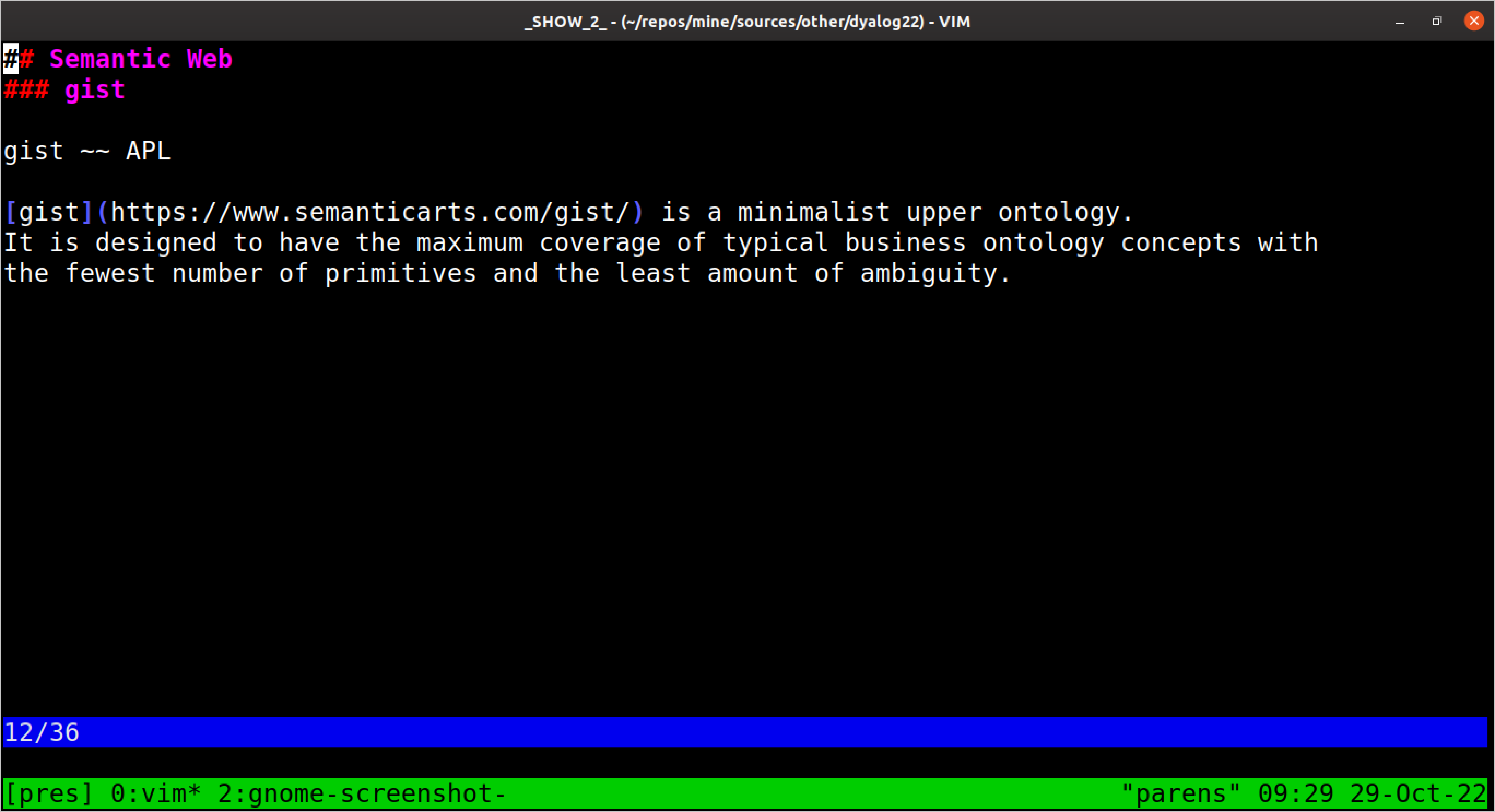
```
[pres] 0:vim* 2:gnome-screenshot-                              "parens" 09:29 29-Oct-22
```



```
## Semantic Web
### Ontologies

"In the context of computer and information services, an ontology defines a set of
representational primitives with which to model a domain of knowledge or discourse."
[Gruber](https://tomgruber.org/writing/ontology-in-encyclopedia-of-dbs.pdf)

"a formal, explicit specification of a shared conceptualization" [Studer](https://www.
sciencedirect.com/science/article/abs/pii/S0169023X97000566)
```

```
11/36
```

```
[pres] 0:vim* 2:gnome-screenshot-                                   "parens" 09:29 29-Oct-22
```

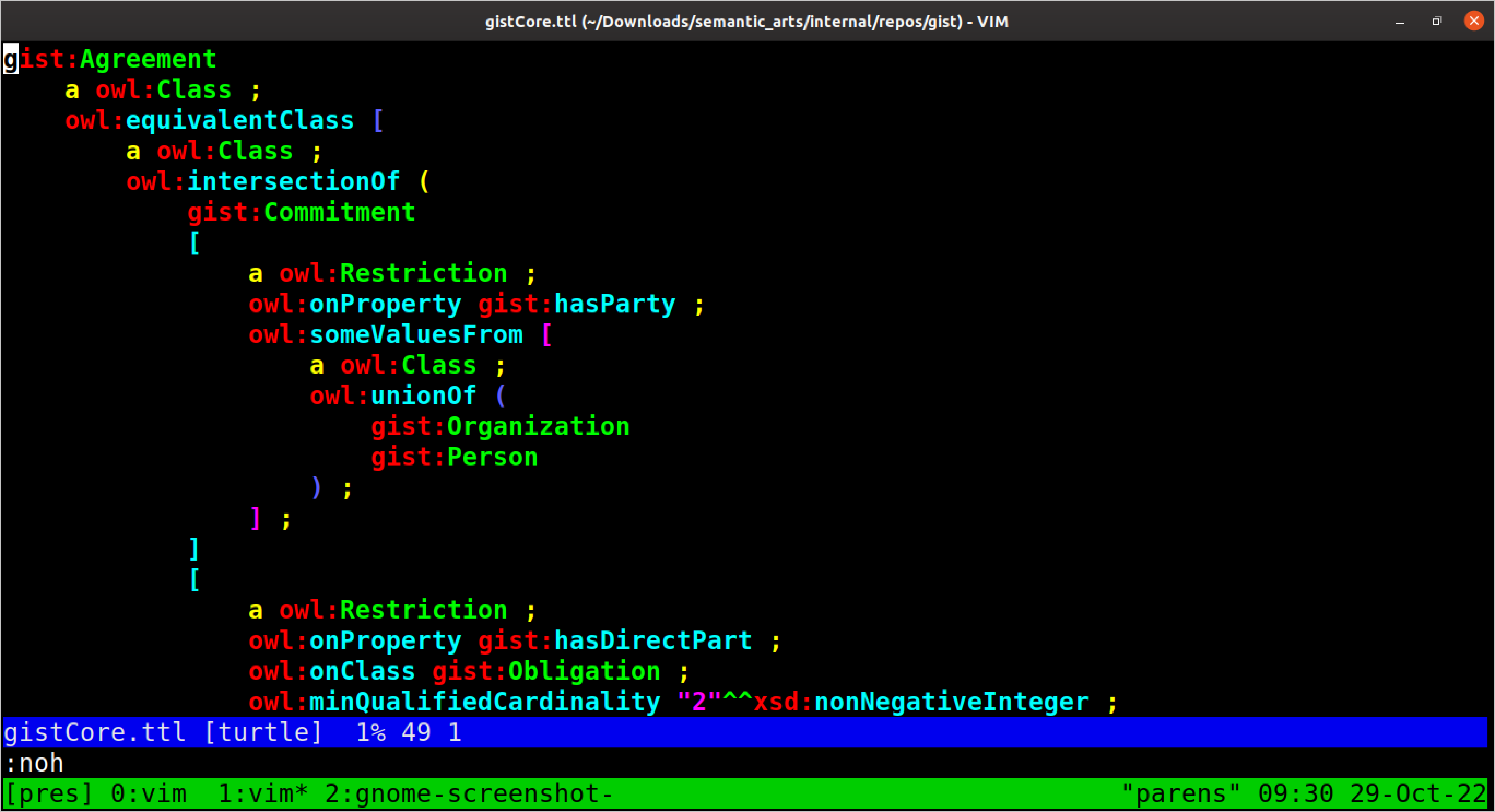

_SHOW_2_ - (~/repos/mine/sources/other/dyalog22) - VIM

Semantic Web
gist

gist ~~ APL

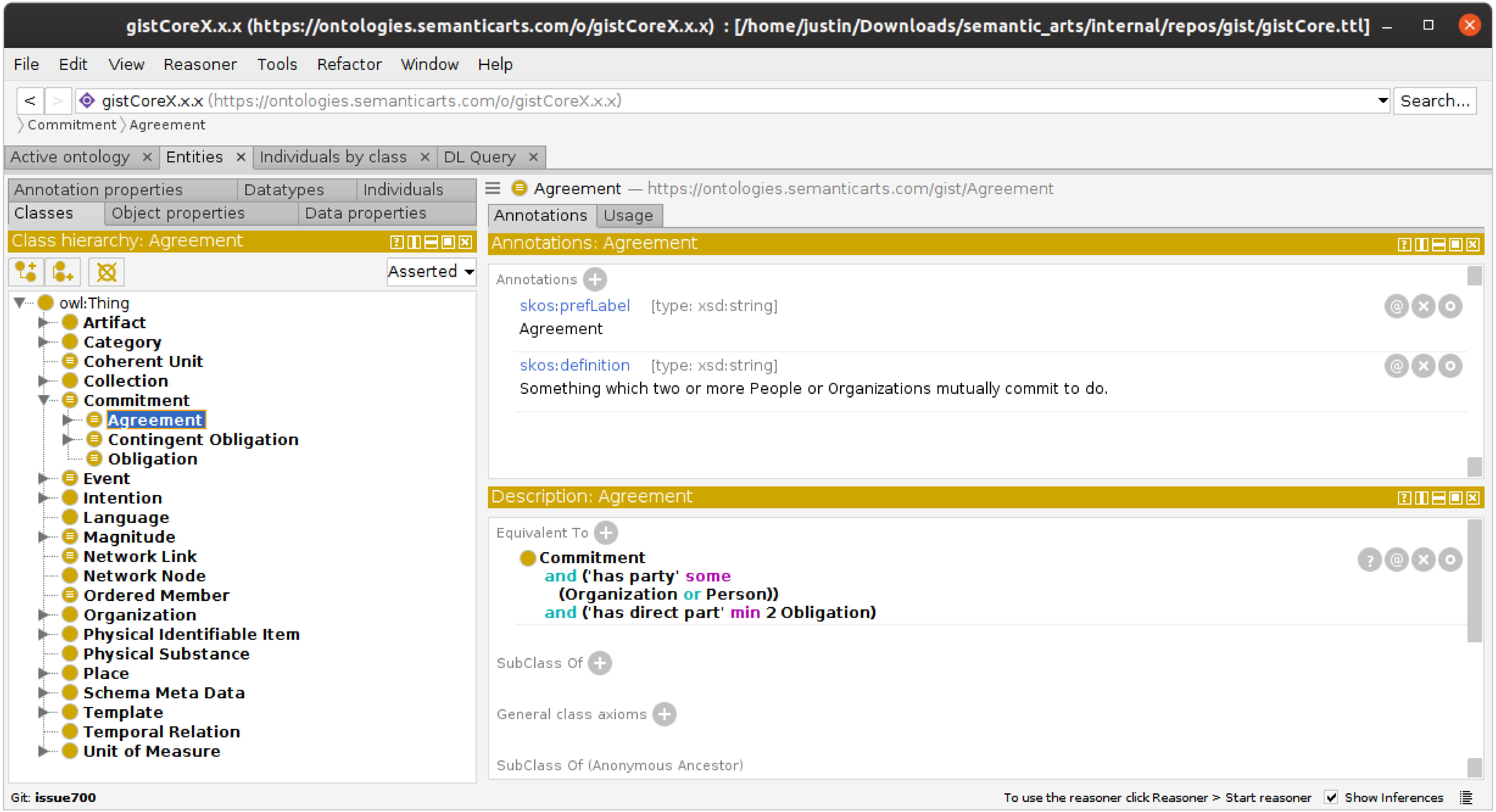
[gist](https://www.semanticarts.com/gist/) is a minimalist upper ontology.
It is designed to have the maximum coverage of typical business ontology concepts with
the fewest number of primitives and the least amount of ambiguity.

12/36
[pres] 0:vim* 2:gnome-screenshot-
"parens" 09:29 29-Oct-22



gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

```
gist:Agreement
    a owl:Class ;
    owl:equivalentClass [
        a owl:Class ;
        owl:intersectionOf (
            gist:Commitment
            [
                a owl:Restriction ;
                owl:onProperty gist:hasParty ;
                owl:someValuesFrom [
                    a owl:Class ;
                    owl:unionOf (
                        gist:Organization
                        gist:Person
                    ) ;
                ] ;
            ]
            [
                a owl:Restriction ;
                owl:onProperty gist:hasDirectPart ;
                owl:onClass gist:Obligation ;
                owl:minQualifiedCardinality "2"^^xsd:nonNegativeInteger ;
gistCore.ttl [turtle]  1% 49 1
:noh
[pres] 0:vim  1:vim* 2:gnome-screenshot-                    "parens" 09:30 29-Oct-22
```



owl:Thing
- Artifact
- Category
- Coherent Unit
- Collection
- Commitment
  - Agreement
  - Contingent Obligation
  - Obligation
- Event
- Intention
- Language
- Magnitude
- Network Link
- Network Node
- Ordered Member
- Organization
- Physical Identifiable Item
- Physical Substance
- Place
- Schema Meta Data
- Template
- Temporal Relation
- Unit of Measure

Agreement — https://ontologies.semanticarts.com/gist/Agreement

Annotations

skos:prefLabel [type: xsd:string]
Agreement

skos:definition [type: xsd:string]
Something which two or more People or Organizations mutually commit to do.

Equivalent To

```
Commitment
 and ('has party' some
   (Organization or Person))
 and ('has direct part' min 2 Obligation)
```

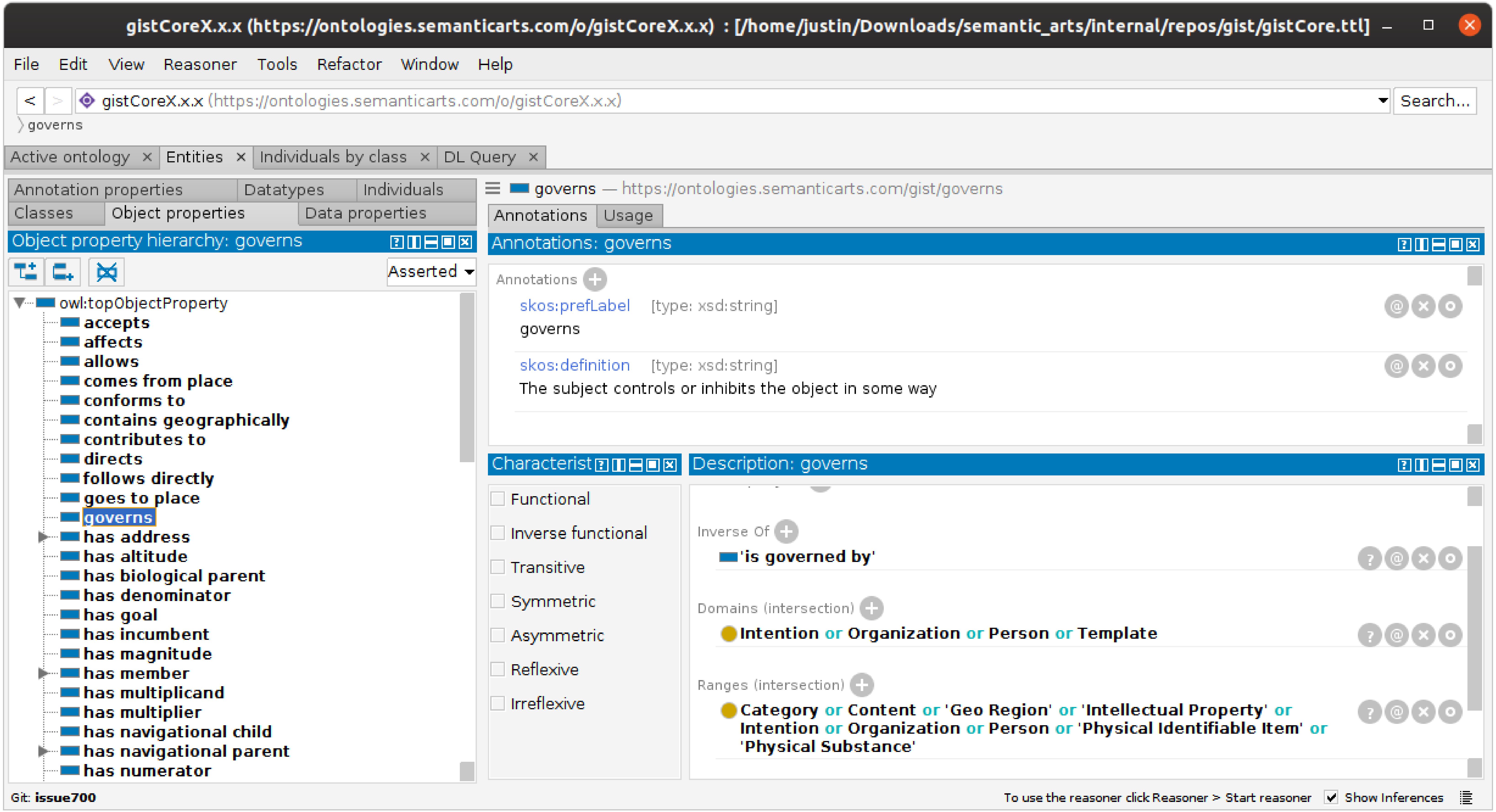
SubClass Of

General class axioms

SubClass Of (Anonymous Ancestor)

Git: issue700

To use the reasoner click Reasoner > Start reasoner ☑ Show Inferences



governs — https://ontologies.semanticarts.com/gist/governs

owl:topObjectProperty
- accepts
- affects
- allows
- comes from place
- conforms to
- contains geographically
- contributes to
- directs
- follows directly
- goes to place
- governs
- has address
- has altitude
- has biological parent
- has denominator
- has goal
- has incumbent
- has magnitude
- has member
- has multiplicand
- has multiplier
- has navigational child
- has navigational parent
- has numerator

Annotations

skos:prefLabel [type: xsd:string]
governs

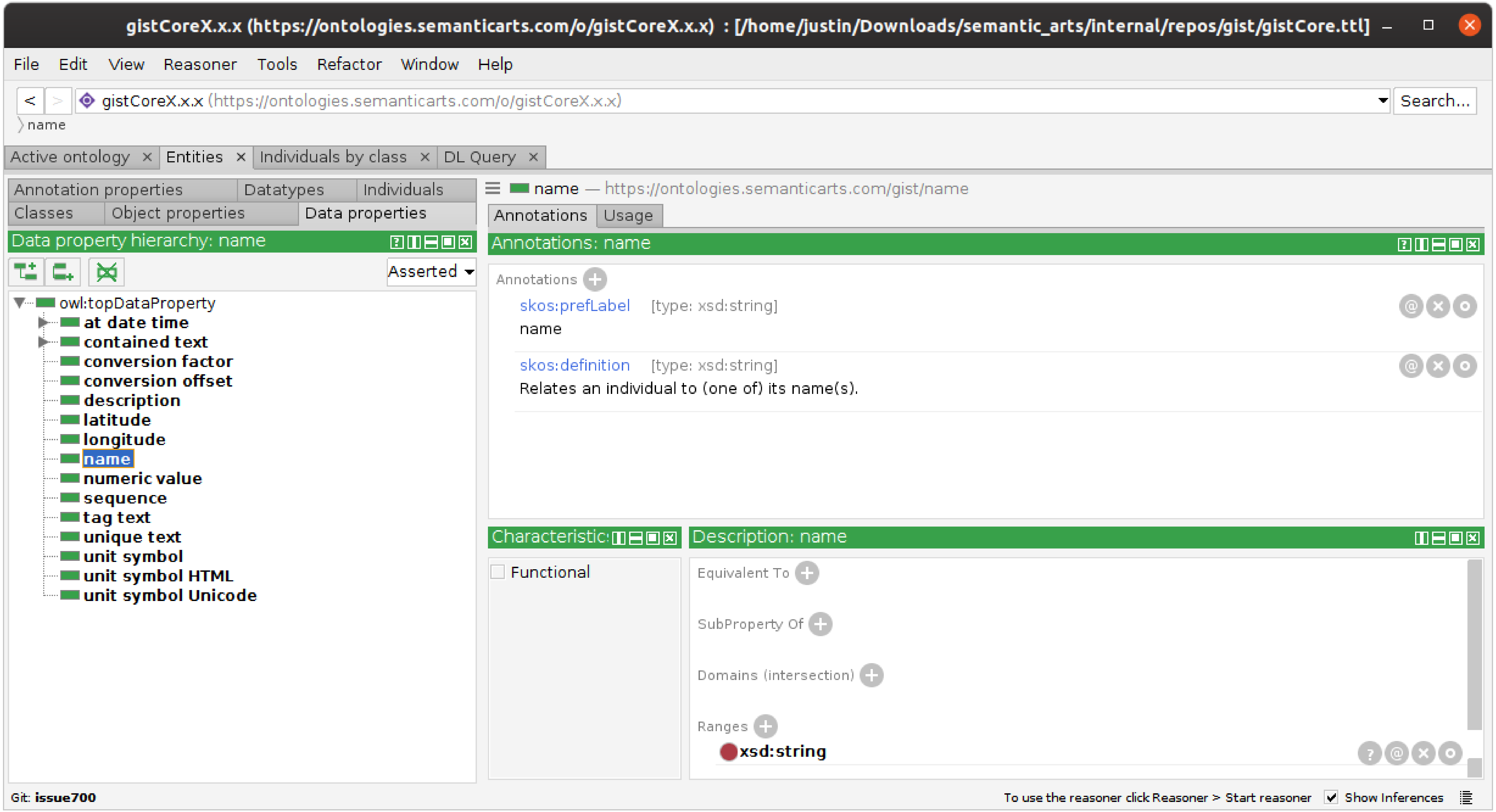
skos:definition [type: xsd:string]
The subject controls or inhibits the object in some way

□ Functional
□ Inverse functional
□ Transitive
□ Symmetric
□ Asymmetric
□ Reflexive
□ Irreflexive

Inverse Of
'is governed by'

Domains (intersection)
Intention or Organization or Person or Template

Ranges (intersection)
Category or Content or 'Geo Region' or 'Intellectual Property' or Intention or Organization or Person or 'Physical Identifiable Item' or 'Physical Substance'



gistCoreX.x.x (https://ontologies.semanticarts.com/o/gistCoreX.x.x) : [/home/justin/Downloads/semantic_arts/internal/repos/gist/gistCore.ttl]

gistCoreX.x.x (https://ontologies.semanticarts.com/o/gistCoreX.x.x)

〉name

Data property hierarchy: name

```
▼ owl:topDataProperty
   ▶ at date time
   ▶ contained text
     conversion factor
     conversion offset
     description
     latitude
     longitude
     name
     numeric value
     sequence
     tag text
     unique text
     unit symbol
     unit symbol HTML
     unit symbol Unicode
```

name — https://ontologies.semanticarts.com/gist/name

Annotations: name

Annotations

skos:prefLabel [type: xsd:string]
name

skos:definition [type: xsd:string]
Relates an individual to (one of) its name(s).

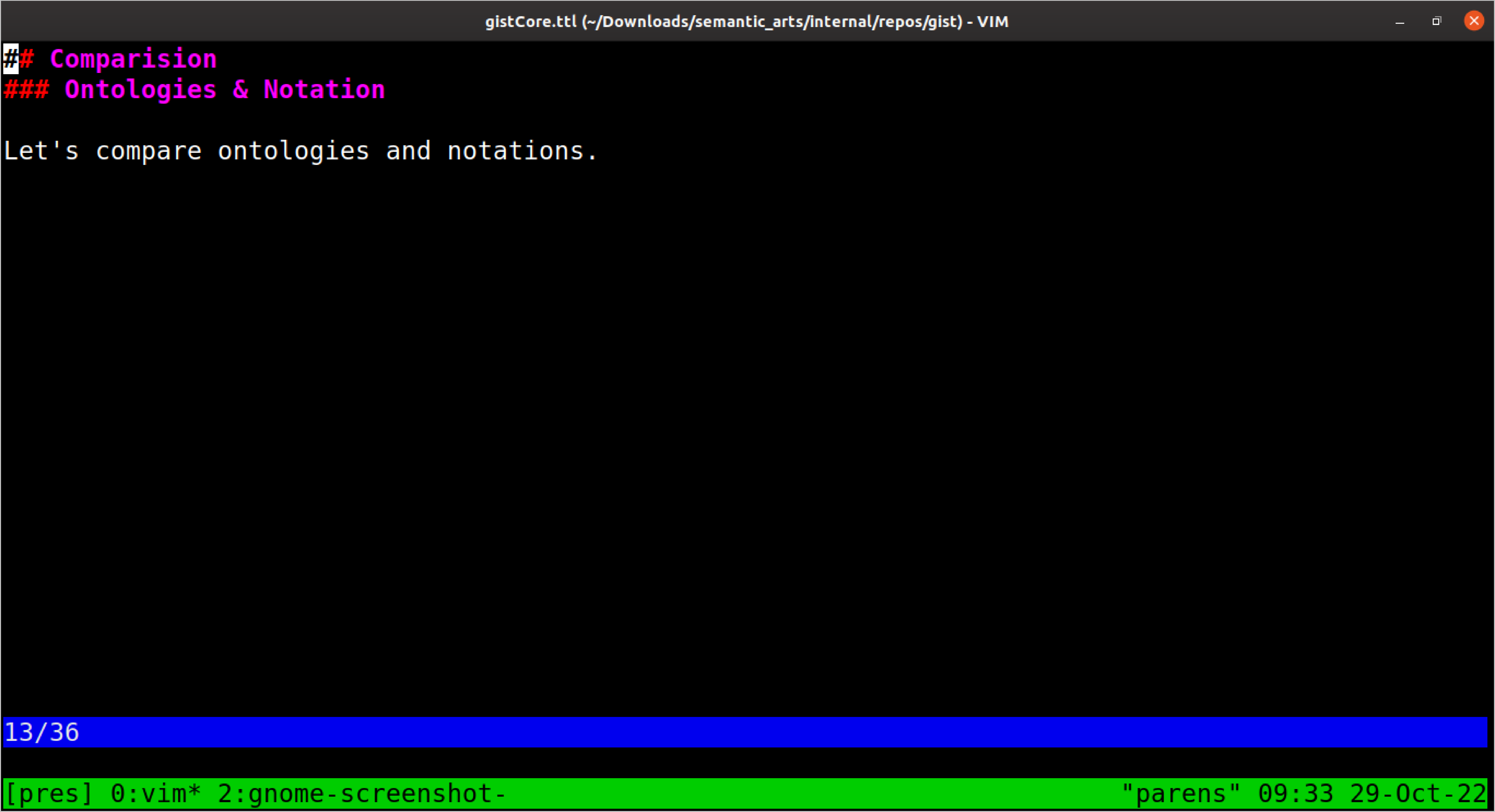
□ Functional

Description: name

Ranges
xsd:string

Git: **issue700**

To use the reasoner click Reasoner > Start reasoner ☑ Show Inferences

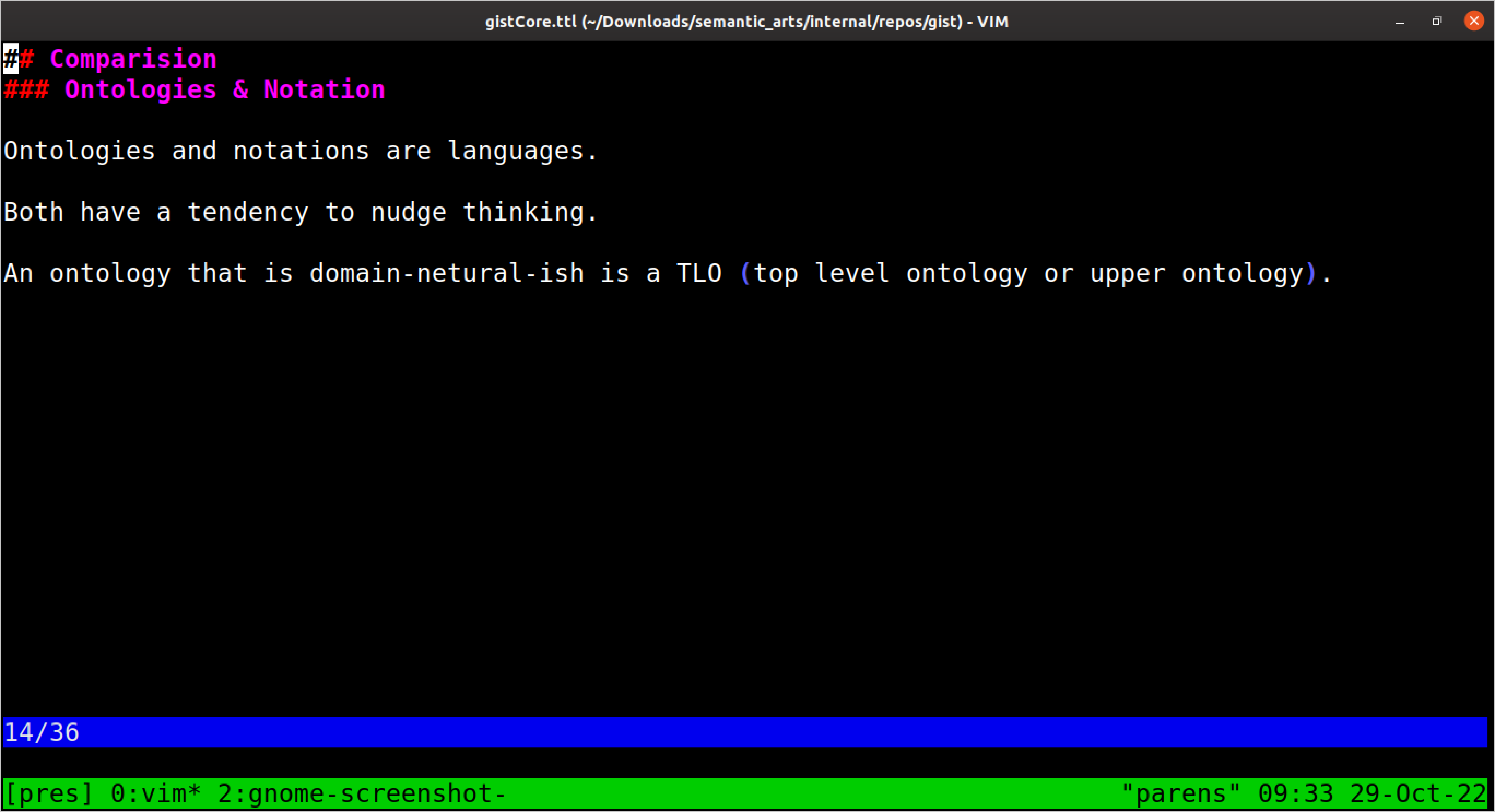


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

```
## Comparision
### Ontologies & Notation

Let's compare ontologies and notations.
```

```
13/36
```

```
[pres] 0:vim* 2:gnome-screenshot-                          "parens" 09:33 29-Oct-22
```


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

Comparision

Ontologies & Notation

Ontologies and notations are languages.

Both have a tendency to nudge thinking.

An ontology that is domain-netural-ish is a TLO (top level ontology or upper ontology).

```
14/36
```

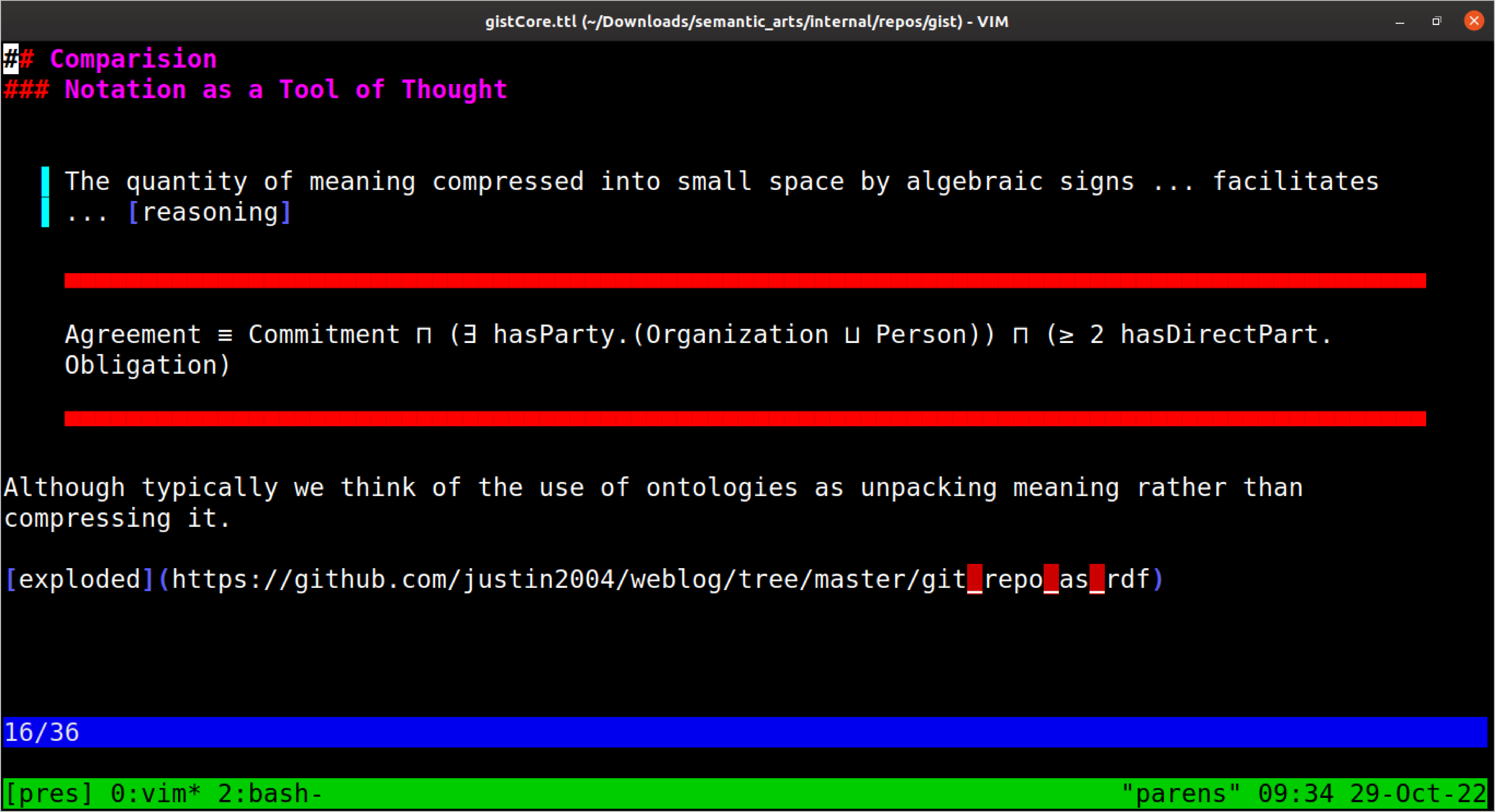
```
[pres] 0:vim* 2:gnome-screenshot-                              "parens" 09:33 29-Oct-22
```

gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

```
## Comparision
### Notation as a Tool of Thought
```

```
15/36
```

```
[pres] 0:vim* 2:gnome-screenshot-                                    "parens" 09:34 29-Oct-22
```


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

Comparision
Notation as a Tool of Thought

> The quantity of meaning compressed into small space by algebraic signs ... facilitates ... [reasoning]

```
Agreement ≡ Commitment ⊓ (∃ hasParty.(Organization ⊔ Person)) ⊓ (≥ 2 hasDirectPart.
Obligation)
```

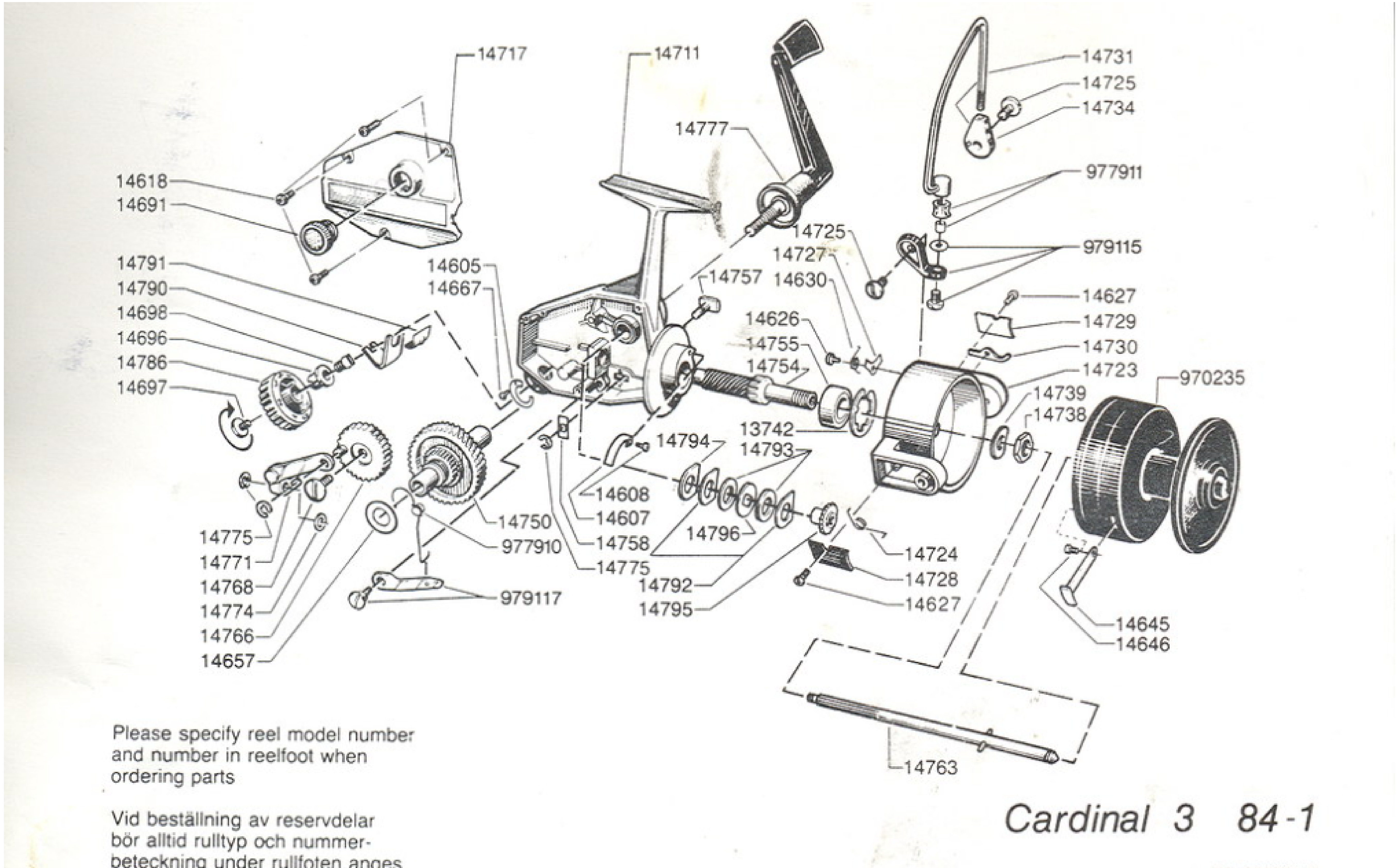
Although typically we think of the use of ontologies as unpacking meaning rather than compressing it.

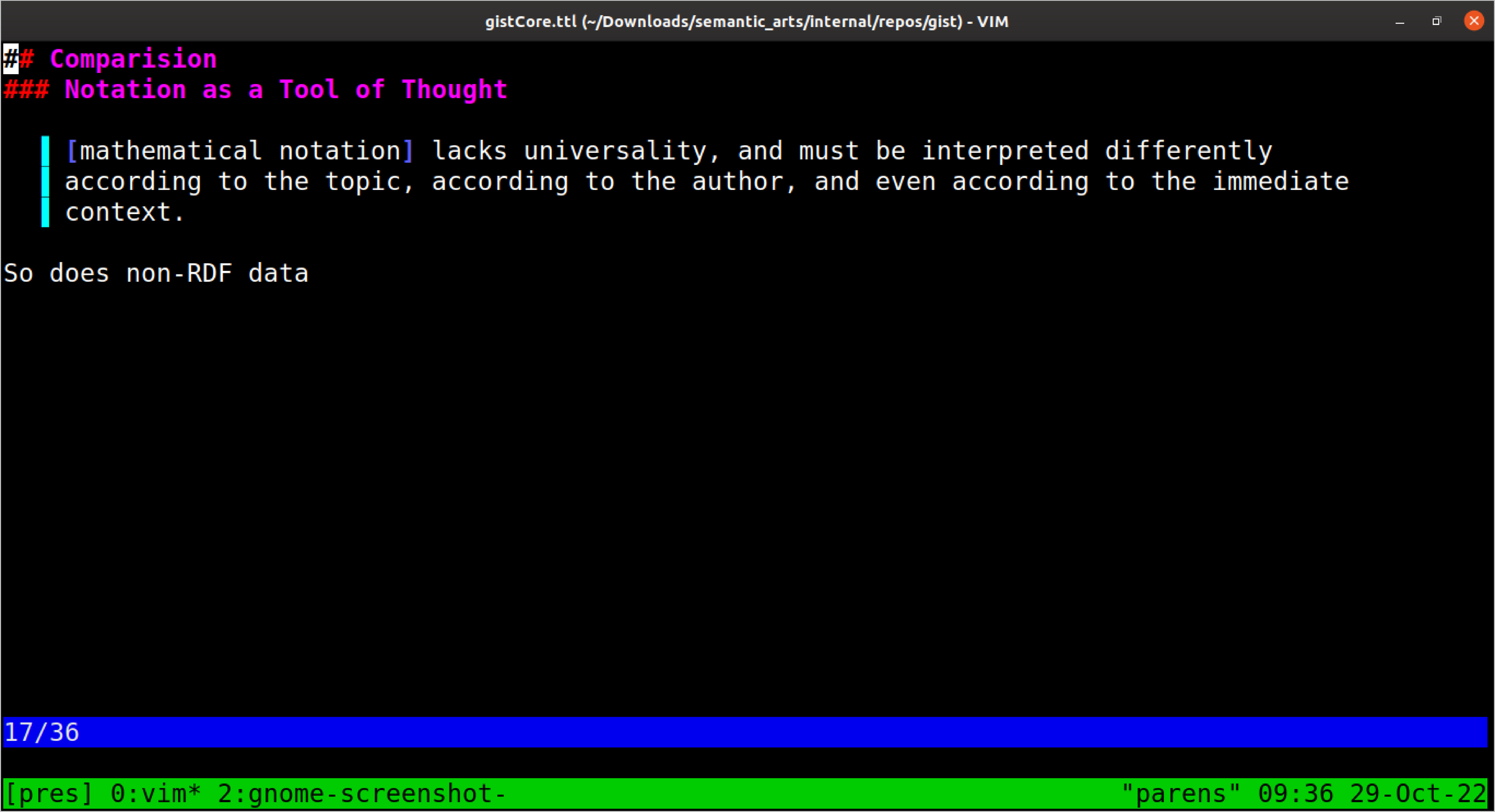
[exploded](https://github.com/justin2004/weblog/tree/master/git_repo_as_rdf)

```
16/36
[pres] 0:vim* 2:bash-                                              "parens" 09:34 29-Oct-22
```

You are encouraged (if you use a thoughtful upper ontology such as Gist) to unpack meaning.

I think of the semantic web as something like the exploded part diagram for the web's data.



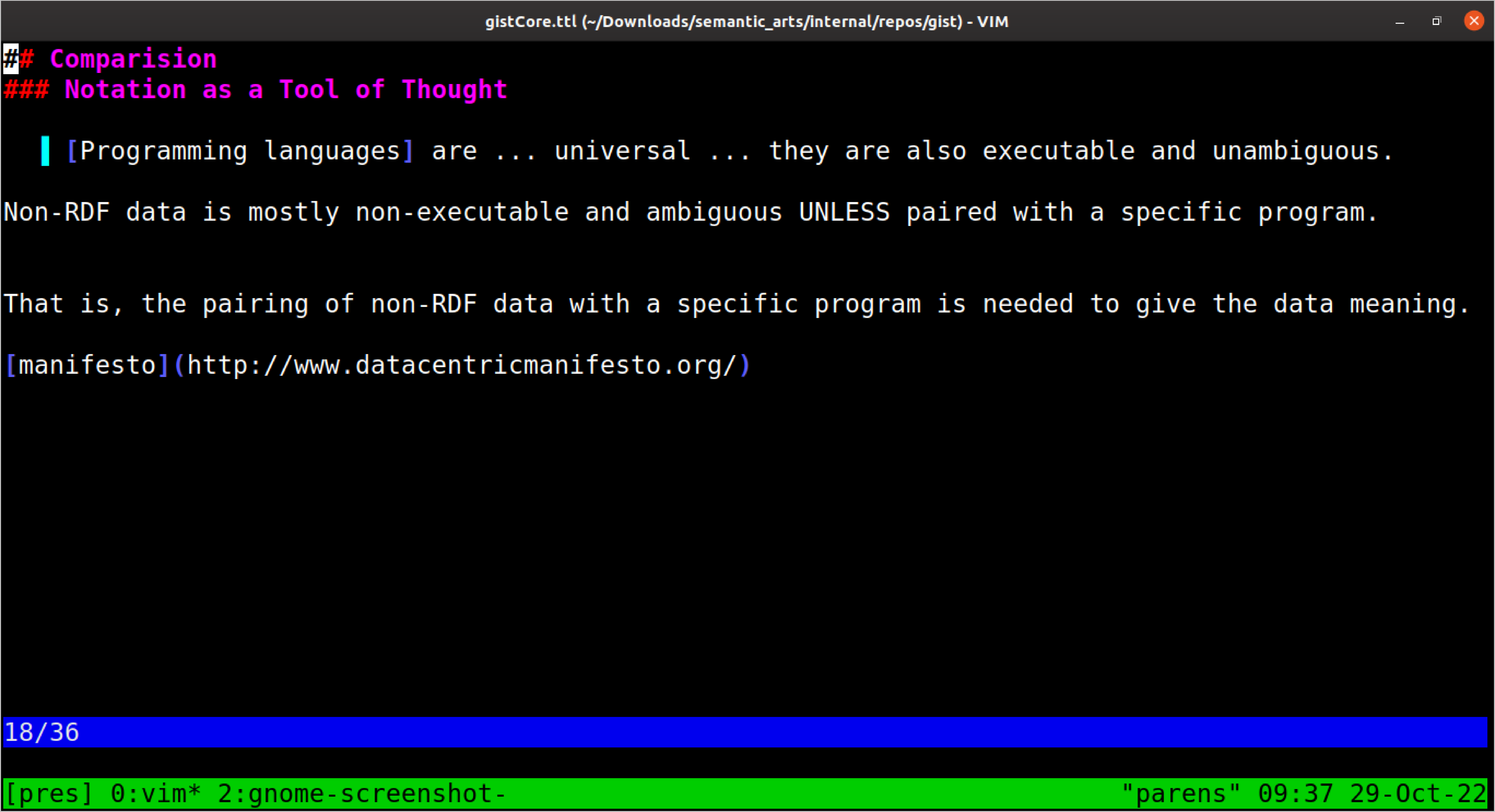

gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

Comparision
Notation as a Tool of Thought

> [mathematical notation] lacks universality, and must be interpreted differently according to the topic, according to the author, and even according to the immediate context.

So does non-RDF data

17/36
[pres] 0:vim* 2:gnome-screenshot-
"parens" 09:36 29-Oct-22

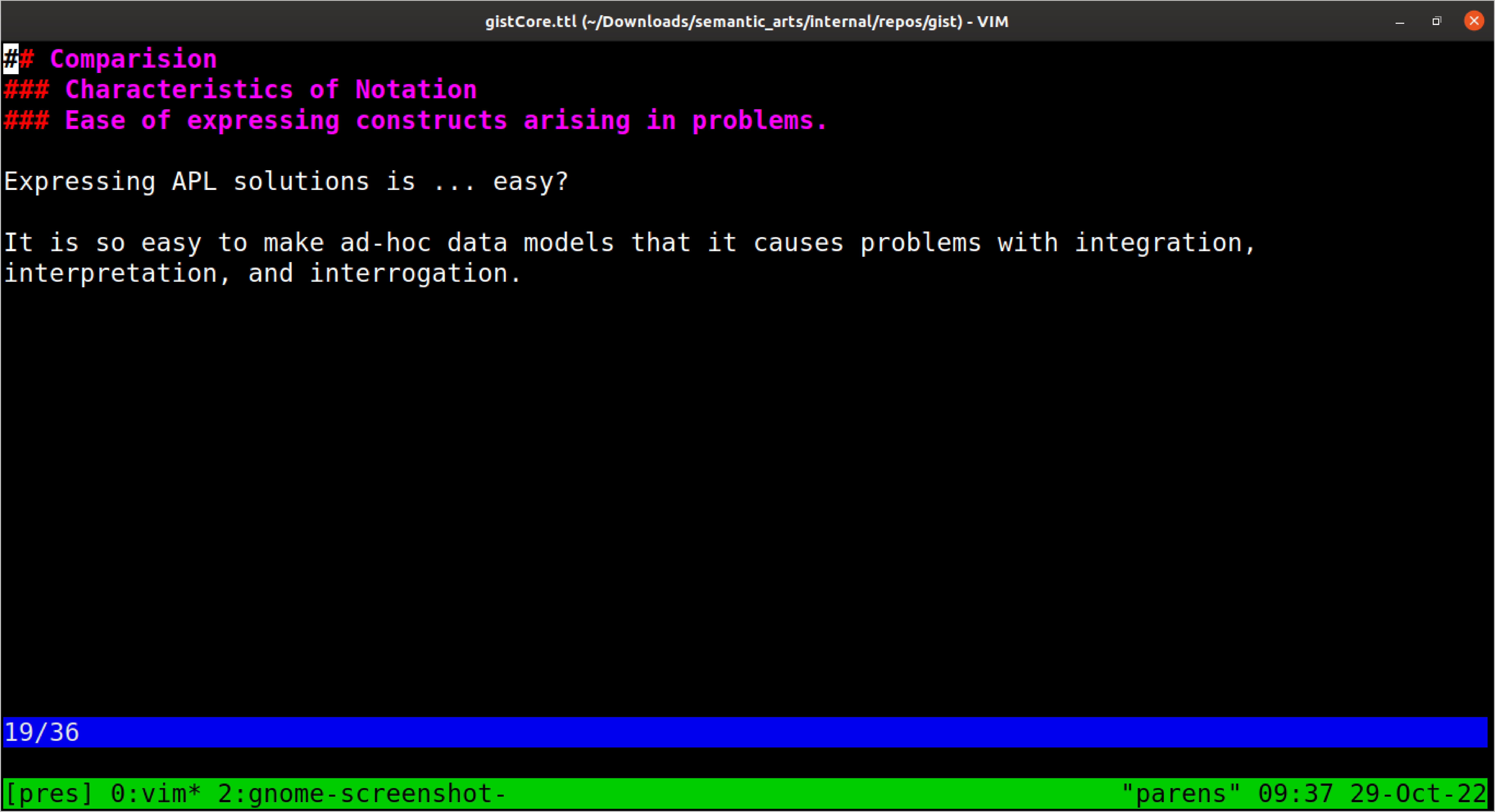


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

```
## Comparision
### Notation as a Tool of Thought

  ▌ [Programming languages] are ... universal ... they are also executable and unambiguous.

Non-RDF data is mostly non-executable and ambiguous UNLESS paired with a specific program.


That is, the pairing of non-RDF data with a specific program is needed to give the data meaning.

[manifesto](http://www.datacentricmanifesto.org/)
```

```
18/36
```

```
[pres] 0:vim* 2:gnome-screenshot-                                 "parens" 09:37 29-Oct-22
```



gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

Comparision

Characteristics of Notation

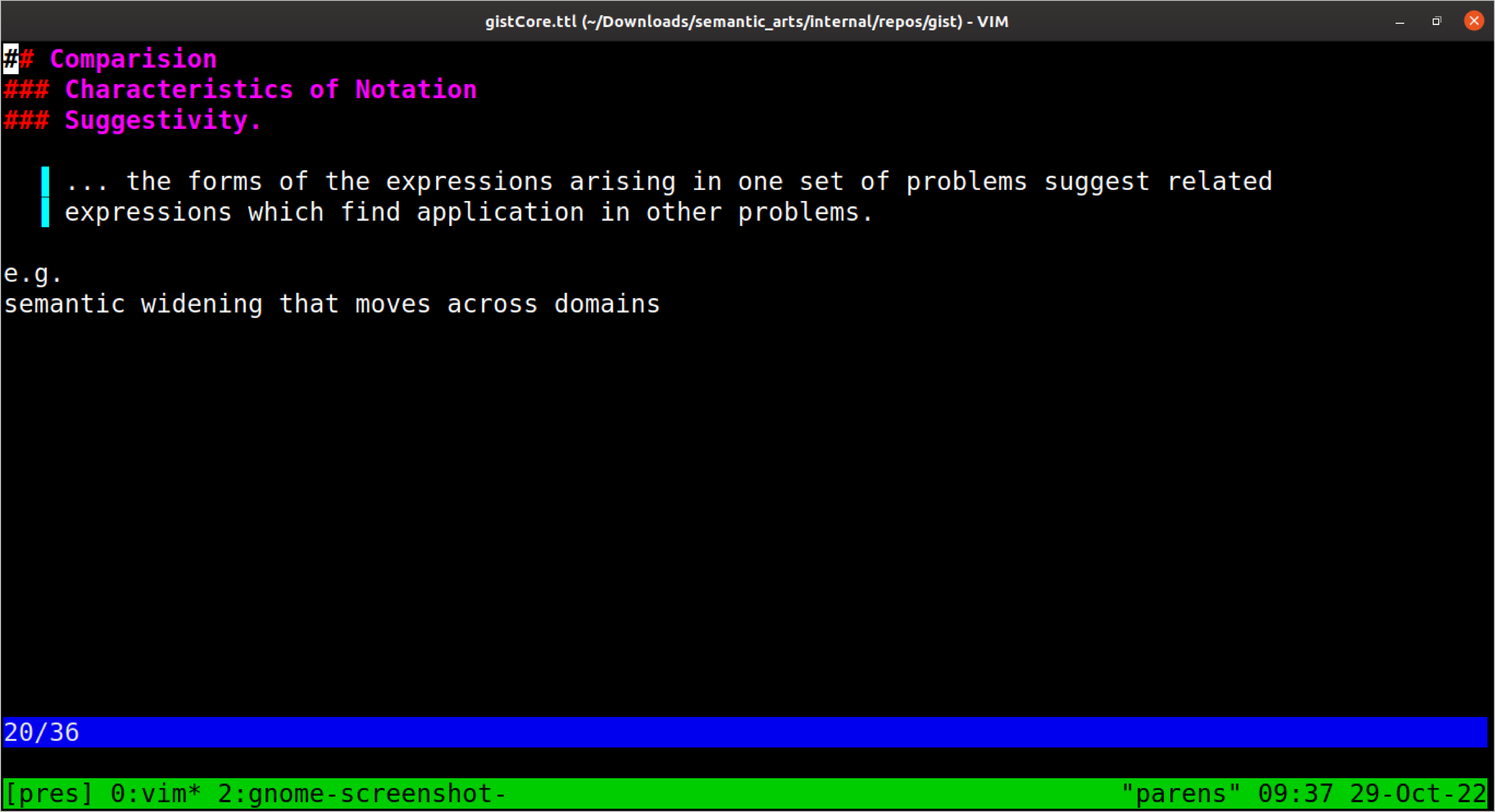
Ease of expressing constructs arising in problems.

Expressing APL solutions is ... easy?

It is so easy to make ad-hoc data models that it causes problems with integration, interpretation, and interrogation.

```
19/36
```

```
[pres] 0:vim* 2:gnome-screenshot-                                    "parens" 09:37 29-Oct-22
```



gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

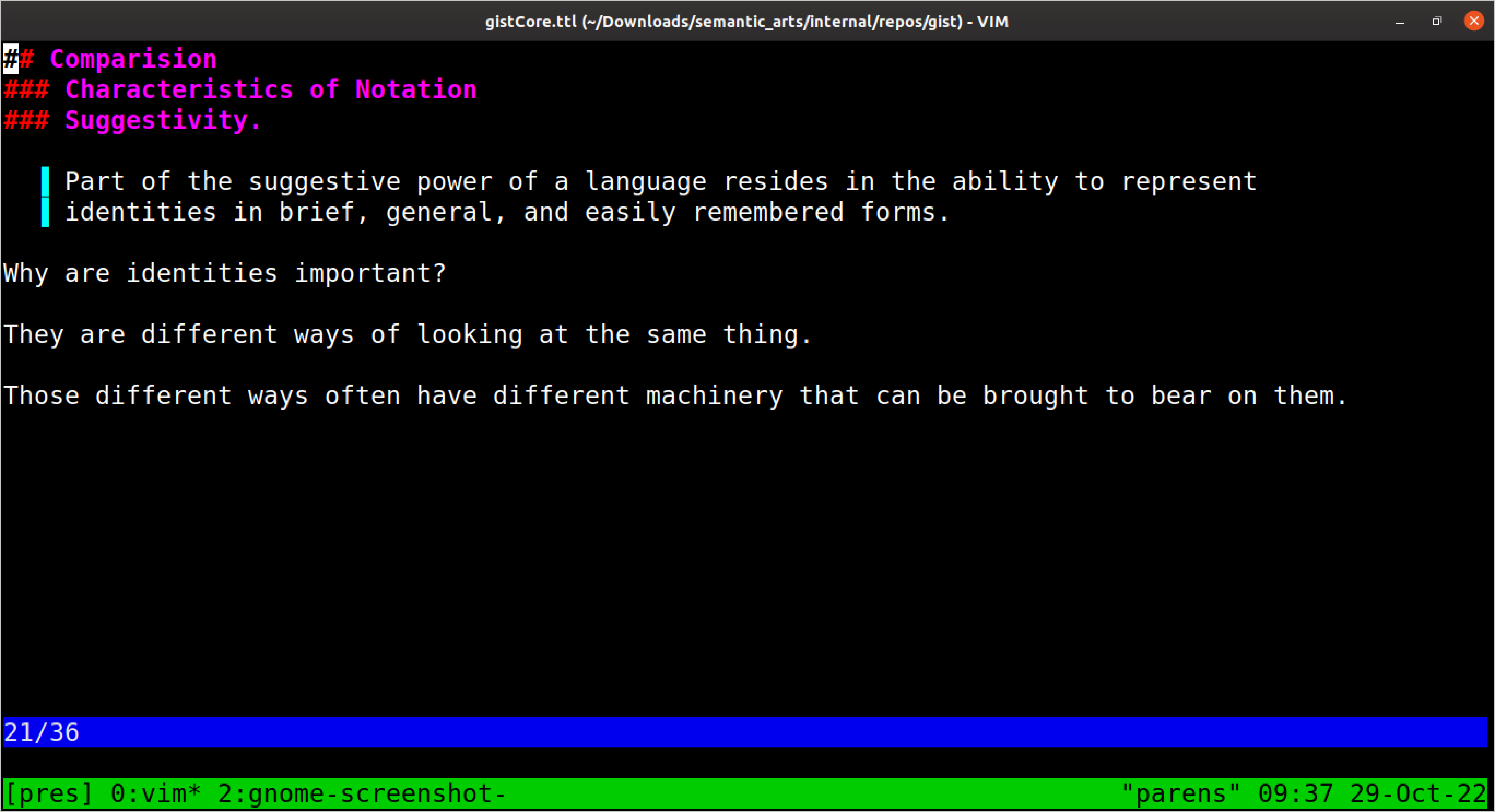
Comparision
Characteristics of Notation
Suggestivity.

> ... the forms of the expressions arising in one set of problems suggest related expressions which find application in other problems.

e.g.
semantic widening that moves across domains

```
20/36
```

```
[pres] 0:vim* 2:gnome-screenshot-                                   "parens" 09:37 29-Oct-22
```


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

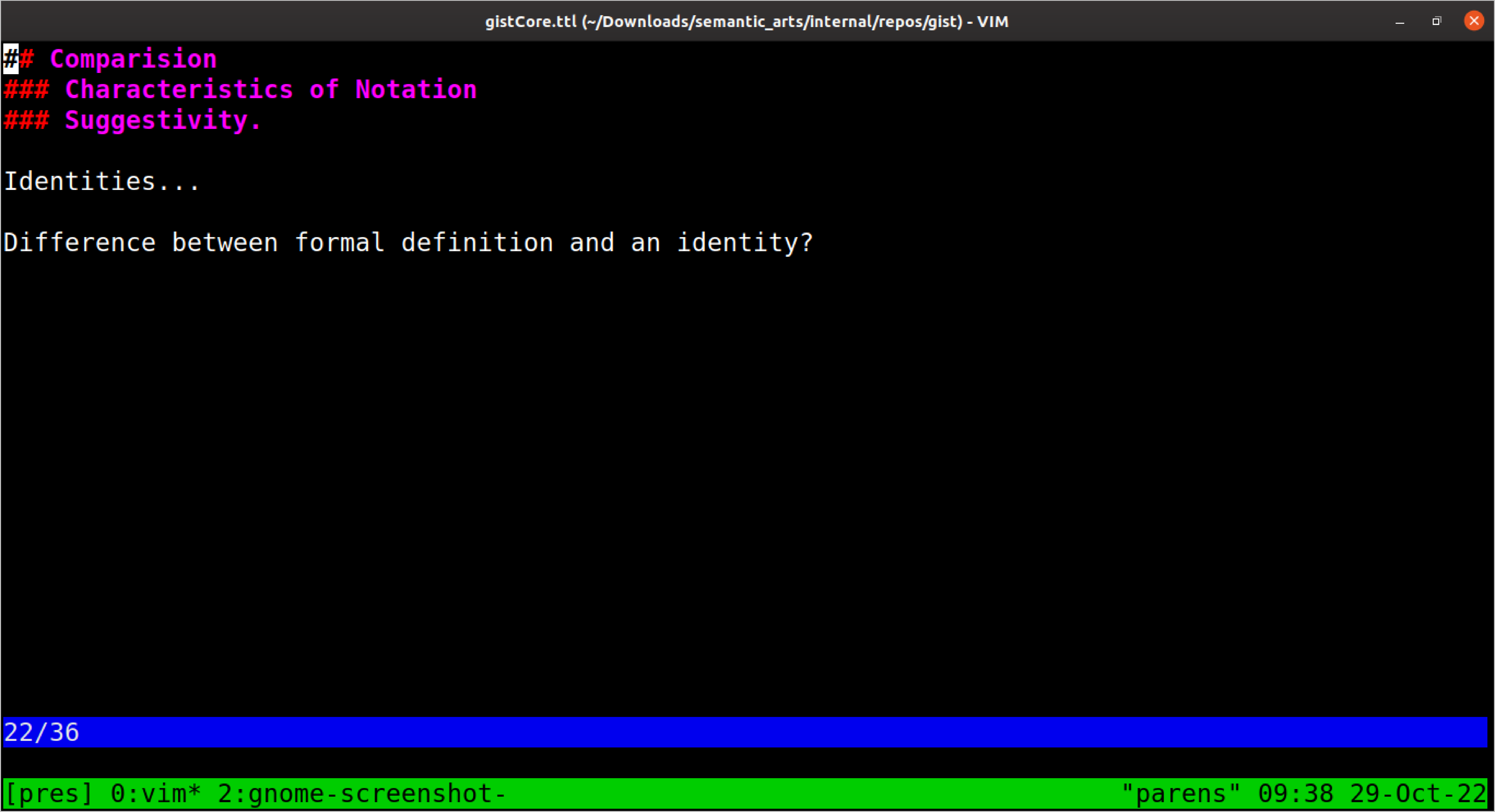
Comparision
Characteristics of Notation
Suggestivity.

> Part of the suggestive power of a language resides in the ability to represent identities in brief, general, and easily remembered forms.

Why are identities important?

They are different ways of looking at the same thing.

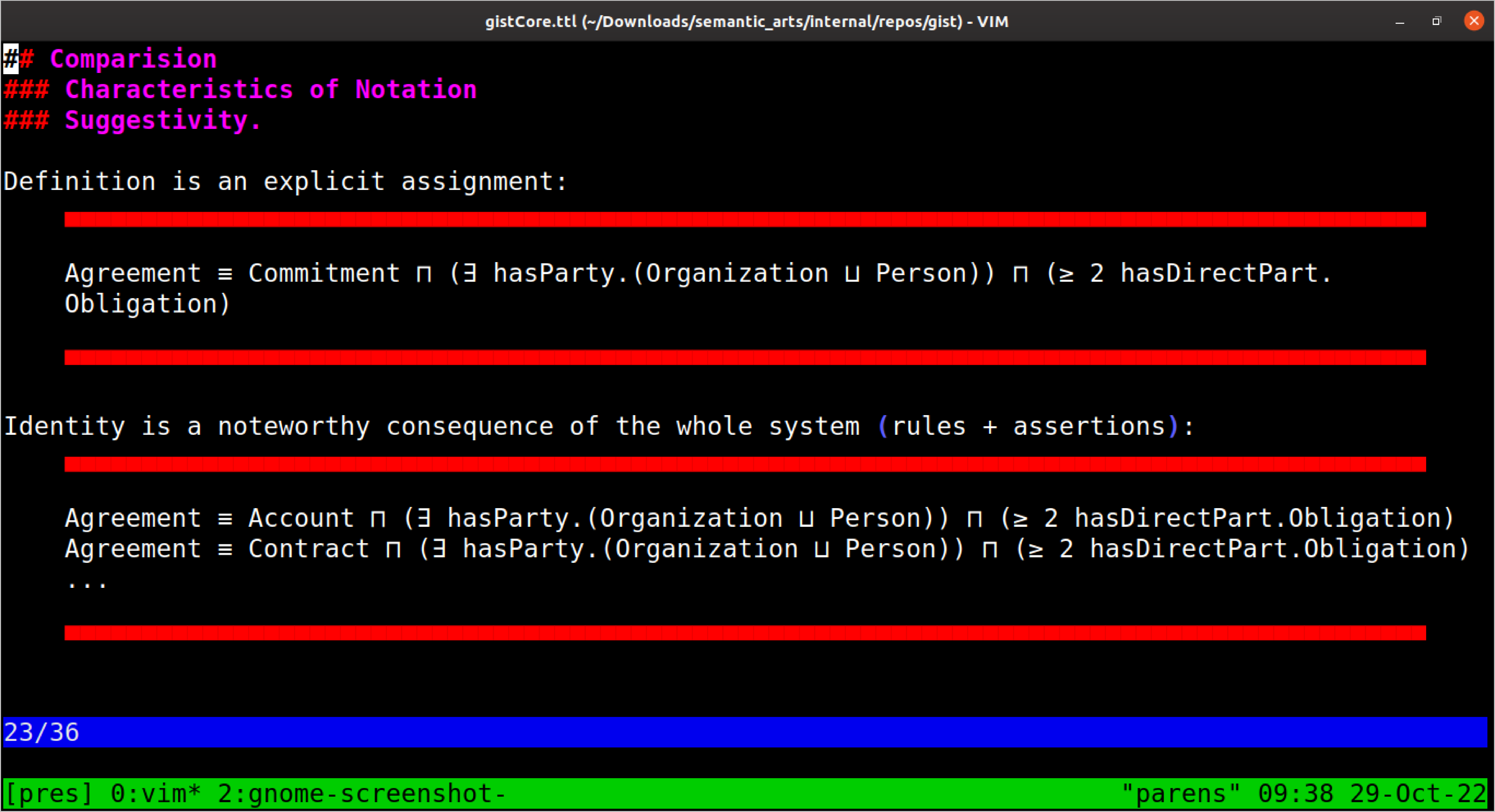
Those different ways often have different machinery that can be brought to bear on them.


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

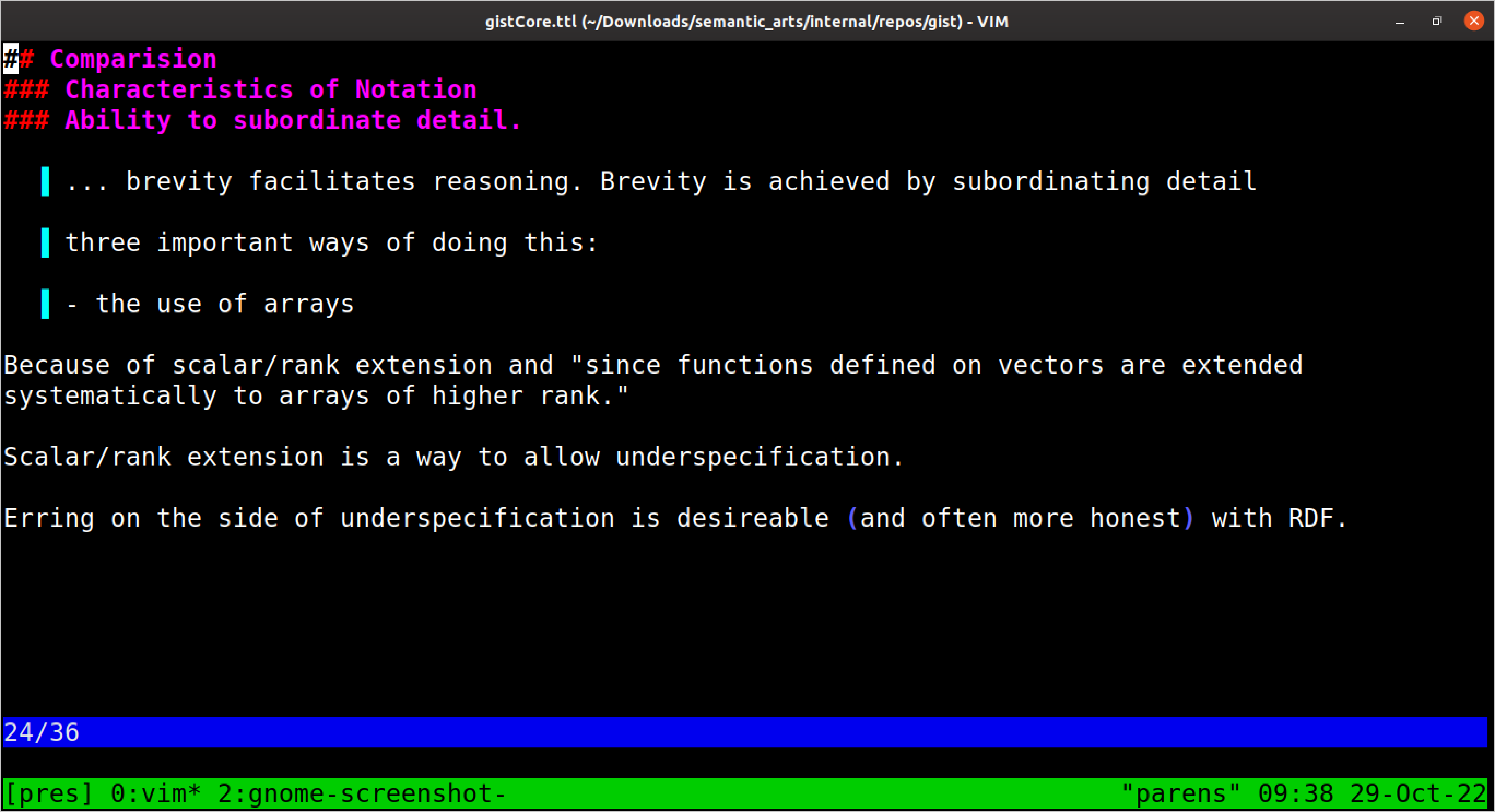
```
## Comparision
### Characteristics of Notation
### Suggestivity.

Identities...

Difference between formal definition and an identity?
```

```
22/36
```

```
[pres] 0:vim* 2:gnome-screenshot-                    "parens" 09:38 29-Oct-22
```


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

```
## Comparision
### Characteristics of Notation
### Suggestivity.

Definition is an explicit assignment:


    Agreement ≡ Commitment ⊓ (∃ hasParty.(Organization ⊔ Person)) ⊓ (≥ 2 hasDirectPart.
    Obligation)


Identity is a noteworthy consequence of the whole system (rules + assertions):


    Agreement ≡ Account ⊓ (∃ hasParty.(Organization ⊔ Person)) ⊓ (≥ 2 hasDirectPart.Obligation)
    Agreement ≡ Contract ⊓ (∃ hasParty.(Organization ⊔ Person)) ⊓ (≥ 2 hasDirectPart.Obligation)
    ...


23/36

[pres] 0:vim* 2:gnome-screenshot-                                    "parens" 09:38 29-Oct-22
```


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

Comparision

Characteristics of Notation

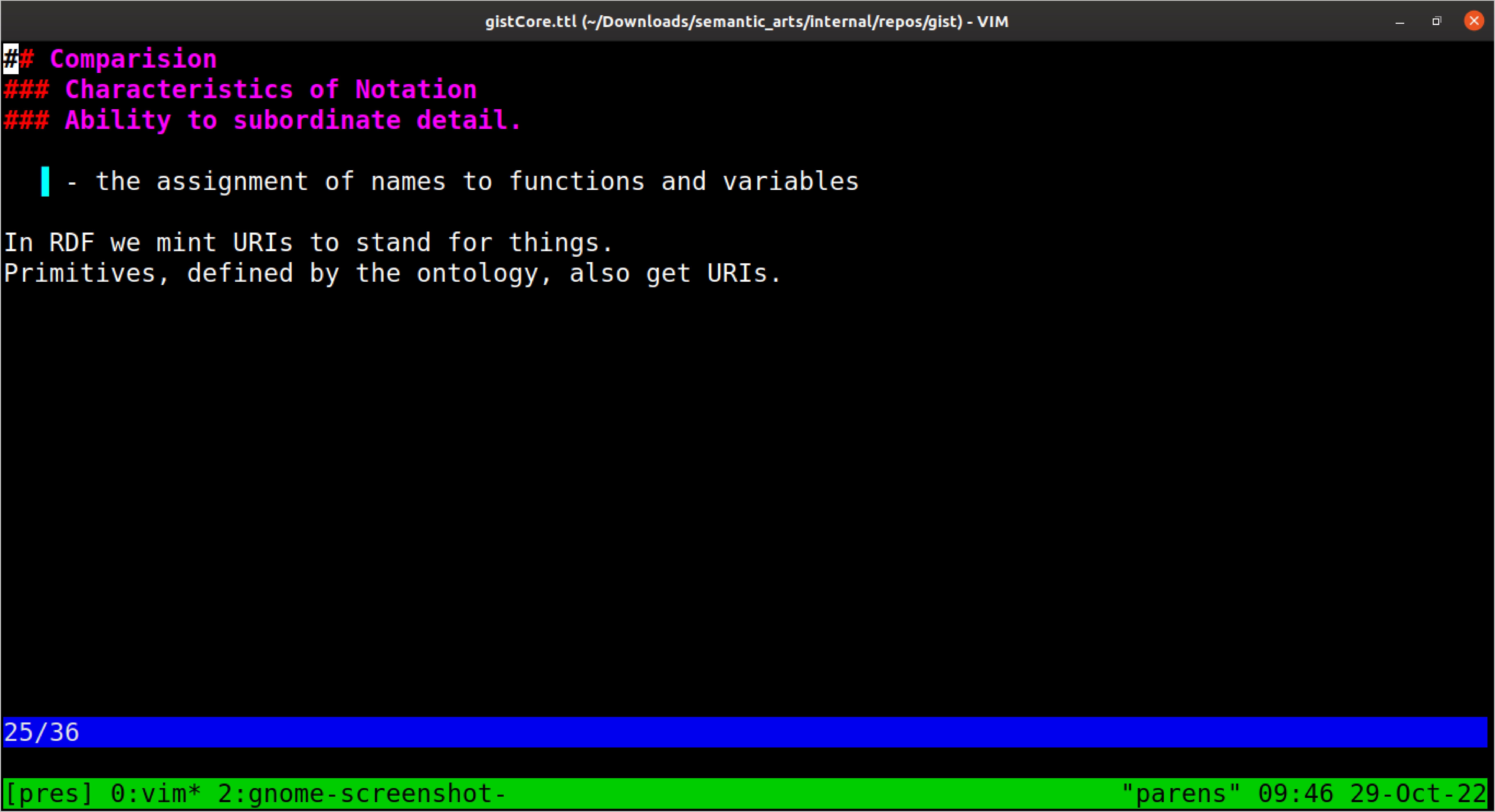
Ability to subordinate detail.

> ... brevity facilitates reasoning. Brevity is achieved by subordinating detail
>
> three important ways of doing this:
>
> - the use of arrays

Because of scalar/rank extension and "since functions defined on vectors are extended systematically to arrays of higher rank."

Scalar/rank extension is a way to allow underspecification.

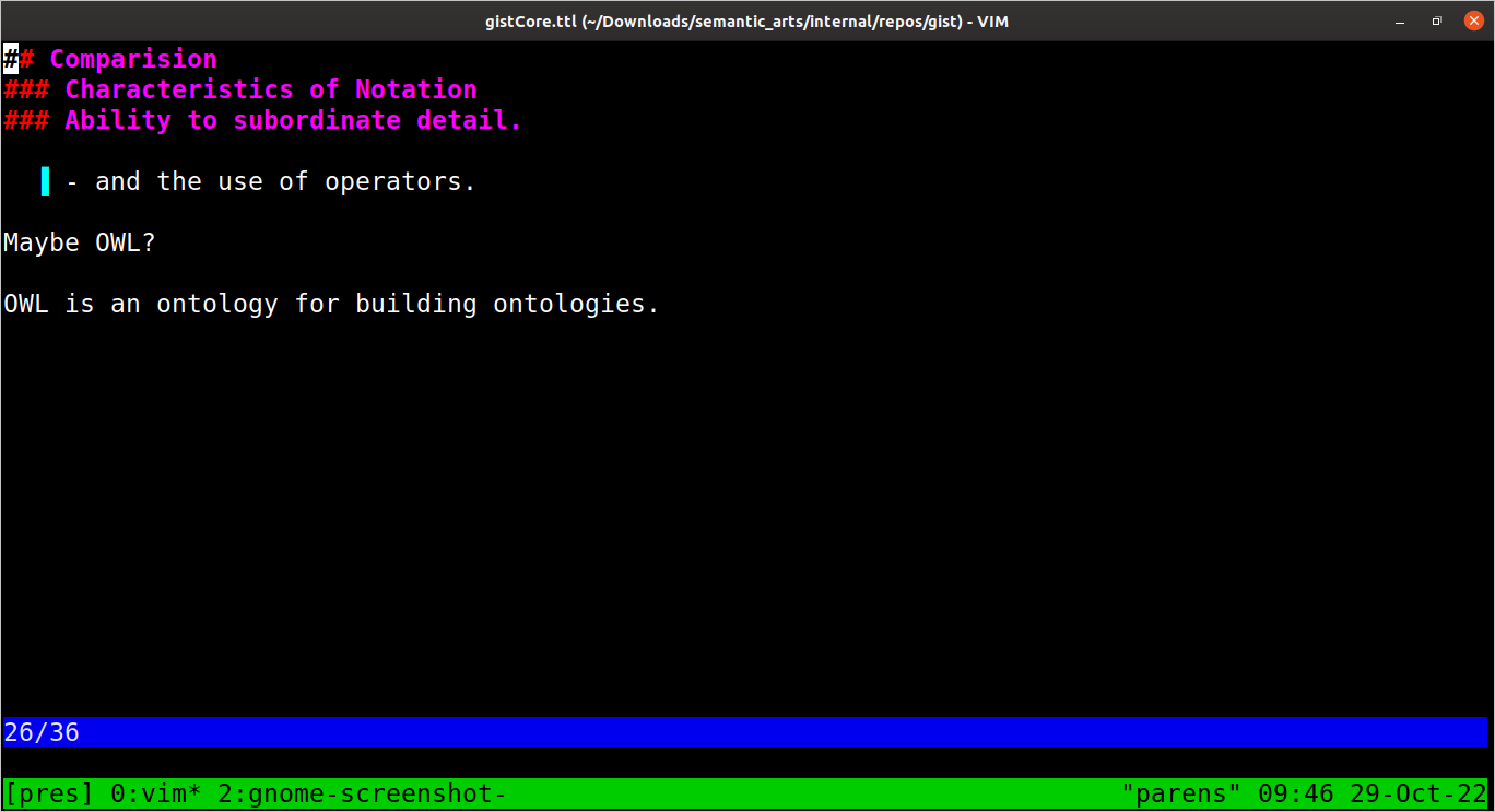
Erring on the side of underspecification is desireable (and often more honest) with RDF.


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

```
## Comparision
### Characteristics of Notation
### Ability to subordinate detail.

  ▌ - the assignment of names to functions and variables

In RDF we mint URIs to stand for things.
Primitives, defined by the ontology, also get URIs.
```

```
25/36
```

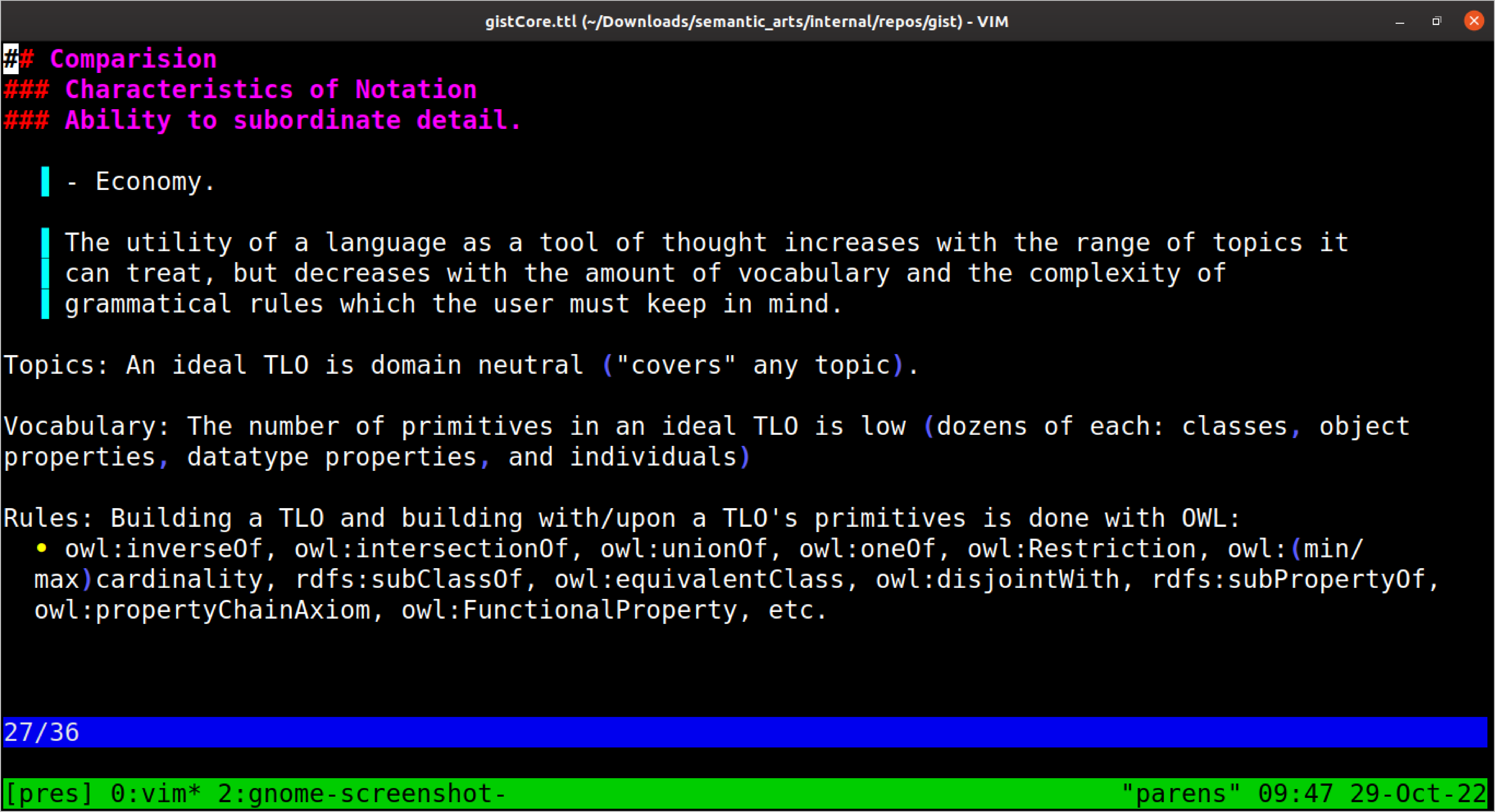
```
[pres] 0:vim* 2:gnome-screenshot-                                   "parens" 09:46 29-Oct-22
```



gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

```
## Comparision
### Characteristics of Notation
### Ability to subordinate detail.

   ▌ - and the use of operators.

Maybe OWL?

OWL is an ontology for building ontologies.
```

```
26/36
[pres] 0:vim* 2:gnome-screenshot-                                    "parens" 09:46 29-Oct-22
```


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

Comparision

Characteristics of Notation

Ability to subordinate detail.

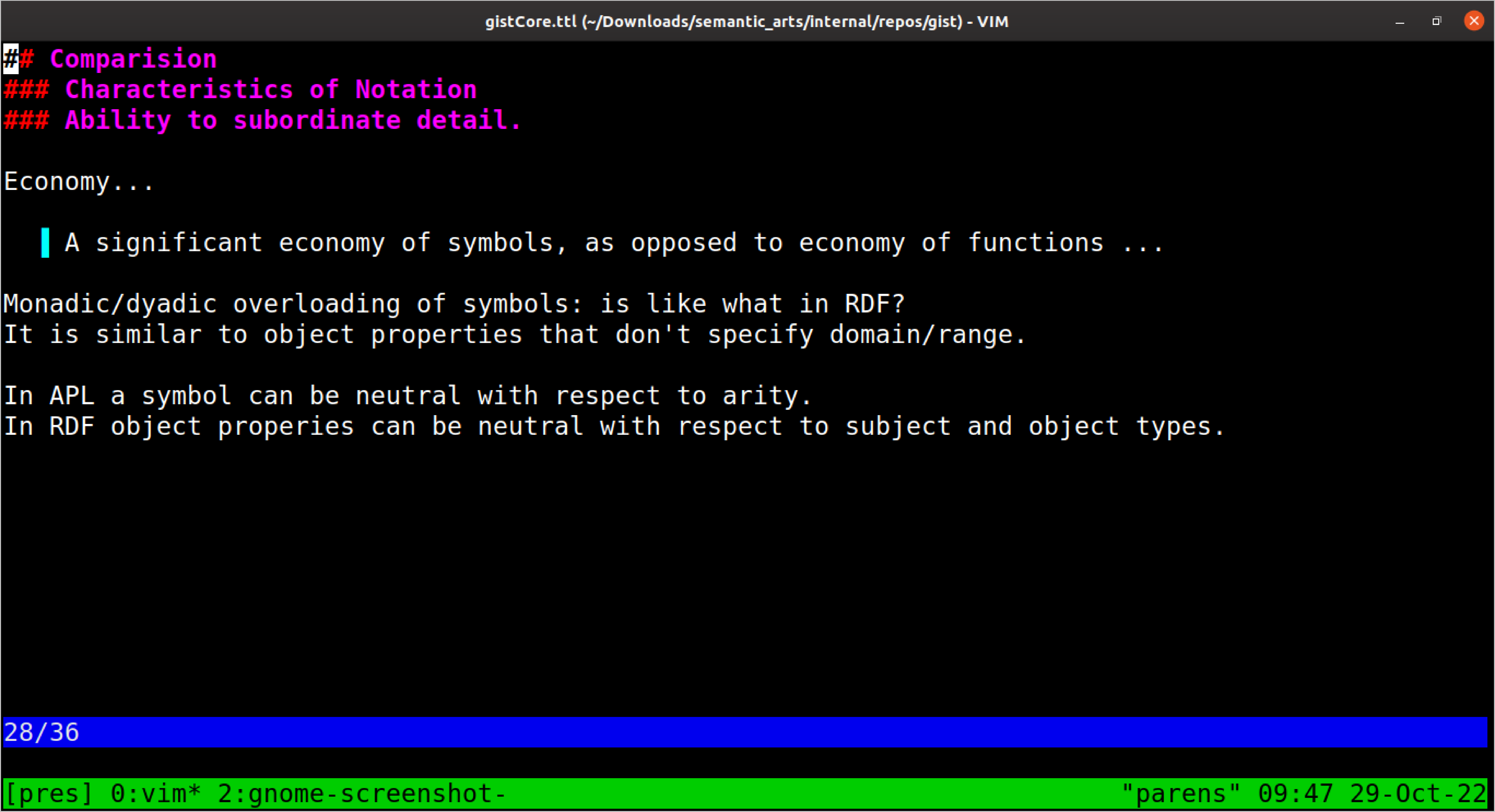
> - Economy.
>
> The utility of a language as a tool of thought increases with the range of topics it can treat, but decreases with the amount of vocabulary and the complexity of grammatical rules which the user must keep in mind.

Topics: An ideal TLO is domain neutral ("covers" any topic).

Vocabulary: The number of primitives in an ideal TLO is low (dozens of each: classes, object properties, datatype properties, and individuals)

Rules: Building a TLO and building with/upon a TLO's primitives is done with OWL:

- owl:inverseOf, owl:intersectionOf, owl:unionOf, owl:oneOf, owl:Restriction, owl:(min/max)cardinality, rdfs:subClassOf, owl:equivalentClass, owl:disjointWith, rdfs:subPropertyOf, owl:propertyChainAxiom, owl:FunctionalProperty, etc.


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

Comparision
Characteristics of Notation
Ability to subordinate detail.

Economy...

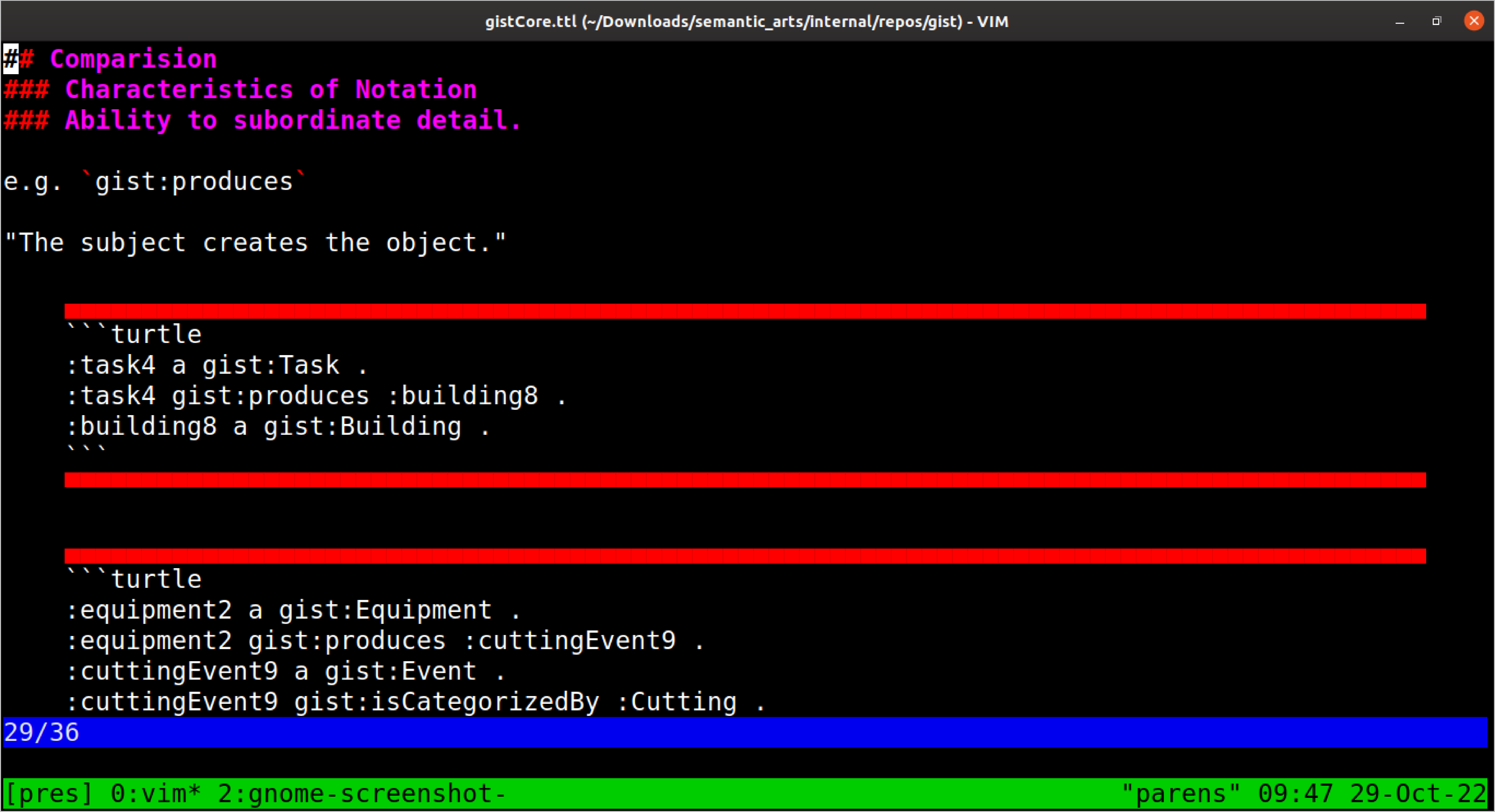
> A significant economy of symbols, as opposed to economy of functions ...

Monadic/dyadic overloading of symbols: is like what in RDF?
It is similar to object properties that don't specify domain/range.

In APL a symbol can be neutral with respect to arity.
In RDF object properies can be neutral with respect to subject and object types.

```
28/36
```

```
[pres] 0:vim* 2:gnome-screenshot-                          "parens" 09:47 29-Oct-22
```


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

Comparision
Characteristics of Notation
Ability to subordinate detail.

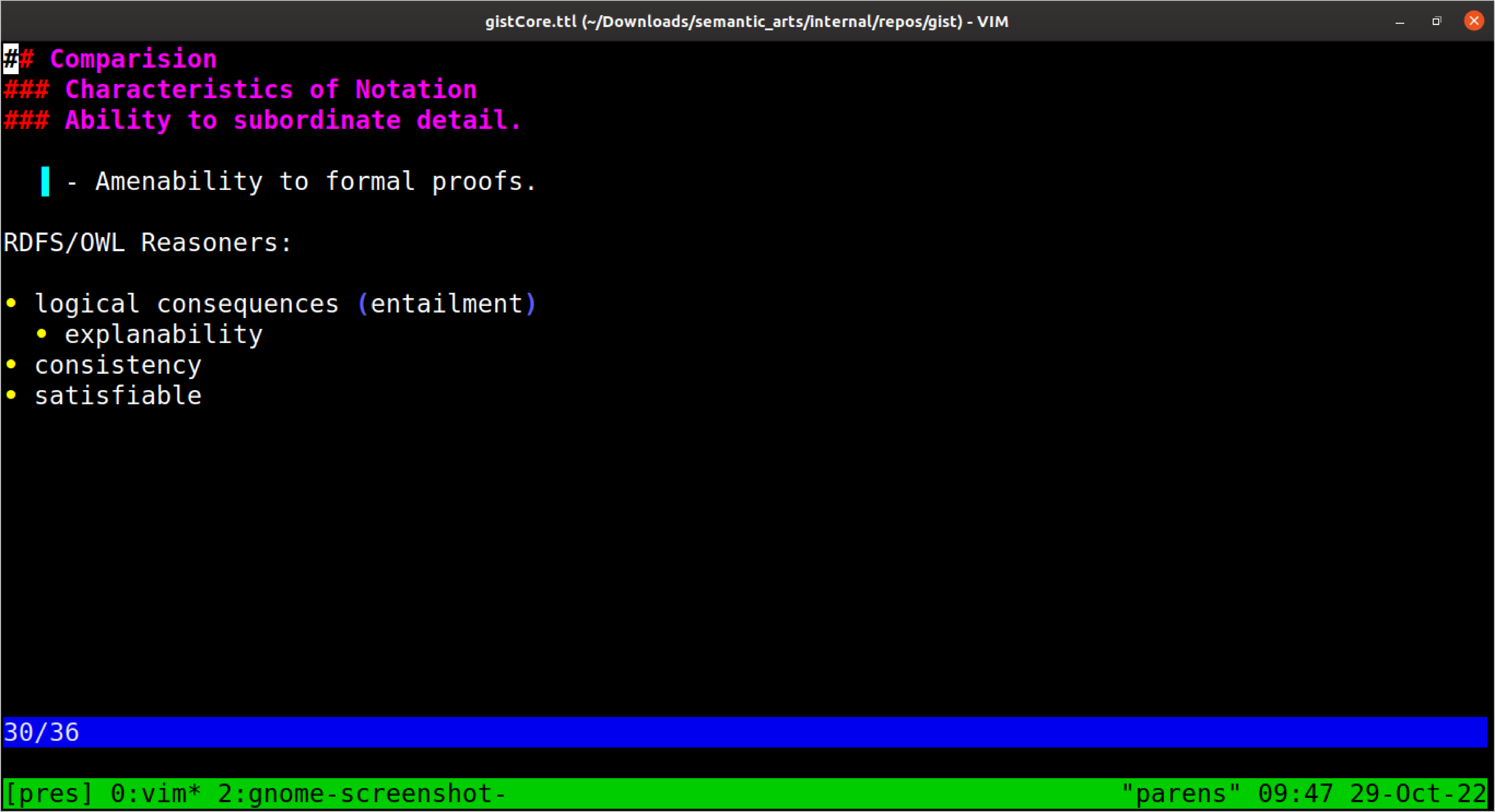
e.g. `gist:produces`

"The subject creates the object."

```turtle
:task4 a gist:Task .
:task4 gist:produces :building8 .
:building8 a gist:Building .
```

```turtle
:equipment2 a gist:Equipment .
:equipment2 gist:produces :cuttingEvent9 .
:cuttingEvent9 a gist:Event .
:cuttingEvent9 gist:isCategorizedBy :Cutting .
```

29/36
[pres] 0:vim* 2:gnome-screenshot-
"parens" 09:47 29-Oct-22


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

Comparision

Characteristics of Notation

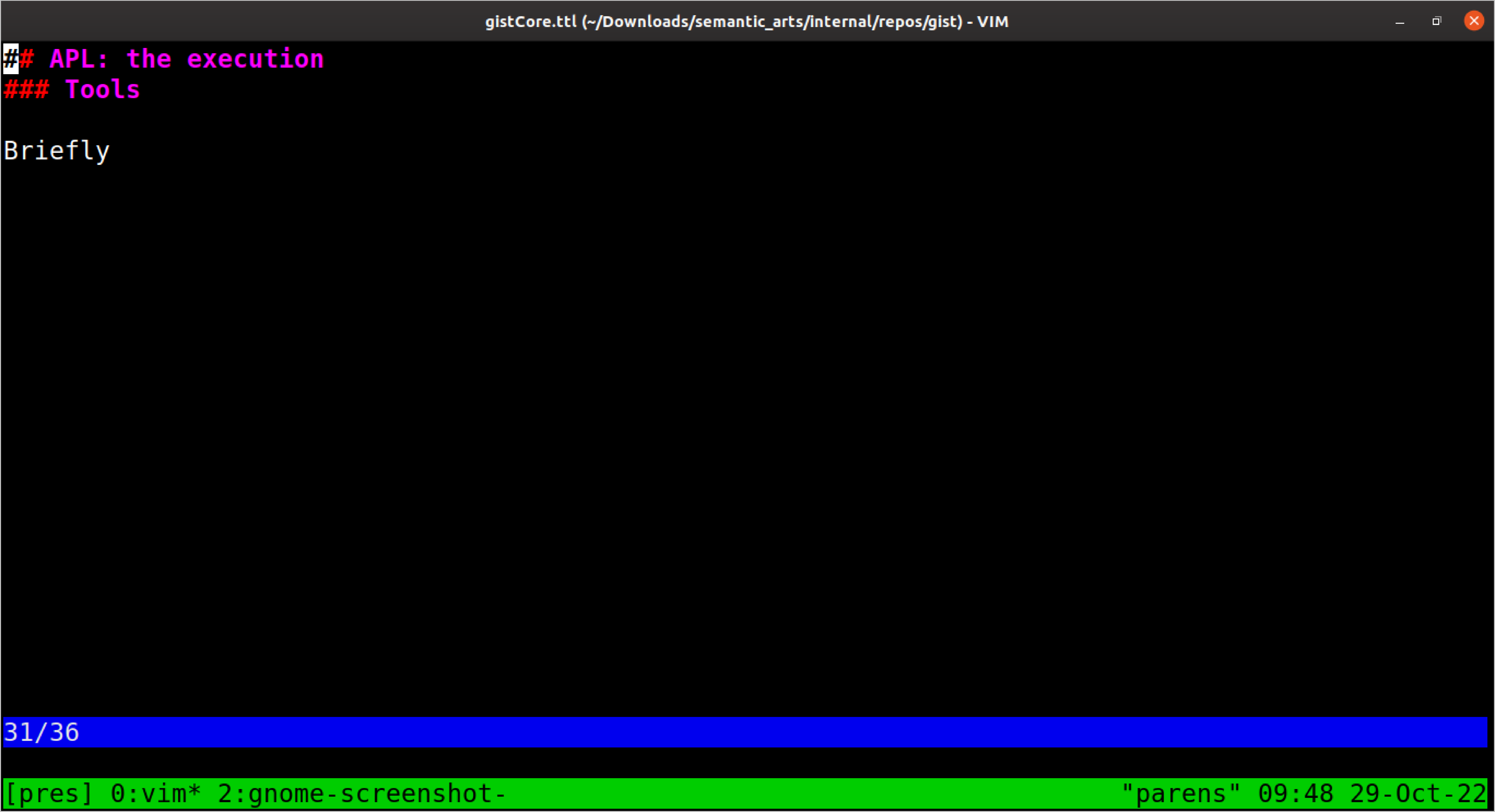
Ability to subordinate detail.

> - Amenability to formal proofs.

RDFS/OWL Reasoners:

- logical consequences (entailment)
  - explanability
- consistency
- satisfiable

30/36
[pres] 0:vim* 2:gnome-screenshot-
"parens" 09:47 29-Oct-22

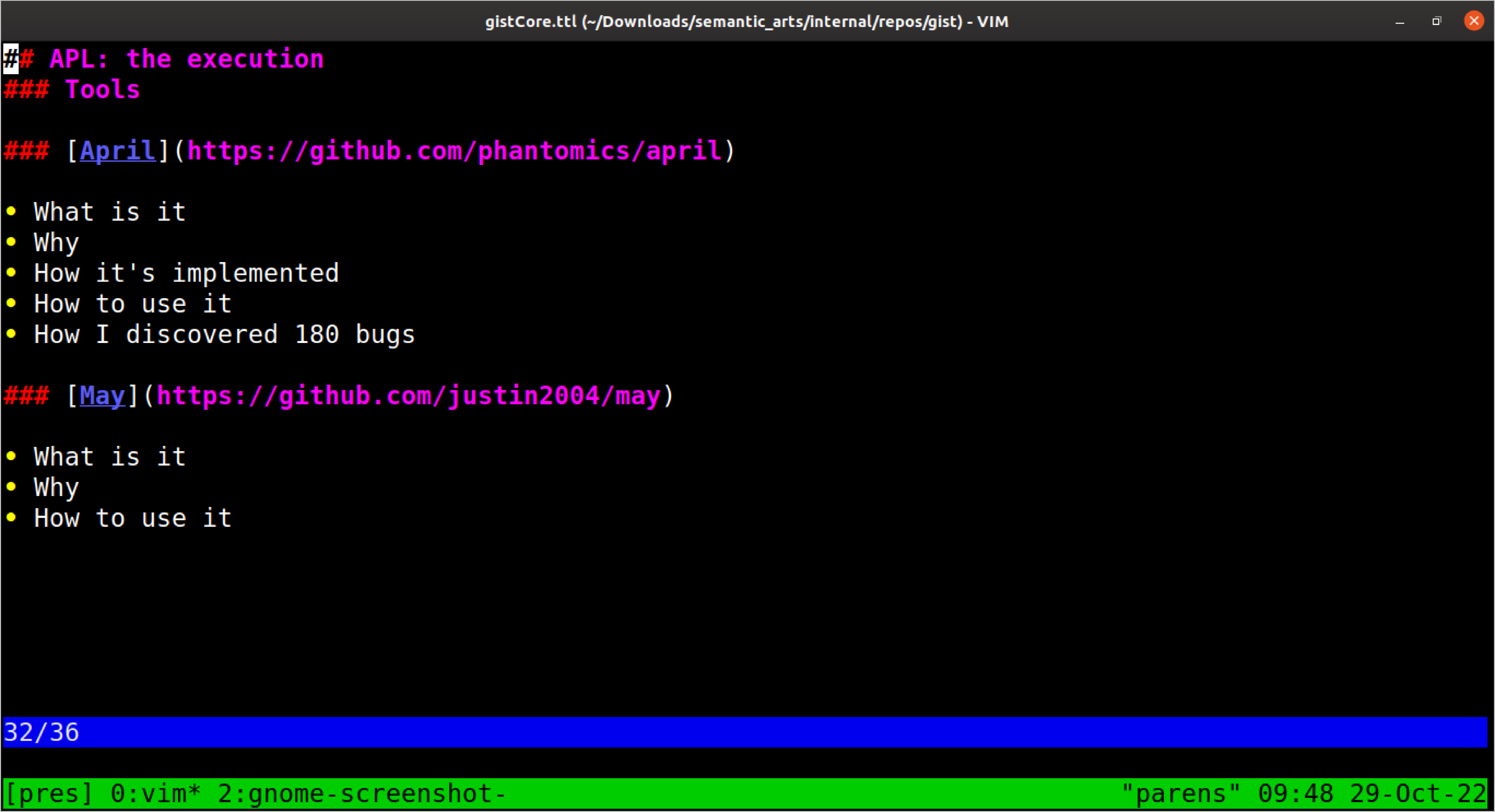


gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

```
## APL: the execution
### Tools

Briefly
```

```
31/36
```

```
[pres] 0:vim* 2:gnome-screenshot-                              "parens" 09:48 29-Oct-22
```



gistCore.ttl (~/Downloads/semantic_arts/internal/repos/gist) - VIM

```
## APL: the execution
### Tools

### [April](https://github.com/phantomics/april)

• What is it
• Why
• How it's implemented
• How to use it
• How I discovered 180 bugs

### [May](https://github.com/justin2004/may)

• What is it
• Why
• How to use it
```

```
32/36
```

```
[pres] 0:vim* 2:gnome-screenshot-                     "parens" 09:48 29-Oct-22
```