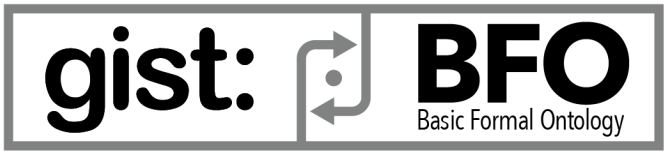
semantic arts

A BFO-ready version of gist

Dave McComb, Co-Founder, CEO, Semantic Arts



The Federal Government and Life Sciences Life Sciences companies are moving toward adoption of the Basic Formal Ontology (BFO).

> **We have aligned the Semantic Arts foundational ontology (gist) with BFO to help these communities.**

This paper describes how and why we did this.

Background

An upper ontology is a high-level data model that can be specialized to create a domain specific data model. A good upper ontology is a force multiplier that can speed the development of your domain model. It promotes interoperability and can be used as the basis for an information system. Two domain models derived from the same upper ontology are far easier to harmonize.

gist (not an acronym, but the word meaning "get the essence of") is an upper ontology, focused on the enterprise information systems domain. It was initially developed by Semantic Arts in 2007 and has been refined in over 100 commercial implementation projects. It is available for free under a creative commons' attribution license at https://www.semanticarts.com/gist/

BFO was developed at the University at Buffalo in 2002 and has been used in hundreds of ontology projects and cited in as many papers. The focus has been on philosophical correctness and has been adopted primarily in life sciences and more recently the federal government. It is available https://basic-formal-ontology.org

BFO and gist share a great deal in common:

- **Simple** – the current version of gist has 211 concepts (98 classes and 113 properties). The current version of BFO has 76 concepts (36 classes and 40 properties). We share the belief that the upper ontology should have the fewest concepts that provide the greatest coverage.

- **Formality** – most of the concepts within both ontologies have very rigorous formal definitions. The axioms within BFO are primarily defined in first order logic, which are not available to the owl-based editors and reasoners - but they have developed an owl

version. Half of their definitions are simple subsumption. The other half have subclass restrictions that don't have as much inferential value as equivalent class axioms. BFO is one of the few other ontologies we have come across that makes extensive use of high level disjoints. It is the combination of formal definitions with high level disjoints that is the best way to detect logical inconsistencies. gist is also highly axiomized. Half of all gist classes have full formal definitions of the equivalent class variety.

- **Breadth** – both BFO and gist were designed to provide covering concepts for the maximum number of domain concepts. A well-designed domain ontology, derived from either starting point, should have few or no classes that are not derived from the upper ontology classes. In the early days of gist, we created some domain classes without derivation. But as we evolved gist we now find "orphans" (classes not descended from gist classes) to be rare. BFO with its high-level abstract classes certainly has the potential to cover virtually all possible domain classes, but in practice we find many BFO compliant ontologies with large numbers of orphan classes.

- **Active Evolution** – both ontologies are in continual use and have active user communities. Both are well organized with major and minor releases including the ability to accept suggestions from users. Both are being used in production systems throughout the world.

Why Now?

In the early days of semantic adoption there were many options for an upper ontology. BFO, Dolce, Sumo and OpenCyc were considered the main contenders.

At Semantic Arts, we didn't see a need to adopt BFO or any of the other upper ontologies. They didn't contain the key concepts that we needed to implement enterprise systems, and they were very hard to explain to subject matter experts and project sponsors. We invest significant effort making sure our ontologies are understood by those that both implement and consume them.

Recently we have considered committing to both schema.org and ISO 15926. Neither of these purports to be an upper ontology. However, when we look at them in detail, we find they are pretty close to being upper ontologies by scope and positioning. In many ways these ontologies are more pragmatic and closer to what we are trying to achieve.

Schema.org is promoted by a consortium led by Google. Its primary use case is to make internet search more accurate by standardizing on many of the terms used for business descriptions. The pragmatic value for companies that tag their content with schema.org is major improvements in web searching. We also know that schema.org can be easily aligned with gist. This is how Schema App (https://www.schemaapp.com) built their offering. While schema.org is a good solution for finding and describing a company's offerings, it wasn't designed for our primary purpose, which is to help a firm run their business.

ISO 15926 emerged from the Oil & Gas industry and is widely used in process manufacturing industries. The architecture is abstract and, in theory, could be applied in a much broader way.

Up until now we didn't see much advantage in reducing flexibility in the pursuit of our core mission by committing to these candidate upper ontology and upper ontology-like models.

Motivation

We were driven to create an alignment with BFO based on input from some of our clients.

The first motivator is the huge volume of life science ontologies that (at least) purport to be based on BFO. The reason we say "purport" is that we have sampled many life science ontologies for their degree of commitment to BFO. Our measure of commitment is what percentage of their named classes are subclasses of BFO classes. Or to use the terminology earlier, the number of orphan classes they contain. We find many where fewer than half of the classes are proper descendants of BFO primitives.

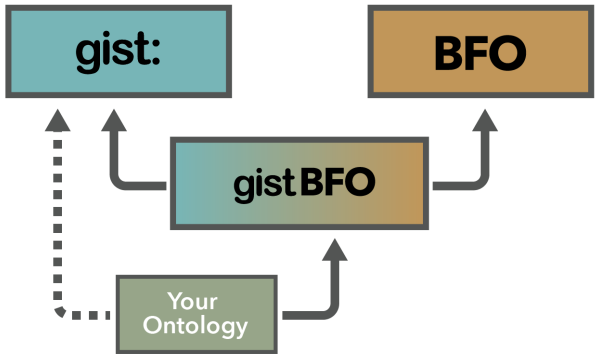
The OBO (Open Biological and Biomedical Ontologies) Foundry is a great resource for ontologies in the life sciences space. That said, there are over 8 million classes in OBO alone, that purport to conform to BFO, which gives other life science ontologies a reason to seek alignment.

The other development was the DoD's publication of "Principles of The DoD-IC Ontology Foundry" (which is still in draft status). In this document the DoD have declared that all ontology related work within the defense community shall conform to BFO (and the Common Core Ontology, which we will pick up in a subsequent white paper).

For people who must conform to BFO (the defense community) this provides them with a more pragmatic way to build domain models while still complying with the directive. For life science practitioners this also provides assurance that their work will align with life science orthodoxy.

How to Get Started

This illustration shows how the key pieces fit together.

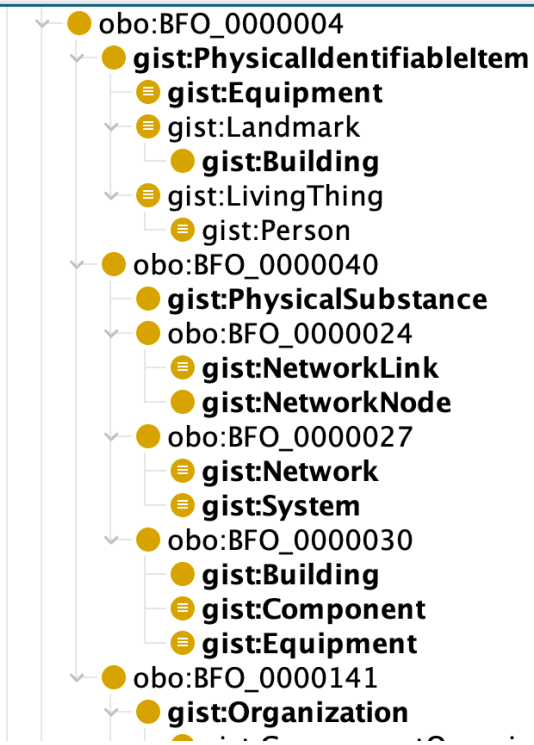


You can download gistBFO at
https://github.com/semanticarts/gistBFO

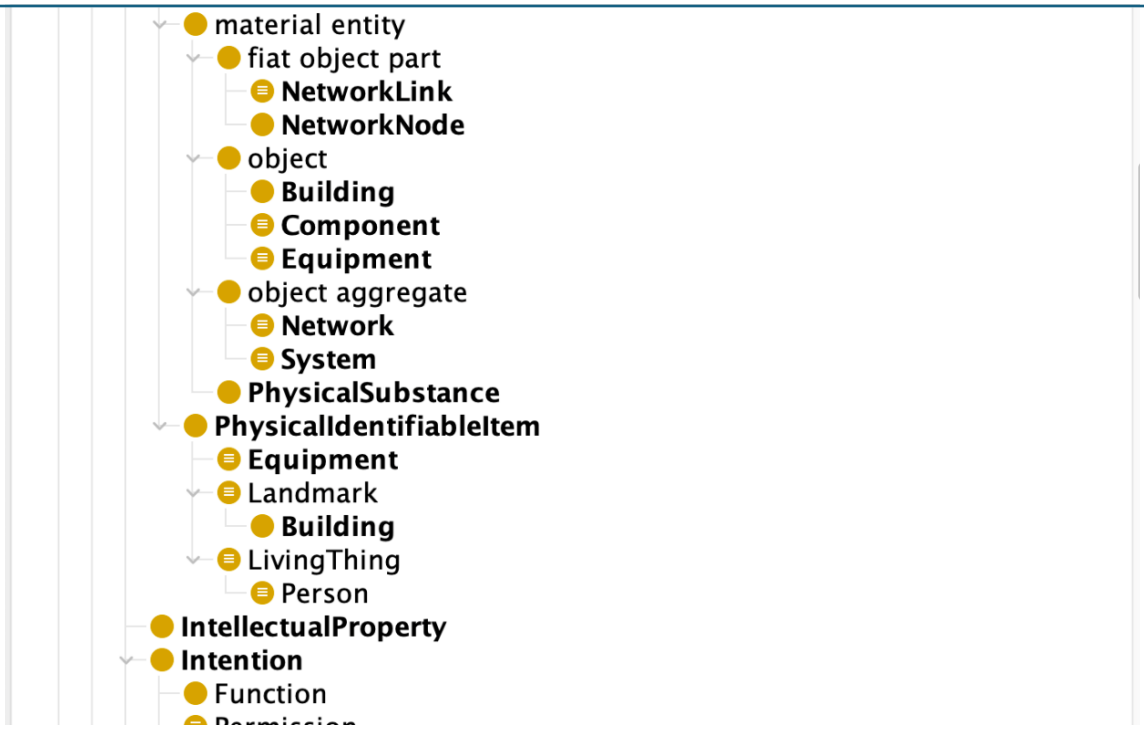
This file will bring in compatible versions of both gist and BFO. These arrows represent the import statements that bring in these ontologies. As we suggest in the tips section you may want to add the redundant import to directly import the same version of gist to your ontology.

This is what you will find when you look at the merged ontology in Protege. It is much easier to see which concepts came from BFO and which came from gist when you view using the "Render by prefixed name" option. The BFO class names are in the obo namespace, start with BFO and are numbers.

The capitalized terms starting with gist are from gist.



The alternative display "Render by label (rdfs:label)" it is still pretty easy to tell how they blended together. The BFO labels are lower case. (the order is slightly different because the labels sort differently from the class names, but the hierarchy is the same)



As you will see, almost all the gist classes are proper subclasses of BFO with three exceptions.

- **Artifact** – things that were intentionally built.
- **Place** – locatable in the world
- **Unit of Measure** – a standard amount used to measure or specify things.

The first two of these are convenience classes that group dissimilar items underneath. The "Artifact" class groups physical and non-physical things that were intentionally built. "Place" groups geospatial regions on the earth with physical items that we often refer to as places, such as landmarks and buildings. Because they subsume items that are disjoint, they could not be subsumed under a single BFO class. But each of their subclasses is aligned with BFO so there is no ambiguity there.

We were not sure where "Units of Measure" fit in BFO, so rather than create inconsistencies we opted to leave UoM out of the BFO alignment. CCO went with our first inclination, which was that it was a "generically dependent continuant" (in gist-speak "content"). In fact, CCO went further and said that it was "descriptive information content entity" which I suppose it could be. But these focus on the content-ness of the unit. A case can be made that a unit of measure (say "inch") is a special case, a reference

case or a magnitude, which in BFO is a "quality," and more specifically a "relational quality." For the time being we'll leave gist:UnitOfMeasure an orphan, but for any specific purpose if people knew that it would be safe, they could declare it a "generically dependent continuant."

If any of our alignments are inappropriate, we'd be happy to change.

We have done some alignment on the properties. There are some structural differences in the use of properties that will probably cause users of gistBFO to either use gist properties or BFO properties and not mix and match, however where there is some equivalence we've recorded.

BFO makes extensive use of inverse properties. There are only 6 properties in BFO that do not have inverses. After years of discouraging the use of inverses, we finally eliminated them altogether in gist. When using an ontology for definitional purposes inverses can be handy, but there are reasons to avoid them in production systems including ambiguity, inconsistency and performance issues. There is a brief white paper here: https://www.semanticarts.com/named-property-inverses-yay-or-nay/ and a longer discussion on here https://www.youtube.com/watch?v=uz2GVWadBjg&list=PLk2kJrehubb4dc3e5Db5Lvv9WMaOhV3V7&index=4

BFO uses domain and range extensively. Gist uses them sparingly. We have observed in other ontologies being over specific on domain and range has made properties less reusable and contributes to unnecessary concept bloat. Because of the abstractness of BFOs classes this isn't as much a problem, but it is a stylistic difference.

In BFO there are only four property pairs that participate in any class definitions: location of/located in, continuant part of/ has continuant part, has occurrent part/occurrent part of and has temporal part / temporal part of. We have aligned with these.

Some tips

We recommend anyone using gistBFO, and especially those that are contemplating building artifacts that may be used by BFO and non BFO communities, to primarily rely on the gist concepts when defining your domain specific classes. Doing so will make it far easier to explain to your stakeholders. And it will not sacrifice any of the BFO philosophical grounding as all the gist concepts (except unit of measure) algin with BFO.

The other advantage, suggested by the dotted line in the "how to get started" section, is that if you have defined all your concepts in gist terms, and you need to implement in a non BFO environment you can just drop the import of gistBFO and the BFO references will disappear, and nothing else needs to change.

If you are using BFO and gist in the Life Sciences arena, you might want to consider what we are doing with our Life Sciences clients: treating most of the classes in OBO as instances in an implementation ontology. Depending on your use case this might involve

punning (treating and class and instance interchangeably) or just use the rdfs:seeAlso annotation to resolve the instance to its class.

Coming soon: CCO ready gist

The Common Core Ontology is a DoD initiated project. It is more similar to gist in that the classes are more concrete and more easily understood by domain experts. It is different from gist in that it consists of 1417 classes and 275 properties and is growing. As such it is almost ten times as complex as gist.

We are working on alignment. Stay tuned, coming soon.