Cyc is a revolutionary AI platform
with human reasoning, knowledge,
and logic at enterprise scale

Trusted,
Transparent,
Actually **I**ntelligent

Technology Overview

info@cyc.com

Contents

1 Cyc is AI that Reasons Logically and Explains Itself Transparently

Cyc® is a differentiated, mature Artificial Intelligence software platform with a suite of vertically-focused products. Unlike Machine Learning (ML)—what most people are referring to when they talk about AI these days—Cyc's power is rooted in its ability to reason logically. ML harnesses big data to do statistical reasoning, whereas Cyc harnesses knowledge to do causal reasoning.

And Cyc works: leaders within Global Fortune 500 enterprises operating in the highly-regulated and highly-competitive healthcare, financial services, and energy industries trust Cyc in applications where their existing investments in ML and data science did not suffice. E.g., if reasoning has to be transparent and answers need to be auditable: ML can *never* give an explicit step-by-step explanation of its line of reasoning behind a conclusion, but Cyc *always* can. For some of these applications, Cyc is being used as a standalone solution, but more often it delivers value by working with and complementing the client's existing ML and database technology assets.

2 There's More to AI Than Machine Learning

Machine learning is one powerful form of Artificial Intelligence. Popular consumer applications such as Alexa, Siri, Netflix, Google Maps, Facebook's news feed, and Tesla's autopilot have been very successful at harnessing the power of machine learning to recognize patterns in massive amounts of data. Likewise for emerging enterprise applications such as robotic process automation, cybersecurity threat detection and response, network infrastructure management, insurance pricing, and the trading of stocks, bonds, and other financial instruments.

But human intelligence is much more than just pattern recognition, and Artificial Intelligence is much more than just Machine Learning. To effectively address your enterprise's mission-critical issues, a person—or a machine—needs to truly understand the problems they're tasked with solving in context, and understand each of the information resources available to call on in the course of solving those problems. To earn the trust of decision makers, he/she/it needs to operate transparently: to be able to explain their conclusions. Machine Learning can't do that, which is why there is a large gap today between high expectations for AI's potential and the reality of the business value actually being delivered. The effectiveness of machine learning tools is limited by the availability of the right kinds of clean, complete, and accurate training data, by a reliance on human data scientists to construct, tune, and refine the statistical models, and by an inability of those siloed one-off models to generalize across domains, functions, business units, or geographies.

As useful as ML can be, it identifies correlation, not causation; it requires a large volume of training data; and it can't explain its results. There is an alternative, a very different sort of AI technology which breaks all three of those bottlenecks.

For the last 35 years, Stanford computer science professor and AI pioneer Doug Lenat has led a large team of world-class computer scientists, computational linguists, philosophers, and logicians, painstakingly identifying and formally axiomatizing the tens of millions of rules that model the real world and human common sense. Cyc's power comes from (i) having that enormous knowledge base, and (ii) being able to logically infer things. Here are a couple examples to illustrate the difference between these two very different types of thinking, statistics (ML) versus logic (Cyc):

- A police officer may statistically profile a person based on his/her appearance and body language, versus actually *investigating* and *deducing* the person's guilt or innocence.
- Until WWII, the "engineering" of large bridges was done mostly by imitating other bridges and just hoping for the best. Today, we understand the material science of stress, load, elasticity, shear, etc., so mechanical engineering models can be built that prevent tragedies like those that the purely statistical approach led to (e.g., the 1940 collapse of the Tacoma Narrows Bridge.)

3 Neither of those approaches to AI (ML's and Cyc's) are new

AI researchers have been training neural networks to recognize patterns since the mid-1960s and have been developing knowledge-based systems capable of performing multi-step logical reasoning since the early 1970s. The former were called "perceptrons" back then, and the latter were called "expert systems". These were, and are, two very different ways of drawing inferences and powering an AI. Both paradigms looked promising, at first, but then each approach encountered enormous obstacles which stalled their progress for several decades. Several things have changed in the last 50 years, which has made it cost-effective, finally, to revisit—and harness—both sources of power.

In both cases, scaling them up demanded vastly faster and cheaper computers, and vastly larger and cheaper storage. Computing cost and speed, and memory cost, size, and speed, have of course improved by several orders of magnitude—more than a factor of 100,000 each—since 1970.

In the case of so-called perceptrons or neural nets, two important additional "missing links" necessary to make machine learning commercially practical were access to sufficient training data and inexpensive parallel processors. Today, thanks to the internet, cloud computing, and databases, Big Data is ubiquitous, while the huge global gaming market has yielded fast, inexpensive, massively parallel GPUs.[1]

In the case of so-called expert systems, it turns out that there were actually **150 additional bottlenecks and missing technologies** back in 1970 (in addition to the need for 100,000x faster/cheaper computers and storage, and access to online data). Here are three typical problems, out of that set of 150, and the solution we finally engineered to handle each one:

1. **Reusability**. Building each knowledge-based system was a long, labor-intensive process, so expert system knowledge engineers inevitably "cut corners" in ways that made their IF/THEN rules almost never reusable in later systems. Each new

[1] Graphics Processing chips which carry out, in parallel, the sort of computations that gate video game frame rates. The very same GPUs turned out to be quite well-suited to parallelize the low-level computations needed by ML.

application had to be built from scratch. To remediate this, we had to identify, collect, and formalize the tens of millions of general rules of good guessing and good judgement that comprise general human common sense.

2. **Efficiency**. Logical reasoning (running a set of IF/THEN rules) was painfully slow, even when there were only a few hundred rules. To remediate this, we identified and implemented over 1,000 special-case reasoners, each with its own idiosyncratic data structures and algorithms. These work together cooperatively as a sort of community of agents to usurp the need for a general (but hopelessly slow) theorem prover. Some of these stylized reasoning agents are domain-dependent, such as how to efficiently balance a chemical equation, and some agents are general, such as caching transitive relations like `during` and `subOrganizations`.

3. **Inconsistency**. Rule-based systems did not deal well with the inevitable inconsistencies of rich, real-world information: once they deduced *False*, bad things happened (per classical logic; see also any episode of Star Trek where a computer is inconsistent.) To remediate this, we had to replace the requirement of *global* consistency of the knowledge base with the notion of *local* consistency.[2] This means having Cyc explicitly model the context in which rules' premises and conclusions are true. We also had to have Cyc reason not by theorem-proving of *True* versus *False* but rather coming up with all the pro and con arguments it possibly can, in each situation, eliminating the self-contradictory ones, and then reasoning about the remaining arguments to decide what to believe in that context. Each context, also sometimes called a "micro-theory", is a first class object in Cyc's ontology, and can be reasoned about by Cyc; that enables it carry out the necessary meta-level reasoning it needs to do *about* arguments.

Over the last three decades we have identified (that's a euphemism for "*painfully tripped over*") and solved (i.e., eventually engineered our way through or around) the other 147 breakthroughs required. To briefly touch on just three more of those 147 critical problems which Cyc had to overcome:

4. Semantically mapping between Cyc's ontology and the schema elements in third-party information sources such as databases and web services so that Cyc can remotely query them when/as appropriate, just as you or I or a subject matter expert would;

5. NLG (natural language generation) to explain Cyc's reasoning chains step-by-step so end users can audit, and therefore trust, the system. In cases where the user disagrees with Cyc, they're able to offer feedback and even provide their own knowledge to override Cyc (in that context); and

[2] A good analogy is how we all know that the surface of the earth is roughly spherical, but we live our everyday lives as though it were flat, and that works well for us almost all the time because it is *locally* flat. In much the same way, Cyc's knowledge base is organized in a multidimensional context space, with nearby contexts being mostly consistent with each other. As inference proceeds, it reaches farther- and farther-flung contexts, and the inevitable contradictions which are encountered are treated just as a sign to stop reasoning further in that "direction".

6. Meta- and meta-meta-level reasoning so Cyc can cope efficiently with amounts of data and numbers of rules that are *four orders of magnitude larger* than anyone else in symbolic AI ever tried to tackle.

As a result of our quarter-billion-dollar “pump-priming” and “infrastructure-building” effort, Cyc today is a mature knowledge representation and reasoning platform that has delivered value in hundreds of deployments for commercial clients and government agencies. It’s real, and it works.

4 So. . . What Exactly Is Cyc?

The Cyc platform consists of four mutually reinforcing components:

1. **The Cyc Knowledge Base**: Built over the course of the last 35 years, by 2000+ Ph.D. scientist-years’ effort, Cyc’s ever-growing common-sense and domain-specific Knowledge Base (KB) is the broadest, deepest, most complete AI tool ever developed. It understands (represents fully) real world contextual nuance that other AI can’t, like culture, emotions, time, space, beliefs, and bias. The Knowledge Base comprises:

    - An **ontology of about 1.5 million general concepts** (e.g., taxonomically “placing” terms like eyes, sleep, night, person, unhappiness, hours, posture, being woken up, etc.);
    - More than **25 million general rules and assertions involving those concepts** (e.g., that most people sleep at night, for several hours at a time, lying down, with their eyes closed, they can be awakened by a loud noise but don’t like that, etc.); and
    - **Domain-specific extensions** to the common sense ontology and knowledge base in areas such as healthcare, intelligence, defense, energy, transportation, and financial services.

    Each element of the Knowledge Base is expressed in a representation language, “CycL”, which is 100% as expressive as natural languages such as English, but is also formal enough that there is an algorithm for producing, automatically, exactly the same conclusions that a person might draw from a set of assertions expressed in CycL. Even the number of *one-step* inferences it can draw is in the trillions, and Cyc typically produces reasoning chains hundreds or even thousands of steps long, during an application deployment. The next component is the key to doing that in real time, at enterprise scale, something that a general theorem prover couldn’t do:

2. **The Cyc Inference Engines**: A battery of **over 1,100 inference engines** which inter-operate as a community of agents to collaboratively attack and efficiently solve problems or questions posed to Cyc in an application. Some are general (e.g., doing graph walks efficiently) and some are domain-specific. Pro/con argumentation and the Cyc context mechanism enable it to find all the step-by-step reasoning chains for and against each plausible answer, and an NL generation system almost

instantly converts those potentially 1,000-step-long logical arguments from CycL into understandable English (selecting the few key steps worth communicating to that user, and letting them drill down for more detail, recursively, wherever they want more fine-grained explanation).

3. **Tools Enabling Intelligent Data Selection and Access**: Cyc complements its own knowledge by accessing the data it needs to solve problems, wherever that data exists. Just as with a human practitioner doing that task, Cyc "looks up" what it needs, as it needs it. This is a sort of *virtual* data-warehousing or data-laking: Cyc does not do a wholesale import of the data into its Knowledge Base. In order to access customer and/or 3rd party databases and web services, a semi-automated schema-mapping tool (which itself uses Cyc!) is included as part of the Cyc platform.

4. **Interfaces and software tools** for creating, modifying, testing, and deploying Cyc's current products and applications across a wide range of cloud- and device-hardware, software, and OS environments. Also included in this component are utilities for editing and extending the KB, creating and running regression tests, and "containerizing" Cyc images for security purposes.

Those four components come together to form Cyc: an Artificial Intelligence reasoning platform and software application development environment used by many of the largest companies in the world to address mission-critical problems in ways not possible by other forms of AI. Besides those four main components, which enable the development of new bespoke Cyc applications, Cycorp clients can also access and leverage a suite of ready-to-deploy industry-specific and capability-specific products that have already been developed on the platform, which in many cases are all that they need.

5 Cyc's Knowledge Base

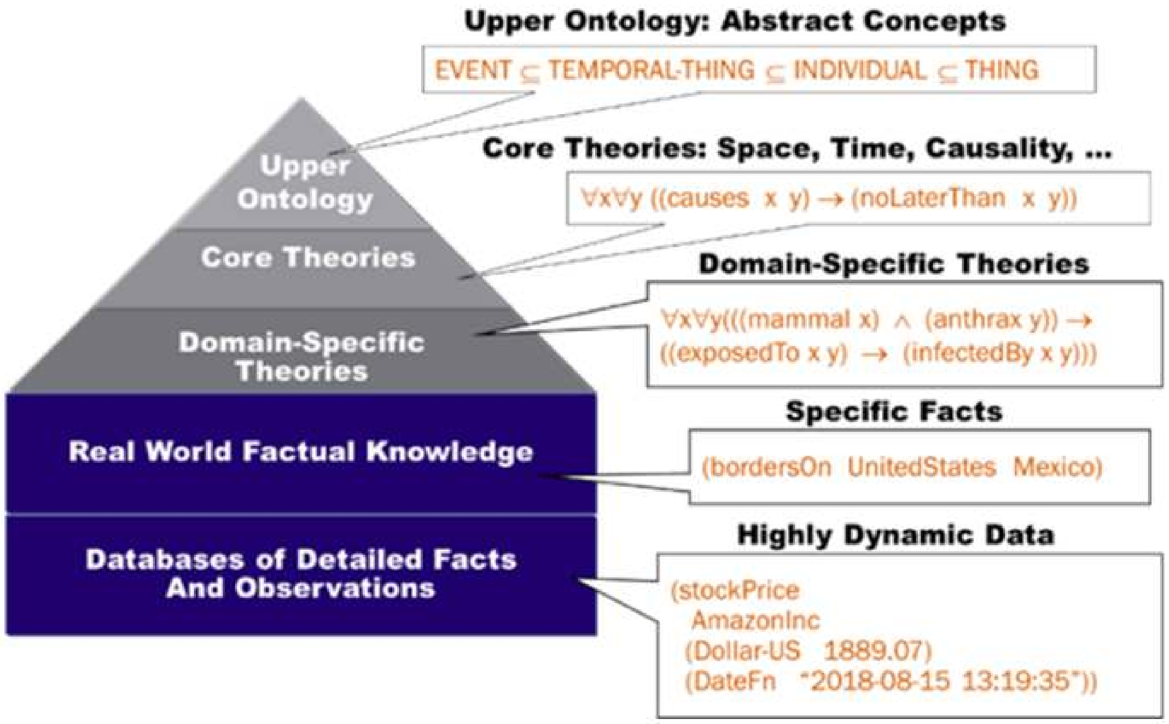
The common sense knowledge in the Cyc KB spans every domain of human experience, including science, culture, weather, financial systems, health care, history, politics, daily human activities, etc. It captures and can reason with the fundamental rules of thumb about "how the world works" that people typically just take for granted, such as:

- No two objects can occupy the same space at the same time.
- Causes start at or before the time that their effects start.
- Human-scale inanimate objects usually don't move on their own without some external force.
- If you own something, you own all of its physical parts.
- An electronic device that is turned on will go off if/when its power supply is interrupted.
- Low supply and high demand together typically lead to price increases.

- Restricting flow in a fluid conduit increases pressure upstream of the restriction.

These rules have wide applicability—that last one, for example, has been used successfully by Cyc in applications where it had to reason about blockages in oil pipelines, about human blood vessels, about traffic arteries, and about drinking straws. To build a Cyc application, then, we start with the entire Cyc KB and extend it with a relatively small number of application-domain-specific terms and rules.

One can visualize the Cyc KB as a pyramid with the most general rules at the top. As you descend through the layers, the axioms get more numerous, and more context-specific, and more prone to having exceptions. E.g., "*Birds fly*" is encoded in Cyc along with the obvious exception clauses: *Except for birds with broken wings, penguins, cassowaries...* Humans learn this type of general knowledge effortlessly and use it constantly, but it's rarely said out loud or written down; and it is not known by, let alone used by, other AIs.


Upper Ontology: Abstract Concepts
EVENT ⊆ TEMPORAL-THING ⊆ INDIVIDUAL ⊆ THING
Upper Ontology
Core Theories: Space, Time, Causality, ...
∀x∀y ((causes x y) → (noLaterThan x y))
Core Theories
Domain-Specific Theories
∀x∀y(((mammal x) ∧ (anthrax y)) → ((exposedTo x y) → (infectedBy x y)))
Domain-Specific Theories
Specific Facts
(bordersOn UnitedStates Mexico)
Real World Factual Knowledge
Highly Dynamic Data
(stockPrice AmazonInc (Dollar-US 1889.07) (DateFn "2018-08-15 13:19:35"))
Databases of Detailed Facts And Observations

Every Cyc application has access to the *entire* Cyc KB—from the upper ontology, through all the middle-level theories of how the world works (weather, traffic, etc.), down to and including all the domain-specific "layers"—with the exception of course of proprietary customer-specific knowledge. This means that unlike one-off ontologies or "knowledge graphs" built from scratch as bespoke solutions, the Cyc KB need only be slightly extended for each new application. Over time, as Cyc has been used in particular domains—for example, in health and medicine, banking and finance, or oil and energy—thousands of additional widely useful domain-specific axioms have been added to the Cyc KB and products have been built utilizing this knowledge to solve industry and cross-industry issues. So, even when developing a bespoke new application using Cyc, a client doing so is standing on the shoulders of the full Cyc KB.

For example, to build a particular new client-specific Cyc-based application in the domain of enterprise network security, it was only necessary to expand the Cyc KB by a factor of .0001 (one ten-thousandth) because Cyc already had rich knowledge of the types of information-bearing things (such as prose text, emails, computer viruses, software applications, and operating systems), concepts of information flow and transfer, and representations of human agents (e.g., potential hackers) with motives, drives, and behavior patterns. So Cyc can be thought of as a new employee who has all the common sense knowledge one would expect an adult to have, and who has worked in the computer domain before. Training such a new employee is simply a matter of building on top of his/her pre-existing knowledge. Cyc-based application developers can take advantage of the same kind of experience because each application builds on the previous ones.

Because the Cyc platform provides an enduring and expanding foundation for future applications, it offers unique benefits to potential clients. The persistence of reusable information means that time and money needn't be wasted re-entering basic principles that are true and necessary for multiple applications.

Another, even more powerful way of thinking of this is as a sort of ***knowledge network effect***: The time and resources required to build your enterprise's $(n+1)^{\text{th}}$ application *decreases as n increases*, because each new application models more and more of your enterprise's structure, policies, procedures, relevant internal and external structured data sources, and Cyc never forgets what it has been taught. The next application is likely to use some or most of that codified enterprise knowledge, which has already been represented and is available for use with no further effort. Cycorp's clients have often taken advantage of this knowledge network effect, benefiting from an accrued "enterprise knowledge equity" in Cyc by reusing domain-specific knowledge in follow-up applications.

There are also benefits in terms of innovation: unlike silo'ed one-off applications, which limit themselves by focusing on producing a narrow range of predetermined outcomes, Cyc can employ its full range of commonsense and expert knowledge to produce novel and unexpected results. Some of the most dramatic Cyc applications are ones where it is abductively generating plausible novel "what if" scenarios about alternative courses of action and possible futures.

5.1 Cyc's Representation is More than Just a Knowledge Graph

We call Cyc's repository of knowledge a "Knowledge Base" (or "KB") to distinguish it from both databases and knowledge graphs, although it shares some features with each of them:

- Like a database, Cyc's KB can be queried via structured queries. But, unlike a database, the Cyc KB contains general reusable knowledge in the form of principles and axioms such as "Public companies are incentivized to maximize profits to benefit their shareholders", instead of just specific facts like "Shares of AMZN were trading at US $1,670.43 at 6:20 PM EST on January 29, 2019".

- Like a knowledge graph, the Cyc KB has a formal semantics that enables new information to be automatically, algorithmically inferred from what is explicitly stated. But knowledge graphs are limited in terms of the expressiveness of the language in which they are written: a "knowledge graph" consists only of "triples" that are binary relations (analogous to the "edges" in a graph) that connect entities (analogous to the "nodes" in a graph). In other words, knowledge graphs are limited to storing knowledge in three-word sentences, like `height(EiffelTower)= 984'`. But CycL—the language of the Cyc KB—is not limited to binary relations: rather, it's as expressive as English. Can you imagine trying to translate all of English—including, for example, today's lead CNN article or a Shakespearean sonnet—into three-word sentences? It's probably possible, but it would be almost impossible to understand, and especially to extract the intended subtexts. CycL is full HOL (higher order logic) and allows for just as much nuance and elegance as English or any other human natural language—much more than any triple-based knowledge graph language (such as OWL or RDF).

For example, here's how we tell Cyc that "In any fluid flow in a pipe-like object, decreasing the diameter of the conveyor increases the pressure of the flowing fluid in all locations of the pipe-like object upstream of the diameter decrease":

```
(causes  FliuidFlow-Translation  Pipe-GenericConduit
  (DecreaseOnSlotFn  diameterOfObject)  conveyor-stationary
  (IncreaseOnSlotFn  fluidPressure)  objectMoving  upstreamFrom)
```

This is a 7-arity declaration of a principle, and a lot of the nuance it elegantly captures would be lost if we tried to force this axiom into one or more of the three-word sentences allowed by a knowledge graph. And this was a particularly simple Cyc assertion—a ground atomic formula (GAF), since it contained no variables, no quantifiers, and no logical connectives (like *and*, *or*, *not*, *if*, *then*,...). Things get more complicated in English—and in Cyc's representation—when one needs to express things like nested modals ("Israel believes that Syria wants the U.N. to expect that...").

Another differentiator between the Cyc KB and a typical "knowledge graph" is that knowledge graphs require global consistency within a "namespace". Cyc, in contrast, allows for global inconsistency and requires local consistency only in the context of individual micro-theories in the KB. This means that Cyc is able to model conflicting points-of-view, which is beyond the reach of a traditional knowledge graph. For example, Cyc can model Expert 1's rules of thumb in one micro-theory, and Expert 2's rules of thumb in another micro-theory, even if those two rule-sets occasionally contradict one another. By modeling both experts, Cyc can present an answer of the form "*Expert 1 would advise you to do X for the following reasoning chain, and Expert 2 would instead advise you to do Y for this different line of reasoning...*". This also forms the basis of the sort of pro/con reasoning that Cyc supports—even a single expert can argue for and against a particular answer—and which our clients tell us helps keep them from falling into cognitive traps like confirmation bias, anchoring, and group-think.

5.2 Expressive Knowledge Representation—What It Is and Why It Matters

Axioms are encoded in the Cyc KB in Cyc's representation language, CycL, which is fundamentally more expressive than any other ontology representation language in use today. The CycL language is as expressive as English, Chinese, etc.: anything at all that anyone can say in any natural language can also be said in CycL *in full*. What does that mean? If a person would infer Z from statements X, Y,... in English, then Cyc will infer the CycL representation of Z from the CycL representation of statements X, Y,...

CycL can be thought of as full First-Order Predicate Calculus (FOPC) with all the voluminous higher-order and modal extensions which have proven to be pragmatically useful in the last 35 years. Of course all of the FOPC connectives, such as *and*, *or*, *not*, and *implies*, are present, as are the universal ("*For all...*") and existential ("*There exists...*") quantifiers.

Terms in CycL can denote individuals (`NewYorkCity`, `PabloPicasso`), natural kinds (`Platinum`, `OakTree`), and relations (`motherOf`, `between`). Unlike the vast majority of other ontology languages in the world—most notably, OWL and Knowledge Graphs—CycL allows relations to have any number of arguments. As one might expect, the one-place predicates (collections or types), two-place predicates (binary relations), and three-place predicates are the most common.

CycL also includes functions—e.g., `LiquidFn` and `BorderBetweenFn`—which can be applied to existing terms in order to construct new terms. This permits the compositional creation of an unlimited number of new "non-atomic" collections and individuals, such as `(LiquidFn  Nitrogen)` which denotes the liquid form of nitrogen, and `(BorderBetweenFn  Sweden  Norway)` which denotes the boundary that separates Sweden and Norway).

One crucial extension CycL makes to FOPC is the ability to handle default reasoning. "Dogs have four legs" is *default* true, and in CycL can be expressed as

```
(relationAllExistsCount  Dog  anatomicalPart  Leg  4),
```

but of course many dogs, alas, have fewer legs. Therefore, standard truth-conditional logic, in which statements are only either true or false, is insufficient. Every axiom in the KB is either *default true* (i.e., generally true with exceptions) or *monotonically true* (i.e., true without exception). Exceptions can be systematic (birds fly, *except* penguins don't) or individual (*that* dog has only 3 legs). Default axioms can be overridden by new knowledge, which Cyc can learn about from a person teaching it, or from reading it from an online data source, or from its own extended "musings" where Cyc itself infers that some individual or some collection must be exceptional in a certain way.

Each assertion in Cyc has one or more justifications—bottoming out in "*that's what I was told to be true in context C by source X at time Y*". An important phrase there is "one *or more*": Instead of using only a single support or line of reasoning to determine if an assertion is true or false, Cyc's inference engine keeps all pro and con arguments available—each argument consists of justification chains showing which proofs (ground facts, rules, and inference methods) were used to arrive at a conclusion. Whenever Cyc needs to decide something, it carries out a decision-making process called argumentation: weighing all the various arguments, pro and con, to see which are *preferred* to which other arguments, to arrive at a truth value for the conclusion. Incommensurable strong pro- and con- arguments result in Cyc deciding not to trust that statement or its negation, which is the right course of action.

5.3 The Degrees of Expressivity of a Knowledge Representation Language

We have ordered them, here, from least to most expressive, and also from most to least efficient:

- **0th order (Propositional Logic)**: Each rule or fact is an opaque token. If I tell you that P, Q, and R are true, you can't conclude much from that, without knowing what those *mean*, and neither could an AI that used only that level of representation.

- **$\frac{1}{2}$th order**: There are a large family of restricted subsets of first order logic which introduce variables and a fixed set of relations and some limited sorts of reasoning with them. E.g., one can say *P: Dogs are Mammals*, and *Q: Mammals are Animals*, and *R: Fido is a Dog*, and such a system can then mechanically conclude that Fido is an Animal. Included here are knowledge graphs, quad- and triple-stores, and RDF. Also here is representing English sentences as strings of words, for sentiment analysis and syntactic searching.

- **Full 1st order (Predicate Calculus)**: Rules can have any number of nested `ForAll` and `ThereExists` quantifiers on their left and right hand sides. E.g., "*If someone is released from prison, there had earlier been a trial at which they were found guilty of some crime which happened still earlier...*".

- **2nd order logic**: Variables can range over predicates. E.g., one can ask: "*Through what pathway are x and y related?*".

- **Higher-order logic (HOL)**: Variables can range over statements; e.g., "Rule x was entered earlier than rule y, but by a novice". Modal operators are allowed, expressing that someone *aims* to make something true, or *believes* that it's true, or *dreads* it being true. Nested modal and temporal operators occur: "*Fox News reported today that The US Supreme Court believed last year that high level White House staffers wanted to delay...*". If you can say something in English, you can capture its full meaning in HOL. I.e., any argument one might carry out in English can be 100% captured and *automatically* carried out in HOL.

5.4 Knowledge Reuse

The term "knowledge reuse" has two slightly different meanings, and fortunately the news is good for both of them:

1. When a Cyc application **is being deployed**, typically over 99% of the knowledge that would need to be represented, in order for the application to operate, is already present in Cyc.

2. When a Cyc application **is running**, typically over 95% of the knowledge that gets used in producing an answer and its justification are things that Cyc learned years earlier—on average about ten years earlier! For example, in the case of a

Cyc application for the Cleveland Clinic[3], to reason about medical procedures, such successfully-reused knowledge included:

- "*Once a physical part has been removed from an object, it generally can't be removed again.*"
- "*When there's a blockage in a conduit, the pressure/backup increases upstream and decreases downstream, and the overall flow through that conduit decreases.*"

The first rule of thumb was useful for disambiguating that in the query "*. . . those patients who had appendectomies in 2016 <u>and</u> 2017. . .*", the word "*and*" must actually mean "*or*". And the second of those two rules of thumb was useful in dealing with blood flow in arteries and veins, but had been earlier used to deal with oil pipelines, roadway traffic patterns, weather phenomena, pumping water, and using a drinking straw.

5.5 Cyc's Knowledge is Easy to Customize for Individual Clients

As we just noted, for any *particular* use-case, Cyc generally turns out to already contain more than 99% of the knowledge it will eventually need to serve as the basis of that particular knowledge-based application. The good news is that adding the next 1% to customize a Cyc product for a client is very straightforward. Typically, that expansion of the Cyc knowledge base happens in two phases:

Phase 1: Cycorp ontologists spend some time (often as little as 4 or 5 hours) interviewing domain Subject Matter Experts (SMEs)—typically either the top internal client experts or the industry's top consultants hired by Cycorp—and kick-start the entry of the new domain knowledge in Cyc, while simultaneously ensuring that Cyc can use its knowledge of the domain to generate the sorts of contextually appropriate tools that will enable future democratization of the knowledge entry process. Cycorp is motivated to end Phase 1 as quickly as possible, to hand off responsibility for adding and curating Cyc knowledge to the client as quickly as possible.

Phase 2: The client uses Cyc-generated knowledge entry tools to fine-tune, maintain, and enhance the Cyc KB as needed to support more and more uses for Cyc across the client's applications. These tools are in the form of web-based forms with type-in boxes and drop-down menus that are tailored specifically to the client's needs, along with additional tools that walk users through the process of helping Cyc to learn the specific principles and axioms that will drive Cyc applications.

5.6 Embracing Inconsistency in the Knowledge Base

It is useful, and sometimes essential, to represent conflicting theories and perspectives. There are many dimensions of this "context-space": things are true at one time but false at another; true at one level of abstraction but false at another; true in one person's belief system but false in another's; and so on. All this is represented explicitly in Cyc, by asserting facts, rules, arguments, etc. in an explicit context. Very few things are true in a completely context-free universal theory! Contexts help avoid contradictions; for

[3] Lenat *et al*, "Harnessing Cyc to Answer Clinical Researchers' Ad Hoc Queries", *AI Magazine*, Fall 2010. https://www.aaai.org/ojs/index.php/aimagazine/article/view/2299

example, one application might model alternative courses of action for an enterprise, and the various consequences shouldn't be clumped together in one general "predictions for next quarter" report since the different scenarios will likely predict significantly different outcomes. In other words, Cyc has therefore been built to maintain *local* rather than *global* consistency. This is achieved by situating each axiom in a particular *microtheory* (essentially, an explicitly represented logical context). Axioms within a single microtheory must be consistent with one another, but need not be consistent with those in other microtheories. Microtheories are first class objects in Cyc's ontology, and it can reason about them just as it reasons about anything else in its knowledge base.

Most of the axioms in a microtheory tend to share many background assumptions. The microtheory mechanism allows these assumptions to be stated once, at the level of the microtheory, instead of having to be repeated for each affected assertion. For example, a Cyc microtheory about normal physical conditions contains axioms such as "*Fluorine is a gas*" and "*Glass has a high amount of shear strength*". These axioms share the domain assumptions that temperature, pressure, etc. are in the normal range for 21st century Earth's surface.

One particularly useful application of Cyc's microtheory mechanism is the modeling of various subject-matter experts in a single field of expertise who have distinct skills, experience, educational backgrounds, training, judgements, problem solving approaches, and opinions; given the same input data they may come to different and possibly even contradictory conclusions, but Cyc can represent competing expert reasoning by placing it in distinct microtheories.

Microtheories are also used to sequester proprietary content in the Cyc KB. In a Cyc-based application that uses this proprietary knowledge, these microtheories would be visible, meaning the content that is represented in them can be used in a proof. A different application that does not have access to these microtheories could be used to answer the same kinds of questions, but it would likely get very different proofs.

Another benefit of Cyc's microtheory mechanism is that it also allows for more efficient inference. When solving certain types of problems, Cyc knows, or is told, that some microtheories must be consulted, while others are irrelevant. This reduces the search space, speeding up the inference process. This is a good segue to our next topic.

6 Inference in Cyc

Cyc performs three types of reasoning during inference:

1. **Deduction**: *If we know that all XYZCo board members met in person at a certain place/time, and B was seen elsewhere at that time, then infer that B is not a board member of XYZCo.* (This is Cyc's primary mode of reasoning.)

2. **Induction**: *Since 83% of all the transportation-related attacks in Spain in the last 40 years have been committed by ETA, tentatively assume that this new one was also done by ETA.* (This is the only type of reasoning used by ML.)

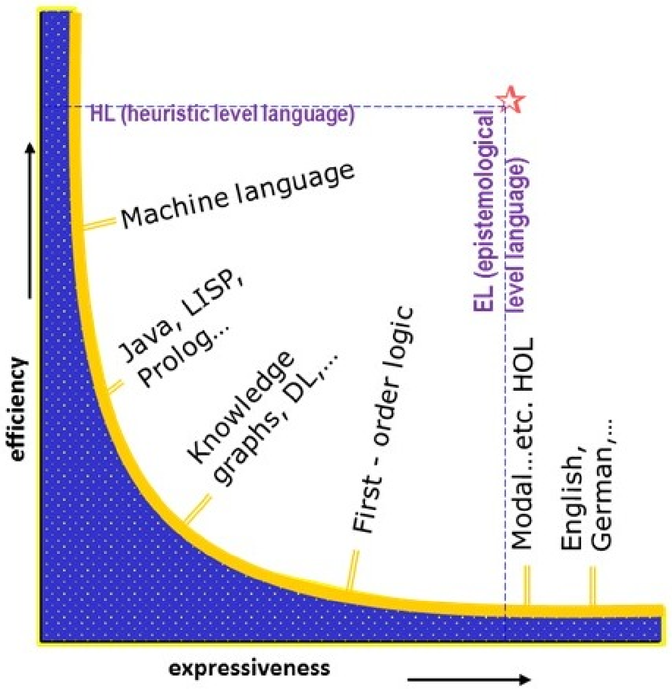
3. **Abduction**: *If pressure appears to drop at one end of a conduit, maybe there's a leak or blockage somewhere along its length.* There are many other possibilities, but this is a good one to check! As another example: *If a vacuum cleaner suddenly goes*

off, see if it has come unplugged. Most of the reasoning that the Sherlock Holmes character calls "deduction" is actually multi-step abduction.

Cyc can also reason by Analogy. Although deduction is sound (100% guaranteed to be reliable), analogy, abduction, and induction are not sound, but they are very useful, and typically they form the heart of a human experts' rules of good judgement. Since almost all the knowledge in Cyc's KB is just *default-true*, even deduction in Cyc is not sound, but, as with human expert reasoning, it is generally the most reliable that it is possible to be. This lack of monotonicity (absolute, permanent truth) is also why it is so important, for almost all Cyc applications, to have the system provide multiple answers, not just one, and provide pro and con arguments for and against each answer, not just present them as oracular truth.

6.1 Reasoning Efficiently

One of the pitfalls of highly expressive languages like CycL is that they can be inefficient to reason over (using Resolution). Since efficient and effective inference over KB content is a requirement for any Cyc-based application, Cyc developers have actually created two different but logically redundant representations of every assertion, fact, rule, etc. that Cyc knows. The examples that have been shown thus far have been written in the highly expressive "Epistemological Level" (EL), which is very similar to the formal sentences that logicians would write in symbolic logic. EL is effectively as expressive as natural language (NL) (i.e., anything that can be said in a NL like English can be represented in CycL), but it is not very efficient. In contrast, the "Heuristic Level" (HL) is on par with programming languages like Lisp and Java, but it is very efficient to reason with. Any CycL statement written in EL can be translated into a corresponding logically equivalent statement in the HL. Because Cyc translates EL expressions into HL expressions, it's as performant as it needs to be for any application.



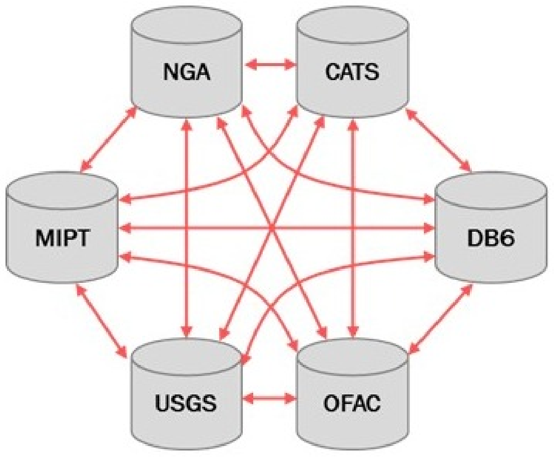
Consider the axiom "Dogs are at least partially tangible". At the EL, Cyc ontologists represent this rule as `(genls Dog PartiallyTangible)` (i.e., anything that is a dog is also at least partly tangible—it has mass, occupies space, etc.). This content can be captured at the HL in the `genls` *hierarchy*, and there are a dozen links between `Dog` and `PartiallyTangible`: `Canine`, `Mammal`, `Animal`, etc. Computers are very efficient at traversing graphs like this, and even more efficient if the transitive closure of the whole graph of that binary relation `genls` is—redundantly—precomputed and stored. In that case it is trivial to conclude quickly—in one step!—that the axiom in question is true. In fact there are over 1,100 redundant representations, not just one, and a CycL assertion might be represented in many different ways, using special data structures for each one and special algorithms that obviate the need to do logical theorem-proving. Think of these working together as a community or team of agents: when a top-level query is being worked on, all the HL modules who could help "raise their hands". One or two get chosen to work for a while on the problem, and they will recursively broadcast pleas for help on subproblems.

There is a second major way that Cyc is able to reason efficiently: meta-reasoning. 90% of the time, Cyc is calling on HL module agents to try to solve the current (sub)problem it's working on. 9% of the time, Cyc is sitting back and metaphorically puffing on its pipe, running meta-level rules deciding whether it should try a different tactic—e.g., trying to disprove something instead of trying to prove it. And 1% of the time, Cyc is running meta-meta-level rules to think about whether it should pick a whole different strategy. Knowledge-based systems historically gained efficiency by eliminating some of their knowledge—it's worth noting that Cyc does the opposite: it adds more meta- and meta-meta-knowledge to gain focus and efficiency; in many applications, this kind of knowledge is exactly what separates the very best experts from the average practitioners.

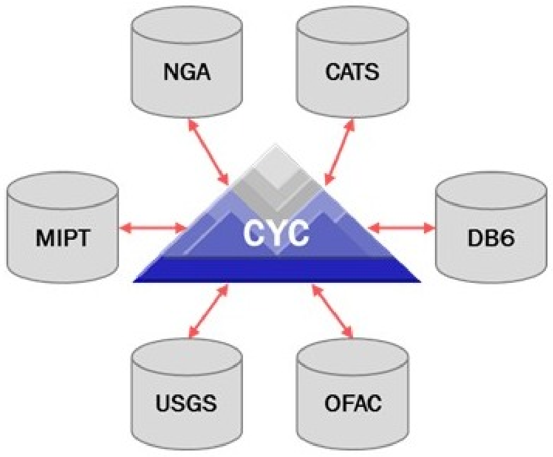
7 Intelligent Data Selection and Virtual Data Integration

It's useful to make a distinction between *knowledge* (the underlying heuristics that allow us to reason) and *data* (facts or statements about specific items in the world): Cyc's knowledge of common sense and domain-level axioms allow it to understand data. Cyc technology allows knowledge and data to each be stored and accessed by Cyc in their own appropriate way. Many kinds of data are best maintained outside of the Cyc KB—e.g., streaming, in databases, or as web services maintained by third parties. For example, stock prices, weather data, company part specs and price lists, etc. Directly encoding in the Cyc KB the stock price of each company on the New York Stock Exchange would be pointless, since that information would rapidly become stale. Rather than continually updating that data within the KB, it is much more appropriate to consult that data at the source, in real time, so Cyc can always use the most up-to-date figures.

This approach allows data to be used by Cyc as if they were actually present in the Cyc KB and approach multi-database interactions as a linear mapping problem as opposed to a quadratic one, which is what other data-warehouse or data-lake solutions require. As such, Cyc's approach requires far less work and is more stable over time. A further advantage Cyc has over those other sorts of solutions is that the customer data continues to live in customer databases, with no need for data migration.



Without Cyc



With Cyc

In other words, Cyc reduces the n^2 problem of integrating n structured data sources down to a linear factor of n. While the Cyc knowledge base does contain some "facts", most of its content—and its power—lies in the general, broadly applicable rules of thumb it contains. Cyc isn't a database; it accesses and calls on the relevant databases and web services—whether they are client sources or third party sources—much as a human performing that same application task would do, without memorizing that data ahead of time. This "semantic knowledge source integration" (SKSI) lets Cyc achieve a sort of *virtual* data-laking or *virtual* data-warehousing: the actual data stay in place wherever it resided, which is especially important if it is being maintained by some third party, and/or if the data is enormous in size, and/or is volatile in its rate of updates. Cyc acts as a sort of *interlingua*, in these applications.

Cyc's ontology is aligned with the schema of a data source—let's say a relational database—by giving Cyc the correspondence or rule that connects each relation to Cyc rules or terms. Cyc is then able to automatically generate arbitrarily complex SQL queries to access any particular field in any particular row, and understands how to interpret what it finds there. Cyc can also automatically query other sources of data as well. These mappings allow Cyc to combine information from multiple databases along with the knowledge in Cyc itself to answer user queries. Similarly, one can explain, in Cyc terms, when and how and why to call on some particular online data service.

8 Cyc Applications

More than half of the largest 15 companies in the world, and their employees, have relied on Cyc applications to address their most critical issues. Examples of commercial solutions Cycorp has delivered in the health, energy, and financial services industries include:

- **Global Energy Supermajor**: Monitoring and advising the company's complex real time oil production processes and alerting engineers when specific conditions of interest (COI) were identified. The Cyc-enabled monitoring and alerting application yielded fewer false positive alarms and fewer false negative missed COIs. For each alert, Cyc generated the pro and con arguments for that hypothesis, and identified the possible remediations and consequences, and produced real time contextual English explanations of each hypothesis, each remediation choice, etc.

- **National Healthcare Provider**: Advising on hospital staffing. This application modeled subject matter experts' understanding of staffing levels in order to understand future needs in each unit of the hospital, thereby mitigating risk, providing savings opportunities, and improving outcomes. It powered a dashboard view of what was happening at the hospital and an accurate prediction of what would likely be happening in the coming minutes, hours, and (small number of) days, and why.
- **Global Investment Bank**: Monitoring compliance with and reporting on internal policies and external regulations.
- **Cleveland Clinic**: Carrying on a clarification dialogue with clinical researchers to help articulate what cross-database query the doctor actually *meant* to ask, and then asking it. This enabled them to become certified to directly provide data reports to accreditation and ranking organizations, rather than having to continue to use costly and time-consuming middleware services.
- **Global Media & Entertainment Company**: Maintaining persistent user models in order to support extended, months-long online chats with their famous characters (versus the shallow, nearly context-free chat-bots and personal assistants one finds ubiquitously today).
- **Global Enterprise Software Vendor**: Deciding which employees in a large organization should be talking to each other, about what, and why, often *non-obviously*, to help one of them succeed at a particularly difficult assignment.
- **U.S. Department of Defense**: Formulating appropriate scenarios to test complex military systems which are being developed in pieces by scores of DoD contractors in scores of U.S. Congressional districts—systems which, by their very nature, one cannot actually test in the real world.
- **U.S. Department of Defense via DARPA**: Modeling students, in this country's largest school district, to help them better understand math and science by *learning by teaching*: i.e. by tutoring an avatar versus "drill and kill" repetition.[4]
- **Global Pharmaceutical Company**: Generating plausible novel scenarios involving opioid fraud, for one of the largest online pharmacy benefits management companies, and then identifying possible cases of those in prescription data, thereby identifying previously-unflagged bad doctors, bad patients, and complicit local pharmacies.

[4] The user doesn't solve math problems directly—he/she watches and mentors an avatar which Cyc controls. Cyc builds up a symbolic model of what that user knows and understands, then makes the avatar be always just a little more confused than that user. If the user gives it good advice, about what it's doing wrong, Cyc lets it stop making that particular sort of mistake. From the user's point of view, it appears as though he/she just successfully taught the avatar something; this dramatically increases the user's engagement in the whole teaching/learning process.

"Cycorp ***delivered efficiently and effectively***. Their AI platform, Cyc, helped make our vision of personalization and ***expert recommendations a reality***.

The ***step-by-step English explanations*** of the reasoning behind Cyc's recommendations ***enabled users to trust the system and is a real differentiator***. Cyc is not just answering ***What***, it is also answering ***Why***."

–Jeriad Zoghby, Global Personalization Lead, Accenture

8.1 When Should an Enterprise Use Cyc?

If your business issue has one or, even better, more than one of the following properties, then Cyc could be a uniquely good fit for that application. If your enterprise has several applications that have such a "fit" then it may be useful to consider using Cyc as your enterprise-wide *semantic knowledge layer.*

1. **Getting the right answer is not enough, it needs to be transparent and fully auditable**: For some applications, being able to provide a step-by-step explanation may help convince a decision-maker; providing and recording such explicit justifications may even be legally required.

2. **Variable employee expertise and expert employee churn**: An organization has a large number of individuals who perform an important task, but the very best of them are *far* better at it than the average one is. Cyc plays the role of a virtual advisor which then amplifies the effective ability of all the rest, even when the best employees move on from their current role (e.g., due to retirement, promotion, or poaching).

3. **Need for different perspectives**: Sometimes the best employees may disagree or have different decision-making styles. Cyc can model each of those n experts separately, capturing their individual "good judgment" and "tribal knowledge". When a new situation or problem arises, Cyc can simulate all the experts actively talking/arguing/working through that problem with each other and with the user.

4. **Disparate data sources**: In many cases, users need to leverage multiple data sources or systems in order to "do their job". Cyc can logically analyze the situation to determine which data sources are relevant, and then access those databases and web services exactly as the expert human user would. No need to create data lakes.

5. **Human bias (i.e. confirmation bias)**: In most situations, even the best of us is prone to think of *one* plausible explanation of what is going on, and *one* plausible course of action, and then find data to support the explanation. Once that happens, it's very difficult to set that aside and generate the second best alternative explanation, etc. Cyc can help by creating and prioritizing multiple alternative hypotheses/explanations/courses of action, and then presenting these alternatives,

along with pro/con arguments for each. Cyc can also check an employees' work to uncover logical inconsistencies or simple gaps in logical reasoning.[5]

6. **"Telephone game" and iterations create human-intensive long lead times for answers**: Often a user *believes* they know the question they want answered, but only when they see the answers do they realize what additional elements, logic, exclusions, etc. they meant to include.[6]

7. **Slow time to value for new employees**: Cyc can be used to train new hires by dynamically deducing what they do/don't yet understand. Cyc then poses questions or simulated challenges that are appropriate given that model. This can also be used to keep experts at the top of their game (akin to the periodic mandated use of flight simulators for airline pilots.)

8. **Complexity of operational environment**: Insights require an understanding of complex topics such as human interaction, real world environments, market forces, changing regulatory regimes, geopolitics, physics, biology, and so on. Traditional databases, triple stores and knowledge graphs are not robust enough—they are too brittle to represent such rich situations, whereas Cyc was specifically created to address just such complexities.

9. **Lack of data to train statistical AIs or analytics**: There just isn't enough "big data" available to train a neural net based machine learning AI. E.g., recognizing a thankfully-rare catastrophic situation, evaluating a potential new product idea, or analyzing unintended consequences of some proposed change in policy, procedure, or operations. Or the data is too spotty or inconsistent to use these statistical approaches.

9 Natural Language Understanding: The Next Frontier

One element *not* listed above is true natural language understanding (NLU). Despite all the hype, the sad fact is that there is no AI today which can automatically read and understand unrestricted text! All that most best-of-breed Natural Language Processing applications can do—and for many applications this is sufficient—is to recognize entities, topics, simple relationships, sentiment, etc. This type of output is often utilized by Cyc in client deployments, but is not true NLU. The bottleneck of applications attempting to work with natural language is that they lack the vast amount of *pragmatics* knowledge we all have, and assume all our listeners/readers have; without it, they can't "decode" pronouns, ambiguous words, ellipses, metaphors, etc. that everyone uses in practically

[5] Combining (5) and (6), Cyc can help untangle situations where multiple sources use the same terms but with different meanings. Or vice versa—e.g., an Amgen researcher used the term *Epogen*, which Cyc knew meant *Epoetin alfa*—so Cyc searched instead for *Procrit* and *Eprex* in any documents authored by Johnson & Johnson.

[6] In one such application at the Cleveland Clinic, for clinical researchers, Cyc carried out a short back and forth interactive query articulation dialogue with the user, which cut the mean time from initial query to an answer deemed acceptable from two weeks (due to the layers of nurses, DBAs, emails, etc. involved) to 2–5 minutes.

every sentence we utter or write. E.g., what are the referents of "he" and "his" in the sentence "*A was mad at B* ***so*** *he stole his pen*" versus the sentence "*A was mad at B* ***because*** *he stole his pen*". Cyc so happens to be exactly what is needed to break that NLU bottleneck, and there already are some successful *stylized* NLU applications powered by Cyc. It is a longer path to full NLU, but our internal Cycorp team is beating its milestones on the roadmap to this goal. Of course *inserting* pronouns etc. into generated text is much easier than decoding them, which is why Cyc is able to do so well at NLG already.

The knowledge represented in Cyc is language-independent, not English-specific. Both directions of natural language processing, NLU and NLG, require some sort of grammar and lexicon that maps words (and idioms) in a natural language, say English, into their various denotations—i.e., what they mean in Cyc. Replacing the English↔Cyc lexicon and grammar rules with, e.g., a German↔Cyc version enables Cyc to produce German paraphrases of anything expressed in CycL.

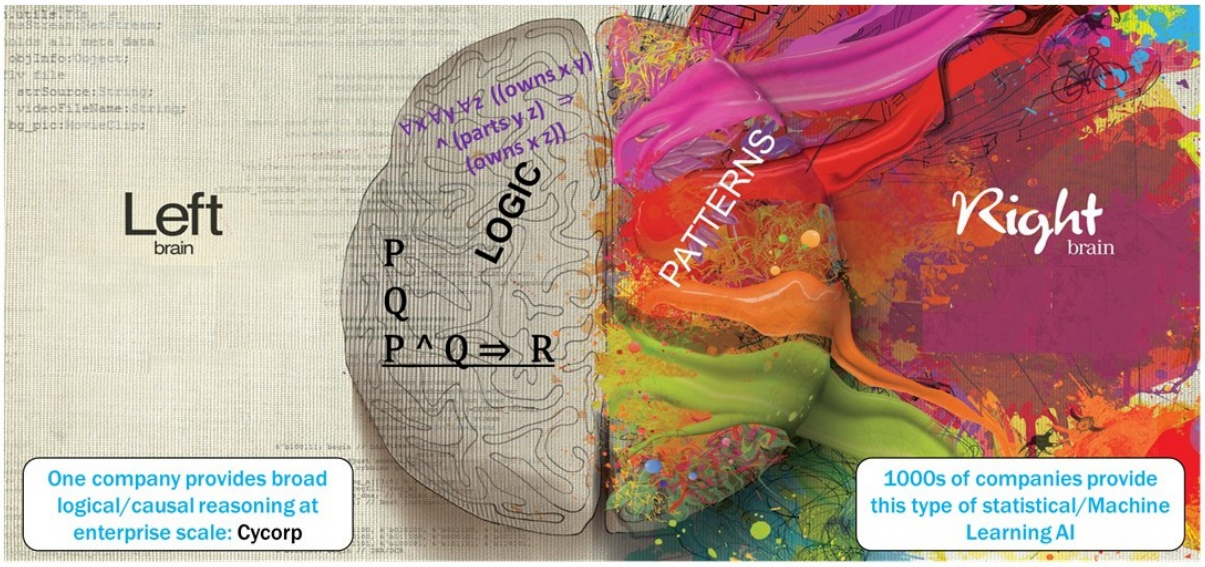
This brings up a related point which is important to make: those lexicons and grammars are language-specific, and comprise only about 2% of the content of the Cyc KB today; the other 98% is language-independent—it has nothing to do with English or any other natural language. The English→Cyc lexicon maps the word "pen" to the Cyc internal ID's (terms) for writing pens, female swans, corrals, penitentiaries, etc. The *knowledge* that writing pens are instruments used to manually write in ink, that writing pens are a few inches long, etc., are relationships and rules one can represent in the CycL language using those internal IDs, not using English words. Cyc can then derive answers to questions, such as whether a writing pen is smaller than a penitentiary, by taking the CycL translation of the question, doing one or more steps of automatic logical deduction on that set of CycL sentences, and Cyc's NLG rules can then translate the result back into, say, English, using the Cyc↔English lexicon and NLG templates.

10 Machine Learning or Cyc? Often the Best Answer is "Both"

One can liken Cyc to left-brain thinking—this is what Daniel Kahneman calls "thinking slow"[7]—whereas the rest of AI today is focused on harnessing right-brain "thinking fast" (training on big data so as to form patterns which can then quickly be recognized in new data).

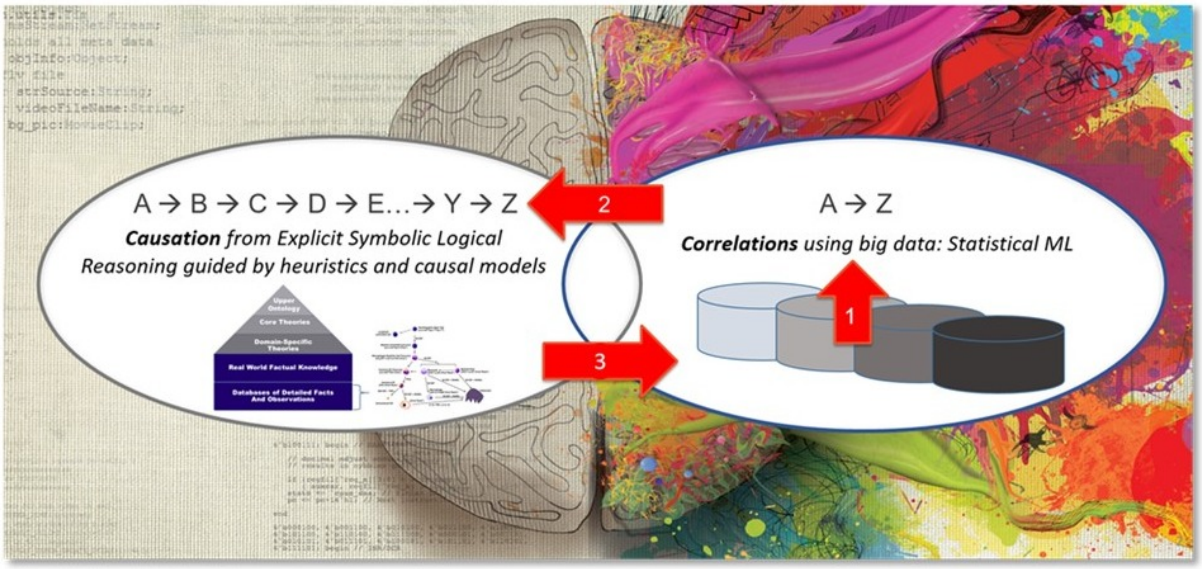
More and more software applications today are based around such "right brain" Machine Learning. Think of ML as one tool in a toolbox: you wouldn't want to build a house *without* using a saw, but neither would you want to have a saw as your *only* available tool. Analogously, Machine Learning based software excels at detecting *correlation*, but not the underlying *causation*.

[7] In his 2011 book *Thinking, Fast and Slow* about left- and right-brain thinking that we all do as human beings.


∀x∀y∀z ((owns x y) ^ (parts y z) → (owns x z))
LOGIC
P
Q
P ^ Q ⇒ R
PATTERNS
Left brain
Right brain
One company provides broad logical/causal reasoning at enterprise scale: Cycorp
1000s of companies provide this type of statistical/Machine Learning AI

For many applications, one or the other of these two paradigms—Cyc or ML—is the best one to apply, and suffices. But for some applications, both types of AI are needed: a synergy between the two is called for. This shouldn't be too surprising—humans clearly have an evolutionary need for both of our brain hemispheres, after all!

Right-brain neural nets and knowledge graphs are good for statistically identifying promising *correlations* (the red circle labeled 1, in the below figure). Left-brain AI (i.e., Cyc) can then think about those correlations in order to formulate possible explanations: causal chains which would account for those regularities (2). The synergy comes full circle if some of those left-brain step-by-step reasoning chains have independently testable predictions (3) which one can then go back to the data to strongly confirm or very strongly disconfirm.

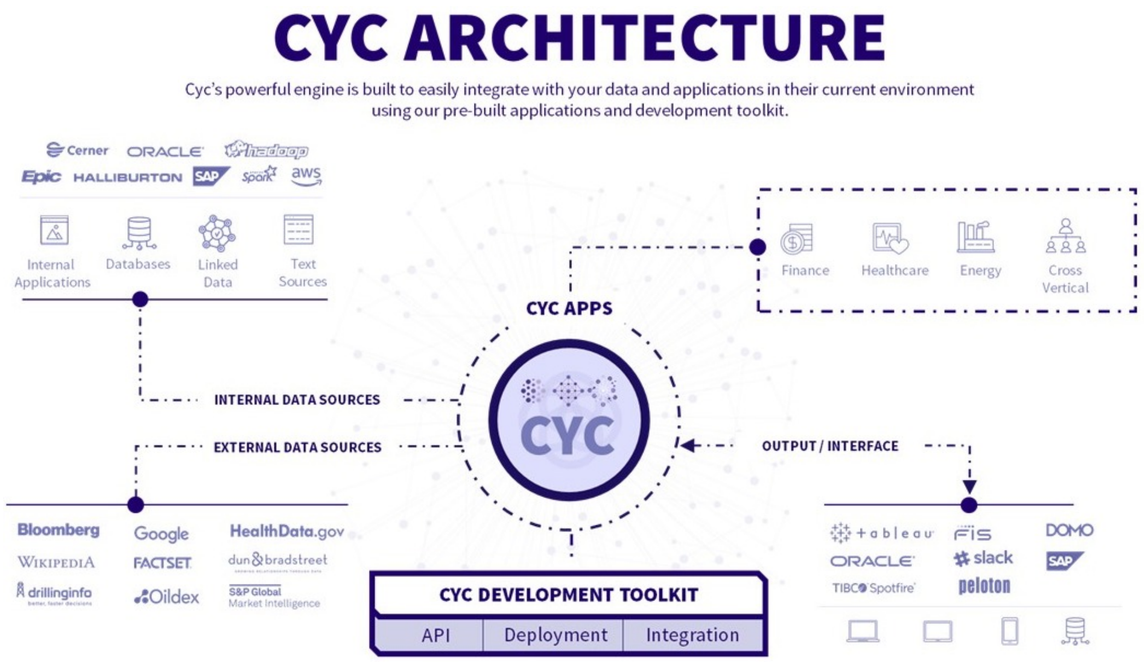

A → B → C → D → E…→ Y → Z
Causation from Explicit Symbolic Logical Reasoning guided by heuristics and causal models
Upper Ontology
Core Theories
Domain-Specific Theories
Real World Factual Knowledge
Databases of Detailed Facts And Observations
A → Z
Correlations using big data: Statistical ML
1
2
3

This approach has been fruitfully applied in several client applications—e.g., with Cleveland Clinic and the National Library of Medicine, to take a statistical correlation from GWASs (genome wide association studies), such as a particular point mutation in a patient's DNA correlating with their having osteoporosis early in life; Cyc then constructed plausible biochemical pathways to explain it ("*this* sNP is next to the gene which if expressed would be *this* protein, which if it were present at site S would catalyze *this* reaction which… (ten steps later) produces elevated levels of bioactive vitamin D in the patient's blood and… (fifteen steps later) impedes bone resorption and leads to osteoporosis"). Going back to the decades-old Framingham data can then confirm this otherwise-insignificant mid-pathway physiological consequence (e.g., elevated levels of bioactive vitamin D).

11 Deploying Cyc

Deploying a Cyc product or developing an application just involves *customizing* the existing Cyc product or platform. Deploying is much like training a new human team member on what data is required to accomplish their job and teaching them the "company way" (i.e. company policies, local laws, etc). New company specific terms may need to be added to its vocabulary (ontology) and new rules and equations may need to be added to its KB. For example, while Cyc already understands how horizontal wells are drilled in tight shale plays, a specific operator's procedures for drilling extended laterals on 4-well pads in the Bakken would be explained to Cyc as part of deploying our Virtual Drilling Advisor product. In some cases, Cyc's existing battery of 1,100 such special-purpose representations and inference engines may be customized as well for efficiency. Finally, the system is flexible for user interface and can deliver output into existing UI and systems via API, or a new interface can be built.

Cyc can be deployed in the cloud or on-site, and will run on an ordinary Windows, Linux, or macOS 64-bit laptop. It needs only 12GB RAM and approximately 20GB of disk space. Its performance is CPU-bound, so the better the processor, the better Cyc's speed (we find an Intel quad-core i7 is more than enough). Cyc installs as a bundle of files, or, if our customers prefer, as a secure containerized standalone package.


CYC ARCHITECTURE
Cyc's powerful engine is built to easily integrate with your data and applications in their current environment using our pre-built applications and development toolkit.
Internal Applications
Databases
Linked Data
Text Sources
Finance
Healthcare
Energy
Cross Vertical
CYC APPS
INTERNAL DATA SOURCES
EXTERNAL DATA SOURCES
CYC
OUTPUT / INTERFACE
CYC DEVELOPMENT TOOLKIT
API
Deployment
Integration