SCE Working Group 1 · Project A · October 2026

Semantic ontologies

Primer for SCE WG1

Matt McKay & Akshay Shanker

Communicating with AI

AI. AI can now write code, solve hard mathematical problems and translate papers into code.

Humans. We need explicit definitions and rules that fix how a model description is interpreted, so that we can:

1. **agree on the meaning** of a paper, a model or a model description;
2. **inspect and verify** what the AI produces;
3. **extend** the set of problems the AI can solve.

Semantics of Semantic Ontologies

Key concepts, definitions and scope

Definitions

Semantics

Semantics. Meaning is a relation between a piece of text and **objects outside language**, the semantic domain.

Missing meaning. A system trained only on text has no way to learn the relation between the text and its meaning (Bender and Koller 2020).

LLMs fill in for missing meaning statistically, from regularities in training text, with no explicit structure that points to a semantic domain.

- A semantic ontology can give an LLM explicit **deterministic** definitions of domain objects and their relations.

Bender and Koller (2020), "Climbing towards NLU", ACL.

Semantic Ontology

Semantics of symbols and text is not enough to understand a system.

Ontology. To understand a statement, we also need the relations, between objects of different kinds.

Semantic ontology. A semantic ontology is a *general* machine-readable account of objects, the kinds they belong to and the valid relations between them.

- Signatures state the types of a relation's arguments and the input and output types of an operation.

Structure of a Semantic Ontology

$$\text{written form} \xrightarrow{\Upsilon} \text{domain of meaning}$$

1. The written form. The written form consists of machine-readable symbols, each with a type and a signature.

- In this deck a type labels a symbol in the written form, whereas a kind sorts the objects in the domain.

2. The domain of meaning. The objects we model, such as state spaces, probability laws, value functions and operators, are sorted into kinds with relations between them.

3. The map Y (upsilon). Explicit interpretation rules specify how typed expressions denote objects of the matching kind.

> **Semantic ontology vs. computational implementation**
>
> Interpreting one written model in a program does not, by itself, specify Y for other models in the domain.
>
> The ontology makes Y explicit through reusable definitions, relations and interpretation rules for its written forms, not execution steps.
>
> Later slides add a second map, P (rho), which attaches a numerical method to the object that Y denotes, so that a model stays apart from how it is solved.

Noy and McGuinness (2001), §1 (reuse and explicit definitions) · Pitts and Winskel, *Denotational Semantics*, §1 (interpretation and execution).

NVIDIA Auto Ontology: customers and payments

Using the ontology below, an AI can retrieve the metric and customer–payment link to write SQL for “Which customers paid most?”

1. Written form	↦ 3. Interpretation	2. Domain of meaning
Type names `Customer`, `Payment`	↦	**Kinds: customer and payment** Objects: $c \in C, p \in P$; P is finite.
Relation field `payment.customer_id`	↦	**Relation (a function)** Signature: $\mathrm{payer} : P \to C$.
Relation field `payment.amount`	↦	**Relation (a function)** Signature: $\mathrm{amount} : P \to \mathbb{R}$.
Relation name `customer lifetime value`	↦	**Relation (a function)** Signature: $V : C \to \mathbb{R}$ (total paid by each customer). $V(c) = \sum_{\substack{p \in P \\ \mathrm{payer}(p) = c}} \mathrm{amount}(p), \qquad c \in C.$

NVIDIA Auto Ontology: customer–payment sample model · AI query generation · FP32 operation semantics (appendix, slide 60).

Substrates for a semantic ontology

NVIDIA Auto Ontology instantiates its semantic ontology as a **graph**, constructed and queried through Python. However, other applications may require different substrates.

REPRESENTATION	HOW THE ONTOLOGY IS EXPRESSED	EXAMPLES
Typed mathematical formalisms	Types, definitions, operation signatures and laws governing the operations.	Math-in-the-Middle (slide 14); mathlib, the library of the proof assistant Lean.
Logic-based ontology languages	Classes, individuals and relations constrained by logical axioms. For example, OWL's `SubClassOf(:Bond :TradableDebtInstrument)` states that every bond is a tradable debt instrument.	FIBO (slide 48) and OntoPlan (slide 23), in OWL; SUMO, in SUO-KIF.
Graph	Typed nodes connected by named relations. RDF expresses links as subject–predicate–object triples.	NVIDIA Auto Ontology (slide 8); OntoPlan's RDF world model (slide 23).
Annotations on models or data	Links from variables or objects to the terms of an ontology.	SemGen's biological models (slide 14); NVIDIA's scene objects (slide 13).

Kohlhase et al. (2017), §§2–3 · Lean/mathlib overview · W3C: OWL 2 overview and RDF graph model; OWL 2 Primer, §§3, 4.1–4.2 (syntax and inference) · Nam et al. (2026), §3.1 · Neal et al. (2019), §2 · NVIDIA, semantic labels. RDF represents statements as subject–predicate–object triples, the predicate naming a relation. OWL axioms and annotations can both be stored as RDF graphs. The OWL snippets use Functional-Style Syntax, with ":" abbreviating FIBO's namespace.

Scope

Possible semantic ontologies for economics

The scope remains open; WG1 could investigate these overlapping candidates.

Data

- Link data columns to quantities such as annual household consumption, with units and timing.
- Specify how data transformations define empirical moments.

Model classes and problems

- Describe model classes and their solution or estimation problems.
- Relate models as special cases or extensions of one another.

Reformulation and estimation

- State when problem reformulations preserve solutions.
- Relate model-implied and empirical moments through a simulated method of moments (SMM) criterion.

The domains can overlap; ontologies for the same domain can use different substrates.

Noy and McGuinness, §3 (scope) · Ma and Stachurski (problem reformulation).

WG1's task

Define the scope of semantic ontologies for computational economics.

1. Choose the **objects and kinds** to describe.
2. State the **relations and operations** between them.
3. Decide which **tasks** the ontologies should support.

Noy and McGuinness (2001), *Ontology Development 101*, §3, Step 1: domain and scope.

Examples of Semantic Ontologies

USE	PUBLISHED FINDING	SOURCE
AI training data (NVIDIA)	NVIDIA's Omniverse simulation platform labels scene objects with identifiers from Wikidata, a public knowledge base, through the semantic-label schema of the OpenUSD scene format. The labels give AI training data its object categories.	NVIDIA specification OpenUSD schema
Robot planning (OntoPlan)	OntoPlan achieved **89% task success**, against **27%** for SayPlan, a comparison system given the same scene facts.	Nam et al. (2026) NeurIPS (accepted)
Robot experience (KnowRob)	The KnowRob knowledge system used an ontology to retrieve grasp poses and forces from recorded robot actions.	Beetz et al. (2018) ICRA
AI program translation	Training a code model to emit symbolic expressions for numeric constants, which a rule-based interpreter converts to IEEE-754 values, raised the share of C functions whose generated assembly passed input–output tests from **3.8% to 92.6% on 21,104 functions**.	Zhang et al. (2024) EMNLP Findings
AI theorem proving (LeanDojo)	Retrieving existing Lean library results raised proof success from **23.2% to 26.3% on 2,000 theorems** requiring results not used in training.	Yang et al. (2023) NeurIPS
Numerical checking (SMT-LIB)	SMT-LIB, a standard input language for automated reasoners, defines floating-point types, operations and rounding modes, so that a reasoner can check a floating-point computation.	Brain et al. (2015) IEEE ARITH

Reported AI gains concern the combined systems described in each paper.

Examples of Semantic Ontologies

USE	PUBLISHED FINDING	SOURCE
Factory planning	Software converted an existing factory ontology and machine-operation descriptions into **267 planning actions in about 15 seconds**.	Malburg et al. (2023) Eng. Appl. AI
Assembly planning (Ford)	Ford's deployed assembly-planning AI used an ontology to interpret vehicle-build instructions and to share that knowledge with other applications.	Rychtyckyj et al. (2016) IAAI
Materials data (MDS-Onto)	MDS-Onto defines terms for X-ray and solar-energy measurements, which software uses to generate data templates and machine-readable datasets.	Rajamohan et al. (2025) Scientific Data
Model composition (SemGen)	SemGen used ontology terms to match variables by biological meaning and combine models in different formats.	Neal et al. (2019) Bioinformatics
Arterial-model assembly	SemGen merged semantically annotated bond-graph modules, whose junction laws conserve mass and energy, into an **86-vessel arterial model from three templates**.	Shahidi et al. (2021) PLOS Comput. Biol.
Biological inference (GO-CAM)	Gene Ontology uses logical rules to infer additional labels for gene functions and distributes them in standard releases.	Thomas et al. (2019) Nature Genetics
Mathematical software	A shared mathematical ontology, Math-in-the-Middle, let three computer-algebra systems, SageMath, GAP and Singular, each perform a step of one polynomial calculation.	Kohlhase et al. (2017) MACIS

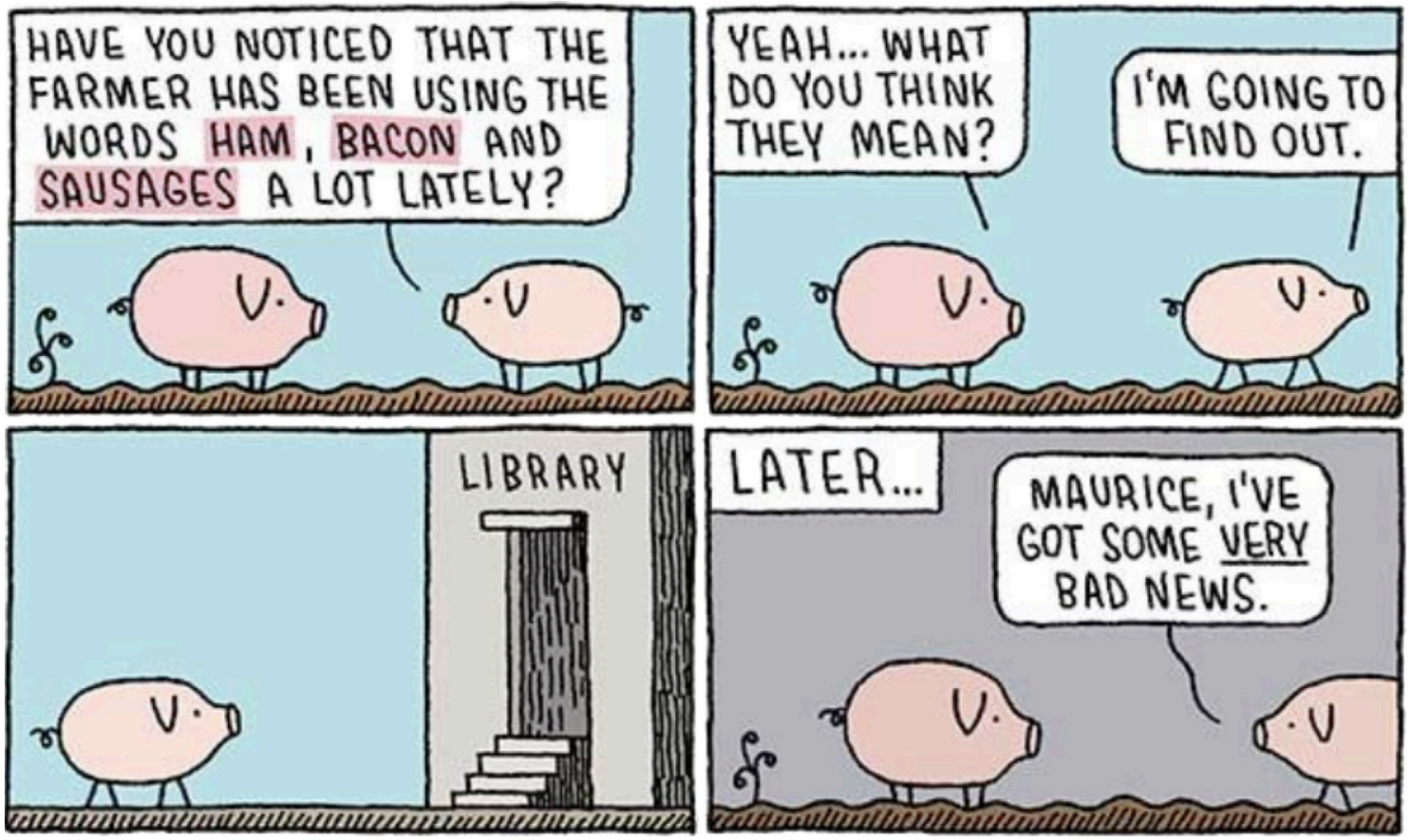
Factory conversion reuses an existing ontology. Arterial-model assembly combines semantic annotations with physical constraints.

Semantic Ontologies in Action

Case studies

Pig goes to the library

What the pig learned in the library



© Tom Gauld, reproduced in Guizzardi and Guarino (2024), "Explanation, semantics, and ontology", *Data & Knowledge Engineering* 153, 102325, Figure 1; shown here for discussion within the working group.

The set-theoretic semantics is not enough

What must a reader know to understand the joke?

Set-theoretic semantics. A function from each word to a subset of a set D of individuals gives the denotations $[\![ham]\!]$, $[\![bacon]\!]$, $[\![sausages]\!] \subseteq D$, the sets of things each word applies to. Such a mapping, Guizzardi and Guarino (2024) say, "would not do Maurice here any good".

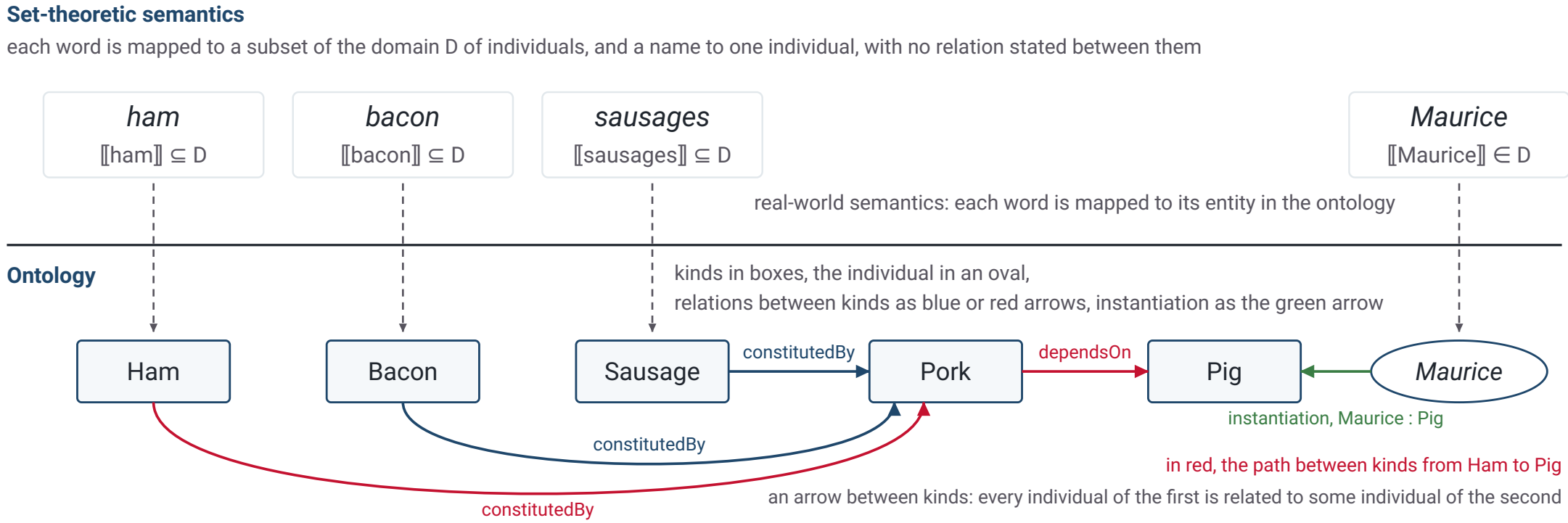
Ontology. The kinds and relations the words are about form a separate system, which the paper states as three facts. Ham, bacon and sausage are food products constituted by pork. Pork comes from a pig. Maurice is a pig.

- In the deck's notation: $Maurice$: Pig, Ham $\sqsubseteq \exists constitutedBy.$Pork (every ham is constituted by some pork), Pork $\sqsubseteq \exists dependsOn.$Pig (every quantity of pork comes from some pig).
- The library holds these three facts, which a dictionary does not.

The joke here arises from the system of kinds and relations, not from the set-theoretic semantics.

Guizzardi and Guarino (2024), "Explanation, semantics, and ontology", *Data & Knowledge Engineering* 153, 102325, §2.1: the quotation, and the three facts (1)–(3) that understanding the cartoon needs, stated there in words; the symbols are the deck's.

When Maurice learns the ontology



Above the line stand the words, the **written form**. Below it stand the kinds, individuals and relations, the **domain of meaning**. The dashed arrows are **Y**, which assigns each word its meaning. The solid arrows state relations between kinds and do not identify Maurice as the source of any particular ham.

Guizzardi and Guarino (2024), §2.1: "Mapping the words 'ham', 'bacon' and 'sausages' to subsets of a semantic domain would not do Maurice here any good"; the three facts (1)–(3); real-world semantics "maps symbols to the entities (types, relations, objects, events, aspects) that are part of this ontology".

Semantics and ontology

Semantics. A set-theoretic semantics maps each word to a subset of a set D of individuals, $[\![ham]\!] \subseteq D$, and nothing more.

Ontology. The ontology is a separate system of kinds, individuals and relations, written as typed symbols.

- Kinds: Pig, Pork, Ham, Bacon, Sausage. The word ham names the kind Ham, whose instances are the individual hams.
- Individual: $Maurice$: Pig. A typed symbol pairs a symbol with the kind of what it names. Here the colon is the instantiation relation, which holds between the individual Maurice and the kind Pig.
- Relations: $constitutedBy$, a relation from Ham to Pork, and $dependsOn$, a relation from Pork to Pig. Each holds between individuals of its two kinds and is not a function from one kind to the other. The kind-level facts Ham $\sqsubseteq \exists constitutedBy.$Pork and Pork $\sqsubseteq \exists dependsOn.$Pig say that every ham is constituted by some pork and that every quantity of pork comes from some pig.

	THE PIGS	THE BUFFER-STOCK MODEL
Written form	ham, $bacon$, $sausages$	m, y, $\mathbb{E}$, v, $\mathbb{T}$ in the Bellman equation
Semantics, $[\![\cdot]\!]$ into a set	$[\![ham]\!]$, $[\![bacon]\!]$, $[\![sausages]\!] \subseteq D$	$[\![\text{Bellman equation}]\!] = \mathbb{T}$, some operator
Left open	foods for pigs, or foods from pigs	which variable $\max$ chooses and which $\mathbb{E}$ averages over, the space of v, the law and timing of y
Ontology, typed statements	$Maurice$: Pig, Ham $\sqsubseteq \exists constitutedBy.$Pork, Pork $\sqsubseteq \exists dependsOn.$Pig	$v : \mathbb{R}_{++} \to \mathbb{R}$, $y : \mathbb{R}_{++}$ with law ν_y, $\mathbb{T} : (\mathbb{R}_{++} \to \mathbb{R}) \to (\mathbb{R}_{++} \to \mathbb{R})$
False agreement	two pigs agree that the words name foods, one meaning foods for pigs and the other foods from pigs	two readers accept the Bellman equation and disagree on the timing of the response to a shock (slide 31)

Guizzardi and Guarino (2024), "Explanation, semantics, and ontology", *Data & Knowledge Engineering* 153, 102325, §2.1 (semantics as a function; an ontology as "a specific theory about the kinds of entity and their ties"; types, individuals and instantiation, where §2.1 writes "the type Pig" for what the deck calls a kind and §3 keeps "kind" for a type whose instances keep their identity; the facts (1) and (2) are stated there in words and written above in the deck's notation), §3.4 (false agreement).

OntoPlan: robot task planning

Motivating example from robotics

Nam et al. (2026), accepted at NeurIPS 2026.

OntoPlan combines a language model with planning software for indoor robot tasks in symbolic simulation. Its ontology defines object types, relations and inference rules, while a matching action model specifies what each action changes.

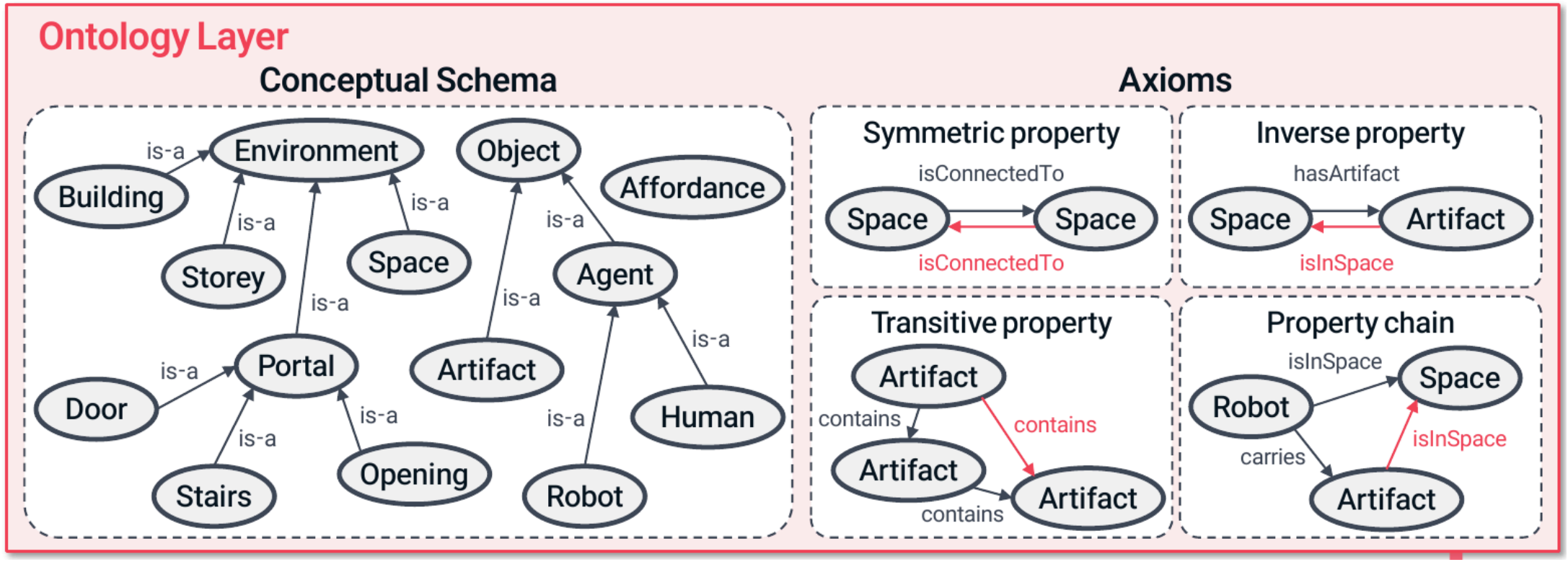
Illustrative task. Move a toothbrush from a cabinet to another room.

MODEL COMPONENT	TOOTHBRUSH EXAMPLE
Objects and types	The toothbrush and the cabinet have type **Artifact** (physical object), while the room has type **Space**.
Relation	The toothbrush is inside the cabinet.
Action effect	After pickup, the toothbrush is held by the robot's hand and is no longer inside the cabinet.
Inference rule	A carried object is in the same room as the robot.

> The language model and the planning software work from the same objects, types and updated relations.

Nam et al. (2026), OntoPlan, §3.1 and Appendix A, Table 7. The task illustrates the paper's state updates.

OntoPlan's ontology defines object types and rules



Written form. The diagram shows type names and logical rules. **Domain of meaning.** The names and rules describe the simulated rooms, robots, objects and relations. **Y.** It interprets `Space` as the set of spaces and `carries` as the relation between a robot and the objects it carries.

Nam et al. (2026), OntoPlan, Figure 2, ontology layer (cropped). The full figure also shows rooms, objects and their states. Y is this deck's notation for interpretation.

Ontology-based planning achieved 89% task success

The same study compared three AI planning systems on 150 simulated robot tasks. All used GPT-4o, the same scene facts and the same 16 primitive actions, and their plans were checked against the same preconditions and effects.

SYSTEM	SUCCESS RATE
OntoPlan	**88.7% (133/150)**
SayPlan	27%
DELTA	13%

OntoPlan retrieves the relevant facts, constructs the plan and records the consequences of each action. DELTA also had formal action rules and plan-constructing software, which means that formal rules alone do not account for OntoPlan's gain.

Nam et al. (2026), OntoPlan, §§4–5 and Appendix E. Results evaluate the ontology together with fact retrieval and planning software.

OntoPlan kept AI inputs small as scenes grew

Within OntoPlan, the authors compared selecting relevant facts with supplying all scene facts as text.

FACTS SUPPLIED TO AI	GROWTH IN THE LARGEST INPUT
Relevant facts selected from the ontology	**1.06×**
All scene facts supplied as text	**8.13×**

Task success was 89% with selected facts and 87% with all facts, a difference that was not statistically significant.

Nam et al. (2026), OntoPlan, Table 6. Ratios compare large with small scenes. Input size counts tokens (text units) in the largest single call per task.

Project Bellman

Algebra for Recursive Dynamic Programming

A mathematical description of a model

This case shows that a machine reading a mathematical description of a model misses part of its meaning. Consider an excerpt from a written statement of the buffer-stock saving model:

> *Each period the household arrives with assets k_t, receives income y_t and earns the gross return $R\,k_t$ on those assets, where R is the gross interest factor. Market resources are then $m_t = R\,k_t + y_t$. Income is i.i.d. with $\log y_t \sim N(\mu_y, \sigma_y^2)$. The household chooses consumption $c_t \in (0, m_t]$, leaving $k_{t+1} = m_t - c_t$, to solve the Bellman equation*
>
> $$v_t(m_t) = \max_{0<c_t\le m_t} \Big\{ u(c_t) + \beta\,\mathbb{E}_t\, v_{t+1}\big(R\,(m_t - c_t) + y_{t+1}\big) \Big\}.$$
>
> *The right-hand side defines an operator $\mathbb{T}$ with $v_t = \mathbb{T}\,v_{t+1}$, which sends the function v_{t+1} to the function v_t, backward from a given terminal value over a finite horizon.*

Context

Implicit context. Economists write code that implements $\mathbb{T}$ from their own understanding of dynamic programming, which the statement leaves implicit.

By itself, machine reading the statement does not give enough information to build $\mathbb{T}$.

- The difficulty is not one of parsing. A machine parses the statement, that is, splits it by grammar rules alone into a parse tree of its symbols, whereas the tree, with no structure added to its symbols, records nothing of the model's meaning. See the appendix on parse trees.
- A semantic ontology makes that implicit understanding explicit.

The missing-meaning problem

The tables on this slide and the next two list some of the meaning that a parse of the buffer-stock statement misses.

MISSING MEANING	IN THE BUFFER-STOCK STATEMENT
Variable bindings	The maximisation binds c_t, while the expectation binds y_{t+1}. The left-hand side binds m_t by defining v_t as a function of m_t, which leaves m_t free inside the right-hand side.
Parameters	The paper defines u, R, β and the law of y_{t+1} elsewhere, in no fixed place. Of these, only β and R are numbers, while u is a function and the law is a probability measure, a difference the statement does not show.
Symbol types	The value function v_t maps $\mathbb{R}_{++}$ to $\mathbb{R}$, with m_t a point of its domain. The tree records both symbols but not that $v_t(m_t)$ evaluates a function at a point of a particular type.
An equation between functions	For each m_t the Bellman equation equates two numbers. Over all m_t it is one equation between functions, $v_t = \mathbb{T}\, v_{t+1}$, in which $\mathbb{T}$ maps v_{t+1} to v_t.

The missing-meaning problem

MISSING MEANING	IN THE BUFFER-STOCK STATEMENT
Whether a symbol is a variable, a function, or a random variable	The symbol m_t, defined by $m_t = R\,k_t + y_t$, is i) the path of one household's resources, ii) the function $(k, y) \mapsto Rk + y$ composed inside the operator, and iii) the random variable whose law is the cross-sectional distribution. A paper writes one symbol for all three meanings, whereas code must represent them as three objects of different types with different uses.
Shock law	The statement fixes $\log y_t \sim N(\mu_y, \sigma_y^2)$. In the shorter form $y \sim \text{LogNormal}(\mu_y, \sigma_y)$, μ_y denotes the mean of $\log y$ under one convention and the mean of y under another.

The LogNormal convention: Project Bellman (the authors' project, slide 33), reference page "Shock distributions".

The missing-meaning problem

MISSING MEANING	IN THE BUFFER-STOCK STATEMENT
Implications of timing stated in words	In a model with entry, beyond the excerpt of slide 27, the sentence "A policy shock raises entrants at date t" is satisfied by two different models, because the shock may reach the cross-sectional distribution at t or at $t+1$, while nothing in the paper records which one the code computes.
What the household knows when it decides	If the return R_t is random, persistent and drawn before the decision, the household's expectation of R_{t+1} depends on whether it observes R_t or only $m_t = R_t\, k_t + y_t$ when it decides.

Multiple possible parsed meanings

Several readings. One well-formed statement admits two or more readings of its parse tree. The appendix gives examples of false agreement, in which two readers accept the same statement and compute different models.

Human readers. False agreement rarely arises when an economist reads the model. An economist has been taught that m_t in $v(m_t)$ is the point at which v is evaluated, not a path of assets across a population.

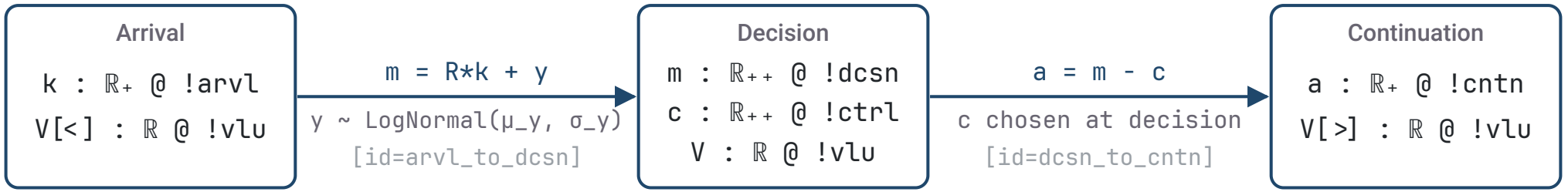
Timing and information. Even so, the timing of events and what the household knows at each decision confuse economists themselves.

AI readers. When we rely on an AI's statistical reading of the text, we cannot be sure that it has read the model correctly, even when its reading appears correct most of the time.

The ontology: one stage of the buffer-stock model

The stage file, which writes one stage (the decision problem of one period) in Bellman-SYM, the model language of Project Bellman (the authors' project), is the **written form**. Its **domain of meaning** contains state spaces, probability laws, value functions and operators. **Y** maps each declaration and block to the mathematical object it denotes.

Written form, the typed stage file



Perches. A perch is one of the three information sets of the stage: arrival `!arvl`, decision `!dcsn` and continuation `!cntn`.

Fields. A field is a state or a value declared at a perch together with its set, as in `m : ℝ₊₊ @ !dcsn`. The role tags `!ctrl` and `!vlu` mark a chosen variable and a value.

Blocks. A block holds the equations between two perches: the transition with the shock, whose law line `y ~ LogNormal(μ_y, σ_y)` declares its distribution, the choice in the `[id=policy]` block, and the expectation.

Bellman-SYM, Stages §§1–4: the documentation example, whose b and w are written here as the calculus's k and m (Bellman calculus, Stages and perches §1, bellman-ddsl/docs) · the law line fixes the convention that slide 30 left to a subfield · Bellman-SYM and bellman-ddsl are internal Project Bellman documentation, not yet public.

What the ontology checks and makes explicit

Under Y (upsilon), each field and block of the file denotes a typed object. The role tag `!vlu` marks a value. The marks `[>]` and `[<]` give the continuation and arrival values $v_{\succ}$ and $v_{\prec}$, whose spaces are $\mathsf{V}_{\succ}$ and $\mathsf{V}_{\prec}$, beside V at the decision perch. The elaborator, the program that constructs those objects, refuses a line that the types do not admit and names the rule.

LINE IN THE FILE	UNDER Y IT DENOTES
`V[>] : ℝ @ !vlu` at the continuation perch	$v_{\succ} : \mathbb{R}_{+} \to \mathbb{R}$, a function of savings, never a number
the choice block `[id=cntn_to_dcsn]` with its `[id=policy]` block	$\mathbb{T}_{\succ} : \mathsf{V}_{\succ} \to \mathsf{V}$, the choice operator
the expectation block `[id=dcsn_to_arvl]`	$\mathbb{T}_{\prec} : \mathsf{V} \to \mathsf{V}_{\prec}$, the expectation operator

A LINE THAT	THE ELABORATOR
adds the shock outside its own expectation, `V[<] = E_{y}(V) + y`	refuses it under `shock-reintroduced`, since y is not known at arrival
lists a continuation value at the decision perch	refuses it under `field-perch`, since the value is a function of savings
puts the mean of the shock in the transition, `m = R*k + exp(μ_y + σ_y^2/2)`	accepts it, leaving the different model visible on that line
writes a second `β` in the expectation block	accepts it, leaving the second `β` visible on its own line
would impose a stationary closure $v_{t+1} = v_t$	has no line for it, since `V[>]` comes from the next period or the terminal value

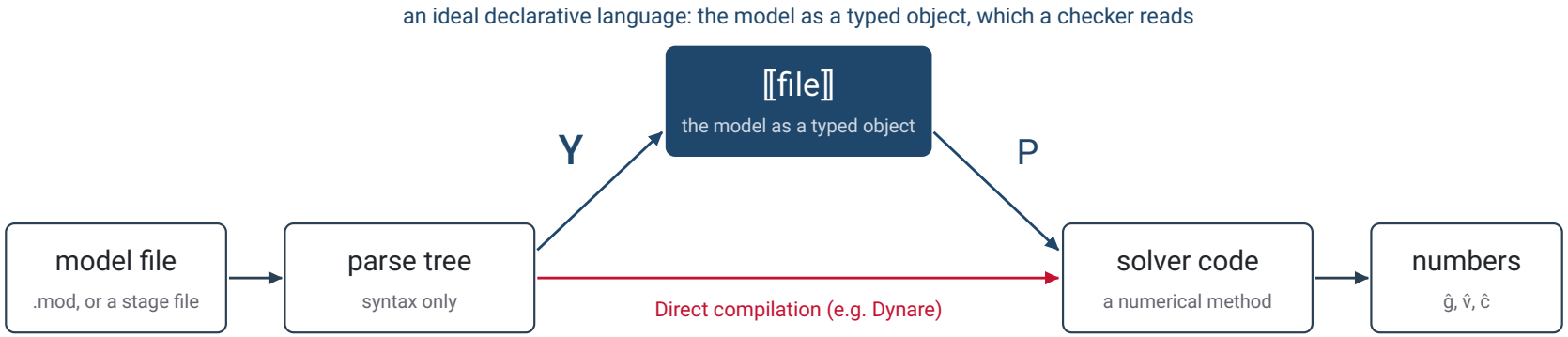
The reference on refusals, rules shock-reintroduced and field-perch; Bellman-SYM, Field equations §5, Perch tagging rules §3, Periods §§2–3, Trellises §2; the internal runs of `elaborate_text` of 13 October 2026.

Current modelling tools and AI

Dynare and other tools already turn model descriptions into computed results.

What Dynare and other modelling tools do

Dynare and Dolo read a model description and apply a numerical method to it, as a program in R or Stata calls a numerical routine. In the figure, the lower route is this computation, carried out in the tool's own language and modelling rules.



With a shared economic interpretation. The upper route states the interpretation separately. Y maps the written model to a typed model object, ⟦file⟧, to which P (rho) then attaches a numerical method.

Conceptual routes in this deck's notation: Y denotes a separately stated, shared economic interpretation. Compilers can already perform semantic checks. Dynare represents equations separately from solution commands. Clang documentation · Dynare manual · Dolo documentation · Bellman calculus, Introduction §1.

What AI needs beyond code compilation

Successful compilation or execution does not establish agreement with the intended economic model.

Shared definitions. Give the AI an ontology that specifies the economic objects, their units, timing and relations.

Links to code. Identify the variables and functions that implement each shared economic definition.

Checks on model agreement. Check declarations against the ontology, review the links to code, and test the numerical results.

> Example. Shared definitions distinguish assets chosen today from resources after tomorrow's income shock, even though both are stored as floating-point numbers.

Y interprets the declared symbols as economic objects. Reviewing their implementation is a separate task. Proposed requirements for AI model generation and translation. Precedent for shared definitions and mappings: Kohlhase et al. (2017), MACIS, §§2–5.

The community library

One model, several numerical implementations

QuantEcon's community library asks each toolkit to implement a shared model and calibration with its own solution method, in a runnable tutorial.

PART OF A CONTRIBUTION	BUFFER-STOCK EXAMPLE
Common economics	Equations, timing, shock laws, normalisation and calibration.
Implementation choices	Solution method, shock approximation, grids, interpolation and software version.
Comparison quantities	Consumption, target resources, marginal propensities to consume and the seven model conditions listed in the submission guide.

> The starting point is the buffer-stock description and the tutorial in the HARK toolkit, submitted as pull request #5 (a proposed addition to the library's repository), open with changes requested.

Community Library · Buffer-stock submission guide. The result tables, metadata format and tolerances are optional draft conventions. Reporting requirements remain under discussion. Status checked 15 October 2026.

Link toolkit variables to shared model definitions

PUBLISHED PRECEDENT	WHAT WAS DONE
SemGen	Ontology annotations matched biological quantities across different model formats.
Math-in-the-Middle	Symbols from SageMath, GAP and Singular were mapped to shared mathematical definitions.
BioModels, a curated database of biological models	Curators annotated model entities and reproduced published simulations.

Proposed for the library. Link each implementation's variables and operations to agreed economic definitions. The links align the written terms. Interpreting those terms as economic objects is what **Y** does.

Neal et al. (2019), *Bioinformatics*, §§2–5 · Kohlhase et al. (2017), MACIS, §§2–5 · Malik-Sheriff et al. (2020), *Nucleic Acids Research*. The library extension is proposed; the papers establish the precedents.

Two estimates of one defined quantity

In the buffer-stock description, **target resources** are the level of resources, measured relative to permanent income, at which expected next-period resources equal current resources.

POINTS IN THE INCOME-SHOCK APPROXIMATION, BESIDES THE ZERO-INCOME POINT	SAVED HARK ESTIMATE OF TARGET RESOURCES
7	**1.3910**
21	**1.3994**

The same model and calibration give different numerical estimates when the shock approximation changes.

> Proposed extension. Each result would declare its model version, quantity definition, normalisation and numerical settings. Reviewers would check these declarations against the code and numerical results.

Saved HARK sensitivity study, §3 · Quantity definition, §1. Both calculations retain the same additional zero-income outcome and asset grid. These are saved estimates, not a new computation.

Use the library to test WG1’s common definitions

Proposed pilot. Start with the full buffer-stock description and the HARK contribution before adding a second toolkit.

Task. Ask an AI to identify comparable results and report differences in numerical settings.

Comparison. Compare the AI’s answers with and without semantic annotations, using the same complete model statement, equations, calibration and run information. Record correct matches, incorrect matches and cases requiring review.

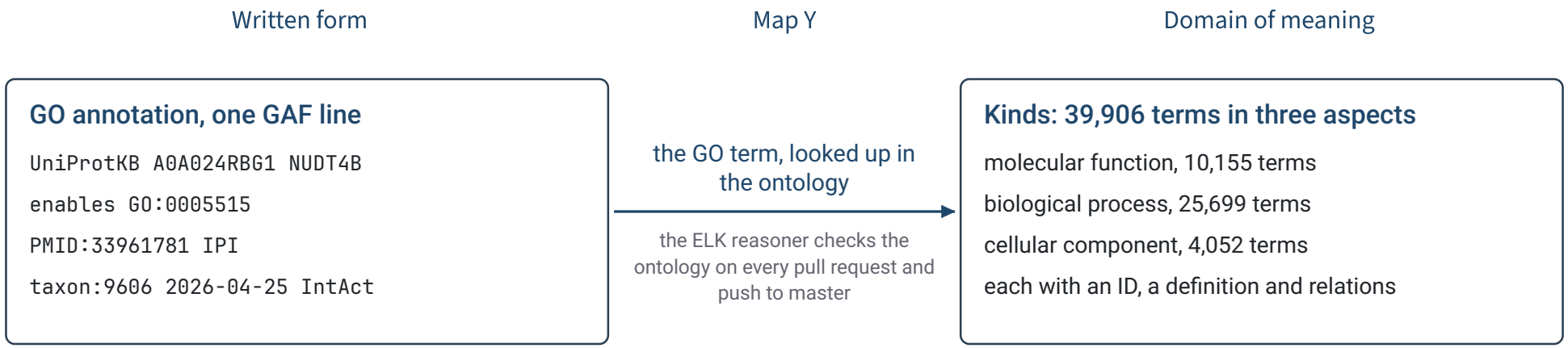
> The library supplies the worked examples from which WG1 can decide which definitions and checks to share.

Proposed pilot, not an existing library feature. The annotations restate supplied facts. The full buffer-stock model includes permanent and transitory shocks. The earlier Bellman stage illustration is simplified.

Appendix

Further examples

Biology: the Gene Ontology

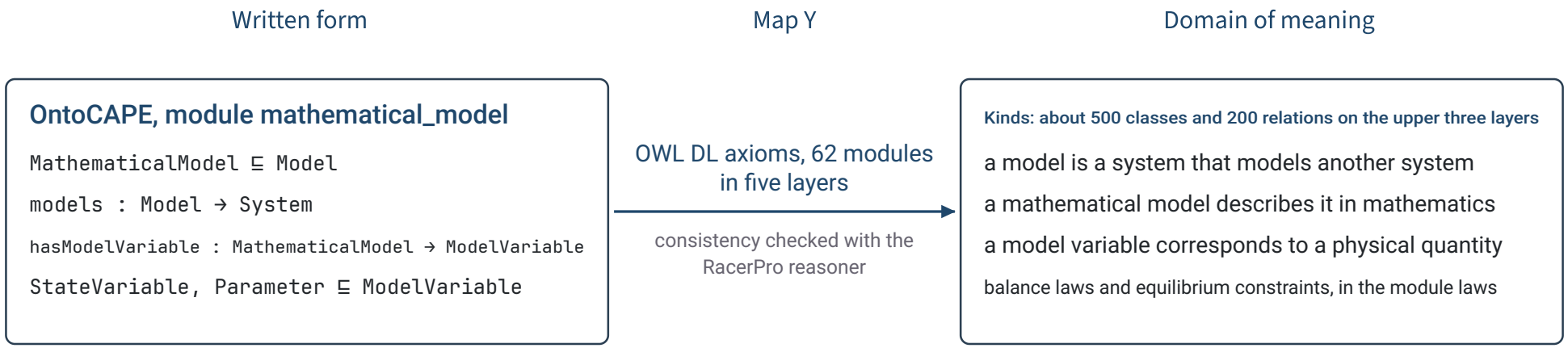


Excludes. The ontology excludes three misreadings. A missing annotation may not be read as the absence of a function, under the open-world assumption, which treats an absent statement as unknown rather than false. A NOT qualifier, which records that a gene lacks a function, may not be read as an inconclusive result. A term may not be read by its name rather than by its definition and relations, since a label must be unique and a definition must always denote the same referents (principles 12 and 19 of the OBO Foundry, the body that sets the rules for biomedical ontologies).

For economics. A model's object is named by a term with a definition and relations, not by a variable name. The role tags `!ctrl` and `!vlu` and the law line of a stage file play that part, as the `sboTerm` of the Systems Biology Ontology does for a rate law in SBML, the exchange format for biochemical models.

Ashburner et al. (2000), "Gene Ontology: tool for the unification of biology", *Nature Genetics* 25(1), 25–29 · The Gene Ontology Consortium (2026), "The Gene Ontology knowledgebase in 2026", *Nucleic Acids Research* 54(D1), D1779–D1792, Table 3 (the release of 22 July 2025) · GAF 2.2 documentation and the human annotation file of 21 May 2026 (geneontology.org); the go-ontology build file, which runs the ELK reasoner · SBO: Courtot et al. (2011), *Molecular Systems Biology* 7, 543.

Engineering: OntoCAPE, the ontology for chemical process engineering



Excludes. The ontology excludes a model confused with the system it models and a model variable confused with the physical quantity it stands for. The relation `models` joins a model to its system, while `correspondsToQuantity` pairs a variable with a quantity, as a declared domain excludes the deck's function on the wrong space (slide 55).

For economics. OntoCAPE is the nearest case to Project A because it is an ontology of models themselves. Model variables, equation systems and solution strategies are kinds in its modules (its sub-ontologies), so that a process model and its solver are distinct typed objects, as Y and P keep a model apart from its numerical method.

Morbach, Yang and Marquardt (2007), "OntoCAPE—A large-scale ontology for chemical process engineering", *Engineering Applications of Artificial Intelligence* 20(2), 147–161 · Morbach, Wiesner and Marquardt (2009), *Computers & Chemical Engineering* 33(10), 1546–1556 · Marquardt, Morbach, Wiesner and Yang (2010), *OntoCAPE: A Re-Usable Ontology for Chemical Process Engineering*, Springer · the OntoCAPE 2.0 distribution (AVT, RWTH Aachen, GPL): system.owl and mathematical_model.owl; technical reports LPT-2008-25 and LPT-2008-28; Morbach (2009), *A Reusable Ontology for Computer-Aided Process Engineering*, dissertation, RWTH Aachen; the AVT page for the RacerPro check.

Earth science: the SWEET ontologies

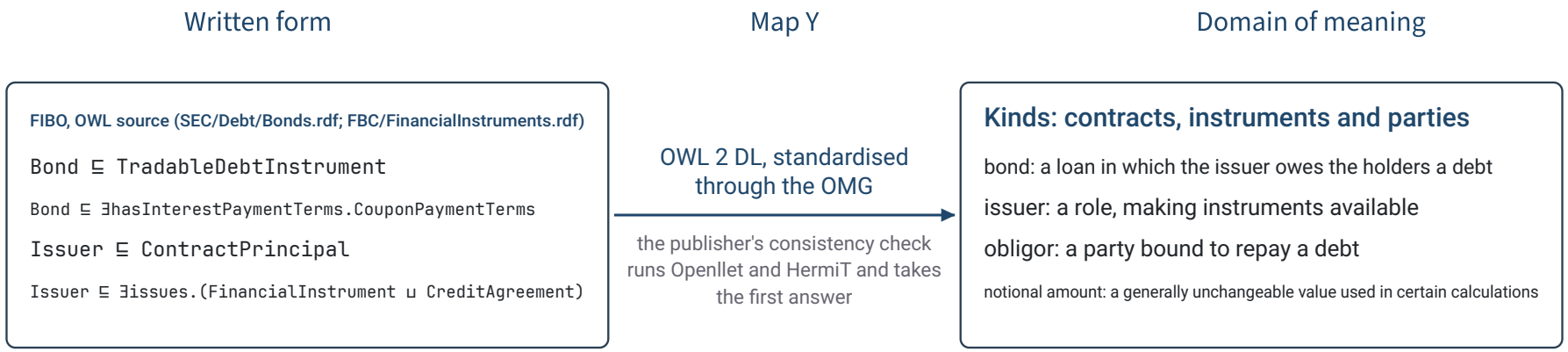


Excludes. The ontology excludes a quantity name read as any of several kinds, since a SWEET class fixes the kind by its definition and by its place among realms (SWEET's divisions of the Earth system), quantities and phenomena. The Climate and Forecast (CF) standard names, published as SKOS concepts (the W3C's format for agreed lists of terms) on the NERC vocabulary server, each linked to its canonical unit, settle the same question for a variable such as `temp` in a netCDF data file (a format for climate data). In the deck's terms this is the question of the shock law (slide 30) and of timing stated in words (slide 31).

For economics. The comparison paper of the special issue that Project A proposes needs a vocabulary of this kind: which object each team reports, a value or a marginal value function, on which state and under which timing, as in the comparison projects that den Haan, Judd and Juillard (2010, 2011) introduced.

Raskin and Pan (2005), "Knowledge representation in the semantic web for Earth and environmental terminology (SWEET)", *Computers & Geosciences* 31(9), 1119–1125 · ESIPFed/sweet, release 3.5.0 (14 July 2022), the header of sweetAll.ttl; the class count from the 3.5.0 source · CF standard names: NERC Vocabulary Server, collection P07, version 95 (16 September 2026), publisher Natural Environment Research Council, owner the CF Standard Names Committee · den Haan, Judd and Juillard (2010, 2011), *Journal of Economic Dynamics and Control* 34(1), 1–3 and 35(2), 175–177.

Finance: FIBO, the Financial Industry Business Ontology

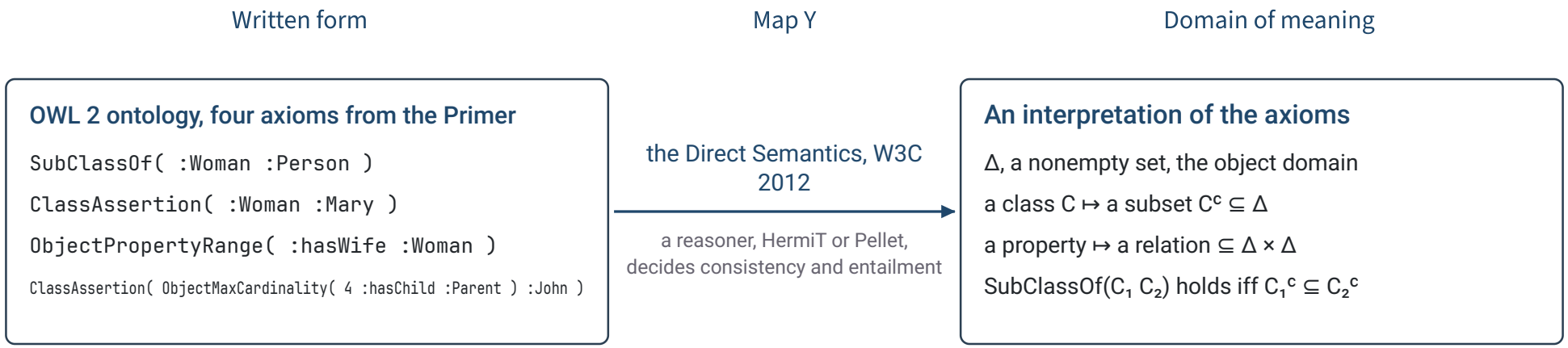


Excludes. The ontology excludes an issuer read as the obligor (the party that owes the payments) of a bond and a notional amount read as a principal, since each is a defined class or property (OWL's word for a relation). The 2023 revision of the Swaps ontology separated the nominal from the notional amount, which its change note says "were confused". In the deck's terms these are a function on the wrong space (slide 55) and the shock law (slide 30).

For economics. FIBO is finance's own ontology, with the Bureau of Economic Analysis, the Bureau of Labor Statistics, the CFTC and the Office of Financial Research among its contributors. The 2026 rule under the Financial Data Transparency Act records it among the standards that commenters proposed. Project A's ontology names the objects of a model as FIBO names the parties and terms of a contract.

EDM Council, FIBO (spec.edmcouncil.org/fibo), release Q2 2026 (14 July 2026), MIT licence; the development-process page on consistency; onto-viewer's consistency checker (Openllet and HermiT) · OMG specifications EDMC-FIBO/FND 1.2, BE 1.1, FBC 1.0 and IND 1.0 (2017–2018) · Bennett (2013), "The financial industry business ontology: best practice for big data", *Journal of Banking Regulation* 14(3–4), 255–268 · Allemang, Garbacz, Grądzki, Kendall and Trypuz (2021), *FOIS 2021*, 112–126 · Financial Data Transparency Act Joint Data Standards, 91 FR 38246 (25 June 2026) · GoodRelations, the ontology schema.org integrated in 2012: Hepp (2008), *EKAW 2008*, LNCS 5268, 329–346.

Knowledge representation: OWL 2, the language ontologies are written in



Excludes. The language excludes any reading of the axioms other than the one fixed by the Direct Semantics, OWL's official rule for what its statements mean, under which "the correct answer to any such question is predetermined by the formal semantics", whatever reasoner computes it (Primer §2). It also does not read a missing statement as false, and it does not assume that two names denote two individuals.

For economics. This is the three-part definition in full: classes, properties and individuals as typed symbols, an interpretation as the domain, and a map whose consistency (that the axioms do not contradict one another) and entailments (what follows from them) a reasoner decides. A stage file's Y plays the part of the Direct Semantics, and the elaborator the part of the reasoner.

W3C (2012), *OWL 2 Web Ontology Language*, Recommendations of 11 December 2012: Document Overview §2.3; Direct Semantics §§2.2–2.5; Profiles §§1 and 5; Primer §§2, 4, 5 and 9 · Baader, Calvanese, McGuinness, Nardi and Patel-Schneider (eds.) (2007), *The Description Logic Handbook*, 2nd ed., Cambridge University Press · HermiT: Glimm, Horrocks, Motik, Stoilos and Wang (2014), *Journal of Automated Reasoning* 53(3), 245–269; Pellet: Sirin, Parsia, Grau, Kalyanpur and Katz (2007), "Pellet: a practical OWL-DL reasoner", *Journal of Web Semantics* 5(2), 51–53.

The five reference ontologies against the three parts of a semantic ontology

Each row places one ontology against the three parts of slide 7, the typed symbols of the written form, the domain of meaning and the map between them, and names in the last column the program that checks the map.

ONTOLOGY	WRITTEN FORM	DOMAIN OF MEANING	MAP Y	CHECKER
Gene Ontology	the GO terms, each by ID	kinds of molecular function, biological process and cellular component	the term in a GAF annotation	the ELK reasoner, on every pull request and push to master
OntoCAPE	OWL classes and relations in 62 modules	models, systems, materials and quantities	the axioms, `models` and `hasModelVariable`	the RacerPro reasoner
SWEET and the CF names	OWL classes, and SKOS concepts for the CF names	Earth-science quantities, realms and phenomena	the class, or the standard name on a variable	any OWL tool; the CF checker against the standard-name table at cfconventions.org
FIBO	OWL classes and properties	contracts, instruments and parties	the class asserted of a record	a consistency check by Openllet or HermiT, whichever answers first
OWL 2	classes, properties and individuals	an interpretation: a set with relations on it	the Direct Semantics	a reasoner: consistency and entailment

Harel and Rumpe (2004), *Computer* 37(10), 64–72, for the three parts · the sources of each ontology are on its slide: Ashburner et al. (2000) and the Gene Ontology Consortium (2026); Morbach, Yang and Marquardt (2007); Raskin and Pan (2005) and the NERC Vocabulary Server; the EDM Council and Bennett (2013); W3C (2012).

The stage file against the same three parts

The stage file of slide 33, the deck's own written form, is read here against the columns of the table on slide 50.

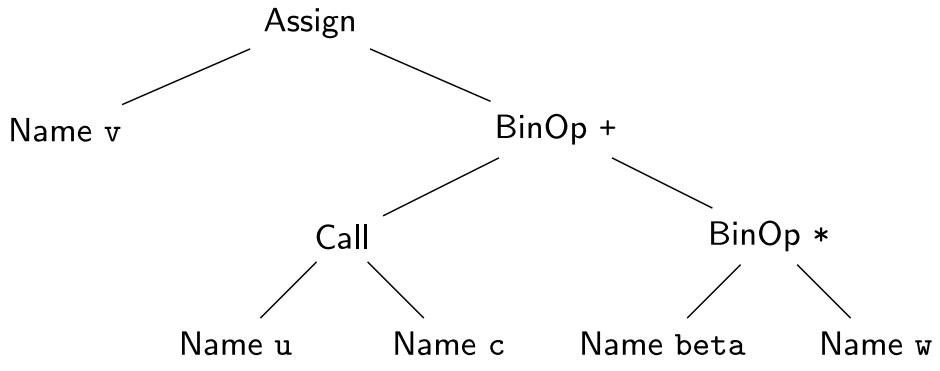
PART	IN THE BELLMAN-SYM STAGE FILE
Written form	the fields and the blocks, each with a type and a perch, and the declared law of the shock (slide 33)
Domain of meaning	state spaces, laws, value functions and the operators between them (slide 7)
Map Y	the perch tags `@ !arvl` , `@ !dcsn` and `@ !cntn` , the role tags `!ctrl` and `!vlu` and the law line (slide 33), interpreted by Y (upsilon), the checked interpretation of slide 53, as the objects of slide 7
Checker	the elaborator (slide 34), which constructs the typed object or refuses the file, naming the rule the file breaks and the line; and the comparator, a program that compares two elaborated files up to renaming and reports, for each kind of item, agreement, difference or an undecided residue, the items it can neither match nor reject
Run on the file of slide 33	The elaborator constructed the object from the file and refused the line `E_{y}(V) + y` , with `y` listed among the block's inputs, under `shock-reintroduced` . The comparator read the file and the documentation example, with b and w for k and m, as one object under that renaming.

Bellman-SYM, Stages §5 (checked interpretation) · bellman-ddsl wiki (internal), Y §1 (the elaborator) and judge §4 (the comparator) · the internal runs: `elaborate_text` of `bellman.factory.elaboration` and `compare` of `meaning_map_training.compare` , 13 October 2026, on the stage file of slide 33 and on Bellman-SYM's documentation example (Stages §4).

Return to slide 28

The parse tree of a written statement

Parsing a written statement into a tree, as every compiler first does, shows what the statement leaves to context.



What the parser does. It splits `v = u(c) + beta * w` into a tree by grammar rules alone.

What the tree records. The tree records only which symbol is applied to which, node by node.

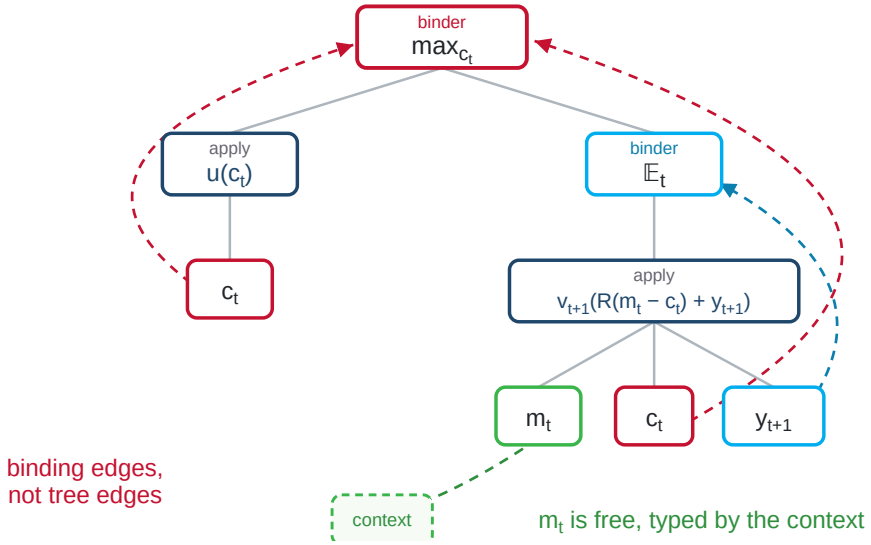
What the tree omits. The tree records nothing about what `v`, `c` or `w` are, which variable is bound where, or whether `v` is a function.

The tree is Python's abstract syntax tree, `ast.parse("v = u(c) + beta * w")`.

Return to slide 28

From the parse tree to the semantic graph

In the model, v is not a number obtained by adding βw to $u(c)$. That $v_t = \mathbb{T}\, v_{t+1}$ sends a function to a function appears only once the binders, types and declarations on the right are attached to the tree by the dotted lines in the figure, which turn the tree into the semantic graph.



binders: $\max_{c_t}$ binds c_t, $\mathbb{E}_t$ binds y_{t+1},
$v_t(m_t) = \cdots$ $\forall m_t \in \mathbb{R}_{++}$ binds m_t

fixed from outside: $u : \mathbb{R}_{++} \to \mathbb{R}$, β, $R \in \mathbb{R}$,
ν_y the law of y_{t+1}, on $\mathbb{R}_{++}$

obtained: $v_{t+1}\big(R(m_t - c_t) + y_{t+1}\big) : \mathbb{R}$,
$\mathbb{T} : (\mathbb{R}_{++} \to \mathbb{R}) \to (\mathbb{R}_{++} \to \mathbb{R})$

The law ν_y is the probability distribution of the income shock. A semantic ontology states the quantifier over m_t on the left-hand side, the binder conventions, the types and the declarations of u and ν_y, none of which the tree records.

Bellman-SYM, Project Bellman's model language, parses a file into such a tree, builds the semantic graph from it by a checked interpretation, which the calculus calls Y (upsilon), and discards the tree (Bellman calculus, Introduction §§1–2; Bellman-SYM, Stages §5).

Return to slide 32

What the expectation integrates

The paper's Bellman equation gives $\mathbb{E}$ the date t as its subscript rather than the variable it integrates. Because the tree specifies neither that variable nor the type of $\mathbb{E}_t$, each box below puts the paper's expression to the left of "read as" and a reading that the tree does not exclude to the right.

The shock leaves the expectation.

$$\mathbb{E}_t\, v\big(R(m-c)+y'\big) \qquad \text{read as} \qquad v\big(R(m-c)+\mathbb{E}_t\, y'\big) \quad \text{or} \quad \mathbb{E}_t\big[v\big(R(m-c)\big)\big]+y'.$$

Since Jensen's inequality gives $\mathbb{E}_t\, v\big(R(m-c)+y'\big) \leq v\big(R(m-c)+\mathbb{E}_t\, y'\big)$ for concave v, the first reading overstates the continuation value. Because it replaces the income risk by its mean, it also removes the precautionary motive, the saving held against that risk (Kimball 1990). The second reading treats y' as known before it is drawn.

The expectation moved through the maximisation.

$$\max_c \big\{u(c)+\beta\, \mathbb{E}_t\, v_{t+1}(\cdot)\big\} \qquad \text{read as} \qquad \mathbb{E}_t \max_c \big\{u(c)+\beta\, v_{t+1}(\cdot)\big\}.$$

Since $\mathbb{E}\max \geq \max\mathbb{E}$, the reading on the right overstates the value and lets consumption respond to income that has not yet been drawn.

The precautionary motive: Kimball (1990), "Precautionary saving in the small and in the large", *Econometrica* 58(1), 53–73. Jensen's inequality is a standard result.

Return to slide 32

A number for a function, and a function on the wrong space

A number where a function belongs.

$v_t = \mathbb{T}\, v_{t+1}, \quad v_t : \mathbb{R}_{++} \to \mathbb{R}$ read as $v_t(\hat{m}_t)$ alone, at $\hat{m}_t = R\,\hat{k}_t + \hat{y}_t,$ one simulated household's resources.

The backward step then returns one number and one consumption $c_t(\hat{m}_t)$, although $\mathbb{T}$ needs v_{t+1} at every $R(\hat{m}_t - c) + y'$. Under this reading the reader has simulated a path where the model asked for a function.

A function on the wrong space.

$\mathbb{E}_t\, v_{t+1}(Ra + y')$ on savings $a = m - c,$ the paper's $k_{t+1},$ $a \in [0, \infty)$ read as v_{t+1} itself, on resources $m \in (0, \infty)$.

The borrowing limit acts at $a = 0$, a point at which a function of resources is not defined, so that the two functions have different domains. A reading that confuses them puts the boundary at a point where the constraint does not act.

Return to slide 32

The discount and the expectation applied twice, and the recursion as a fixed point

The discount and the expectation applied by both steps once $\mathbb{T}$ is split.

$$v_t = \max_c \left\{ u(c) + \beta \, \mathbb{E}_t \, v_{t+1}(\cdot) \right\} \qquad \text{read as} \qquad v_t = \max_c \left\{ u(c) + \beta^2 \, \mathbb{E}_{y'} \mathbb{E}_{y''} \, v_{t+1}(\cdot) \right\}.$$

Since the Bellman equation writes β and $\mathbb{E}_t$ once each, an implementation that splits $\mathbb{T}$ into the choice and the expectation must decide which step applies β and which applies $\mathbb{E}_t$. An implementation that applies β and $\mathbb{E}_t$ in each of the two steps makes the household discount at β^2 and draw income twice.

Dated value functions read as one stationary function.

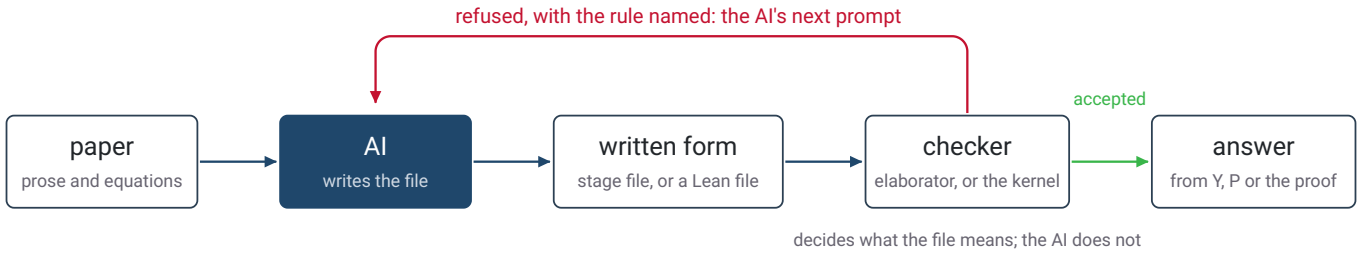
$$v_t = \mathbb{T} \, v_{t+1}, \;\; v_{t+1} \text{ an input} \qquad \text{read as} \qquad v = \mathbb{T} \, v, \;\; \text{a fixed point.}$$

A stationarity condition that the statement never makes replaces the backward recursion from a terminal value. Once the terminal value has disappeared from the problem, the last period's consumption is chosen as if another period followed.

How an AI works with a semantic ontology

Return to slide 6

The AI writes against the ontology



Autoformalization. The AI translates the paper into the written form, which the checker accepts or refuses (Wu et al. 2022, where the checker decided syntactic correctness and people judged faithfulness; AlphaProof, Hubert et al. 2026, which learned by reinforcement with Lean's kernel, the core program that checks each proof step, as the verifier).

Refuse and repair. The checker's refusal, with its rule named, is the AI's next prompt, as the error message of a proof assistant (a program that checks proofs) is in Baldur, a proof-repair system (First et al. 2023). The elaborator's rules `shock-reintroduced` and `field-perch` are messages of this kind.

Constrained generation. The AI's output is restricted, as the AI produces each word of its output, to the grammar of the written form, so that only its meaning remains to be checked (Poesia et al. 2022; Geng et al. 2023).

Compute, then answer. From the accepted file the ontology's programs compute Y's object and P's numbers (or, for a proof, the kernel checks it), so that the AI's answer to the user is built from that output rather than from its own guess.

Round trip. A second program writes the accepted object back into prose for a judge, a program or a second AI, to compare with the paper (Li et al. 2024), or regenerates the object for comparison with the original (Allamanis et al. 2024). Matsya, the project's assistant service, runs the first loop with the elaborator and two judges and the second with its comparator.

Wu, Jiang, Li, Rabe, Staats, Jamnik and Szegedy (2022), "Autoformalization with large language models", *NeurIPS 2022*, 32353–32368 · Hubert, Mehta, Sartran et al. (2026), *Nature* 651, 607–613, published online 12 November 2025 · First, Rabe, Ringer and Brun (2023), "Baldur: whole-proof generation and repair with large language models", *ESEC/FSE 2023*, 1229–1241 · Poesia et al. (2022), "Synchromesh", *ICLR 2022*; Geng, Josifoski, Peyrard and West (2023), "Grammar-constrained decoding for structured NLP tasks without finetuning", *EMNLP 2023*, 10932–10952 · Li, Wu, Li, Wei, Yang, Zhang and Ma (2024), "Autoformalize mathematical statements by symbolic equivalence and semantic consistency", *NeurIPS 2024*, 53598–53625; Allamanis, Panthaplackel and Yin (2024), "Unsupervised evaluation of code LLMs with round-trip correctness", *ICML 2024*, PMLR 235, 1050–1066 · Matsya's architect mode: bellman-ddsl (internal), terminology, 9 October 2026.

Return to slide 6

The AI reads from the ontology

Retrieval over a knowledge graph. The AI answers from a graph of entities and relations rather than from passages alone (surveyed in Pan et al. 2024; GraphRAG, Edge et al. 2024, which answers from summaries of entity clusters). Matsya retrieves passages of the project's documents with their sections, links and tags, and reads the typed object itself when a question names a stage file.

Training on generated pairs. The ontology's programs generate pairs of a file and its prose, or the comparator labels two files as one object or as two, so that a model can be trained on the pairs. AlphaProof's roughly one million natural-language problems, auto-formalised into about 80 million formal statements, are the largest instance. The project's comparator module, `meaning_map_training.compare`, labels such pairs, while the package's own training revises Matsya's instruction texts and settings against the round trip, with the language models' weights fixed.

What separates the two groups. In the five uses of the previous slide the checker, not the AI, decides what the file means, which makes the AI's reading inspectable. In these two the AI's reading improves but stays statistical, which is why they do not replace the checker.

Pan, Luo, Wang, Chen, Wang and Wu (2024), "Unifying large language models and knowledge graphs: a roadmap", *IEEE Transactions on Knowledge and Data Engineering* 36(7), 3580–3599 · Edge et al. (2024), "From local to global: a graph RAG approach to query-focused summarization", arXiv:2404.16130 · Hubert, Mehta, Sartran et al. (2026), *Nature* 651, 607–613, on the auto-formalised training problems · `meaning_map_training.compare` : bellman-ddsl (internal).

Return to slide 8

NVIDIA FP32: fused multiply-add

1. Objects and rounding. We use finite FP32 (IEEE 754 single-precision) values x, y, z. For the calculations below, RN_{32} rounds a real number to nearest FP32, with ties to even.

2. Interpretation Y. Fused multiply-add rounds once; separate multiplication and addition round twice.

$$\begin{array}{ccc} \text{written form} & \xrightarrow{\Upsilon} & \text{domain of meaning} \\ \texttt{fma}(x,y,z) & \longmapsto & \mathrm{RN}_{32}(xy+z) \\ \texttt{add}(\texttt{mul}(x,y),z) & \longmapsto & \mathrm{RN}_{32}(\mathrm{RN}_{32}(xy)+z) \end{array}$$

3. NVIDIA's example. Set $a = 1 + 2^{-23}$ and $b = -(1 + 2^{-22})$.

$$\mathrm{RN}_{32}(a^2 + b) = 2^{-46} \quad \text{(fused multiply-add)},$$

$$\mathrm{RN}_{32}\big(\mathrm{RN}_{32}(a^2) + b\big) = 0 \quad \text{(multiply, then add)}.$$

The input types agree, but the rounding rules and results differ.

Whitehead and Fit-Florea, NVIDIA, *Floating Point and IEEE 754*, §2.3.