Dependently Typed World Models

Toward Proof-Carrying Learned Simulators

Jordan Rule

jrule@uchicago.edu

August 2026

Abstract

World models are the internal working pictures through which modern learning systems rehearse possible futures. They compress experience, imagine consequences, and help an agent choose before it acts. Yet their most important commitments—what a latent state is allowed to mean, when evidence is sufficient for planning, and which actions are legitimate under uncertainty—are often left implicit. This paper argues that dependent type theory offers a practical way to make those commitments explicit and machine-checkable. We introduce the notion of a *proof-carrying world model*: a learned simulator whose states, predictions, and action proposals carry evidence that their preconditions are met. We connect this view to Joint Embedding Predictive Architecture (JEPA), where learning already proceeds from partial views toward predictive sufficiency rather than exhaustive reconstruction. The result is a research program in which world models remain empirical and data-driven while becoming clearer about admissibility, provenance, and safe use in multi-agent settings.

Keywords: world models, dependent type theory, Rocq, JEPA, model-based reinforcement learning, proof-carrying programs, alignment, multi-agent systems.

1. Introduction

One of the most consequential ideas in artificial intelligence is that an agent need not meet the world only in the instant. It may, before acting, carry an inner stage on which candidate futures can be rehearsed. In model-based reinforcement learning, that stage is the world model: a learned internal simulator that compresses experience into a latent state and uses that state to forecast what actions might bring about [Sutton, 1990, Ha and Schmidhuber, 2018, Hafner et al., 2020, Schrittwieser et al., 2020, Hafner et al., 2023, Ye et al., 2021]. The practical appeal is easy to see. An agent that can imagine cheaply does not need to experiment as recklessly in reality.

The idea has also widened. In one lineage, world models are used explicitly for planning. In another, adjacent lineage, predictive systems such as Joint Embedding Predictive Architecture (JEPA) learn latent representations by asking what one partial view of a situation should allow a system to say about another [LeCun, 2022, Assran et al., 2023]. These models do not insist on reconstructing every pixel or every sensory detail. They aim instead for a more selective kind of understanding: enough internal structure to support prediction, abstraction, and downstream decision-making.

And yet the central mystery remains curiously underdescribed. A latent state in most current systems is useful, trainable, and often powerful, but semantically reticent. It carries little machine-checkable explanation of what it is allowed to represent, which fragments of evidence may be combined into it, or what constraints its imagined futures are supposed to respect. These commitments, if they are written down at all, typically live outside the model—in design docs, policy layers, runtime checks, or after-the-fact audits.

This paper argues that dependent type theory offers a way to move some of that structure inward. In a dependently typed setting, a value can travel together with evidence that it satisfies the conditions required for its use. Proof need not arrive as a bureaucratic seal applied later; it can accompany the object at the moment of construction. We draw in particular on the Rocq proof assistant [The Coq/Rocq Development Team, 2024], using dependent records and constructive proofs to make world model interfaces explicit.

Before any formal machinery, however, there is a gentler picture. One can think of a world model as something assembled from partial views: a few observations here, a capability hint there, a short trace of what happened when an action was taken. Some views fit together; some contradict one another; some are simply too thin to justify high-confidence plans. This patchwork intuition is simple enough to grasp in ordinary language, yet it already echoes JEPA's core move: learn to predict what is hidden from what is seen, without pretending to know everything. Our claim is that dependent types can give this predictive practice a clearer grammar of admissible use.

Our contributions are:

1. A conceptual framework—the *proof-carrying world model*—that articulates how dependent types can annotate the operational interfaces of a learned simulator (§3).

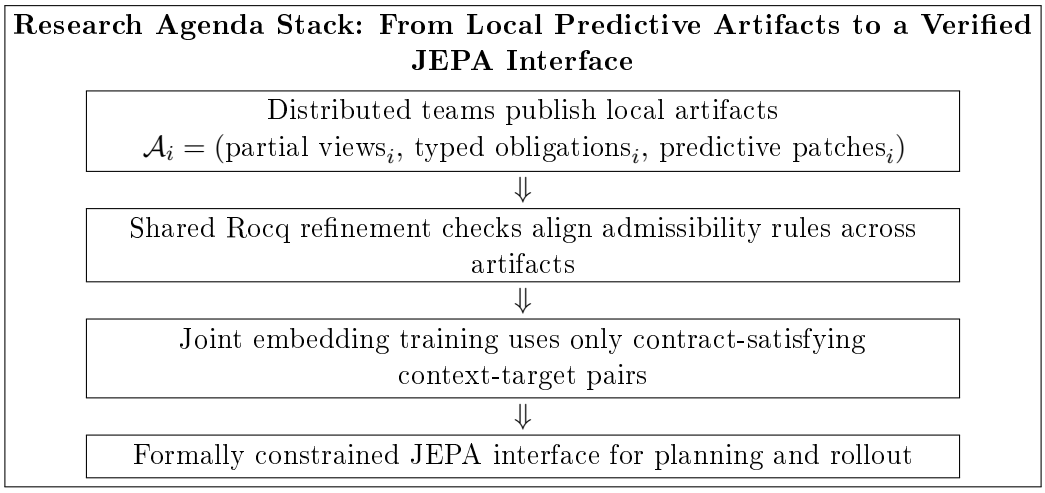

Figure 1: Illustrative pipeline for a globally collaborative program. The claim is not that learning disappears into logic, but that learned latent updates are filtered through shared, explicit admissibility contracts.

2. A practical bridge from partial views to typed predictive contracts, showing how JEPA-style representation learning can be coupled to explicit admissibility obligations for planning under uncertainty (§4).
3. Small Rocq formalizations illustrating these ideas, from basic proof-carrying state through typed context-target prediction and evidence-aware action proposals (§5).
4. A discussion of research directions, including explicit connections to alignment-aware planning and multi-agent trust (§6).

We do not claim a complete implementation. We offer a research direction and a working vocabulary for future development.

Figure 1 provides a compact visual of the agenda: local predictive artifacts feed shared admissibility checks, which in turn stabilize a JEPA-style planning interface.

2. Background

2.1. World Models in Reinforcement Learning

A world model, in the narrow reinforcement-learning sense, is a learned surrogate for the dynamics of an environment. Formally one may write it as a map $f_\theta : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z} \times \mathcal{R}$, sending a latent state and an action to a predicted successor state and reward. But that notation only hints at the broader intuition. A world model is a compressed, internal picture of how things change: what tends to follow what, which possibilities remain open, and which imagined futures are worth spending computation on.

The modern lineage is familiar. Dyna first made explicit the advantage of learning and planning in tandem [Sutton, 1990]. Ha and Schmidhuber's World Models popularized the image of an agent dreaming inside its own latent space

[Ha and Schmidhuber, 2018]. Dreamer and its successors showed that long-horizon imagination could become a practical engine for control [Hafner et al., 2020, 2021, 2023]. MuZero and EfficientZero demonstrated that one need not reconstruct the visible world in full detail in order to plan effectively [Schrittwieser et al., 2020, Ye et al., 2021].

JEPA extends the same family resemblance in a slightly different direction. Rather than learning to regenerate an entire input, a JEPA-style system learns to predict the representation of a withheld target view from an available context view [LeCun, 2022, Assran et al., 2023]. What matters is not visual fidelity but predictive sufficiency: the latent state should preserve the structure needed to say something reliable about what lies beyond the current glimpse. In that sense, JEPA is not a departure from the world-model story so much as a sharpening of one of its oldest instincts.

Across these traditions, a recurring tension remains between *predictive skill* and *explicit meaning*. A model may be extraordinarily good at forecasting what happens next and still be silent about what sort of state it is claiming to inhabit, what sources of evidence it has mixed together, and what operational limits should govern its use. This gap is the starting point for our proposal.

2.2. Dependent Type Theory and the Rocq Proof Assistant

Dependent type theory [Martin-Löf, 1984, Coquand and Huet, 1988] begins from a simple but powerful thought: the description of an object may be allowed to mention the object itself. Instead of saying merely "this is a vector," one may say "this is a vector of length n," and insist that n be part of the object's type. A familiar example is $\mathsf{Vec}(n)$, the type of vectors of exactly n elements. More generally, a value of type $\{x : A \mid P(x)\}$ is both a value in A and evidence that it satisfies $P(x)$.

For readers outside formal methods, the philosophical point matters more than the notation. In an ordinary program, one often constructs a value first and checks later whether it meets the relevant conditions. In a dependently typed program, one can require those conditions at birth. Some classes of mismatch never become runtime problems because the system refuses to construct the wrong object in the first place.

Rocq (formerly Coq) [The Coq/Rocq Development Team, 2024] implements the Calculus of Inductive Constructions [Coquand and Huet, 1988, Paulin-Mohring, 2015], a rich dependent type theory supporting:

- *Inductive types* and pattern matching;
- *Dependent records* (`Record`) that bundle values with their correctness conditions;
- *Proof terms* as first-class citizens of the same language as programs;
- *Extraction* to OCaml or Haskell, enabling certified programs to run outside the proof environment.

The MathComp library [Gonthier et al., 2013] provides mature foundations for constructive formalization in Rocq and practical proof engineering at scale.

2.3. Partial Views, Predictive Sufficiency, and Contracts

Before introducing heavy formal structure, it helps to start with a familiar scene. An agent rarely observes a complete world. It sees fragments: a camera crop, a noisy sensor packet, a short action history, a partial map of who said what. JEPA-style learning asks what one such fragment should allow us to infer about another [LeCun, 2022, Assran et al., 2023]. This is a remarkably practical stance for autonomous systems: do not reconstruct everything; recover enough structure to make the next decision responsibly.

Dependent types sharpen that stance. They let us say: this prediction is usable for planning only when certain evidence conditions hold. Instead of treating latent vectors as universally admissible, we can treat them as claims with scope. Some claims are broad enough for low-risk suggestion. Others are narrow and should not authorize consequential actions. The technical language is precise, but the intuition is plain: uncertainty is not an afterthought; it is part of the object we manipulate.

2.4. From JEPA Representations to Typed Planning

In many deployments, the central failure mode is not poor prediction in isolation but overconfident use of prediction in the wrong context. A model can be right on average and still dangerous in the moments that matter. The bridge we propose is therefore interface-level: keep JEPA as the statistical engine for partial-view prediction, and place a typed contract at the boundary where predictions become plans.

Under this view, a planner does not merely consume a latent state. It consumes a *checked* latent state, paired with explicit witnesses about provenance, coverage, or confidence regime. The point is not to freeze learning into rigid symbolism; it is to stop treating all latent states as equally action-worthy.

3. A Framework for Proof-Carrying World Models

3.1. Motivation

Most learned latent states are described, at bottom, as arrays of numbers. That description is enough for optimization, but it says very little about meaning. It does not say whether the state was assembled from compatible evidence, whether it presupposes a certain capability scope, or whether it is even the kind of state a planner should be allowed to imagine from. Those judgments are usually external to the model.

Our proposal is to make admissibility *intrinsic*. If a state is meant to stand for "a history built from non-conflicting observations under capability scope κ," then its type should say so. If a plan is meant to require certain permissions, the need for those permissions should be visible in the plan's construction. Scope confusion then ceases to be a policy bug discovered late; it becomes a type error discovered early.

3.2. Proof-Carrying State

We define a *proof-carrying state* as a dependent record bundling a representational value with a machine-checked admissibility witness.

Definition 3.1 (Proof-Carrying State)**.** *Let S be a set of representational states and let* $\mathsf{Adm} : S \to \mathsf{Prop}$ *be an admissibility predicate. A* proof-carrying state *is a pair (s, h) where $s \in S$ and $h : \mathsf{Adm}(s)$.*

The key property is psychological as well as technical. One is no longer asked to trust that a state has been blessed somewhere upstream. The blessing is part of the object. Construction and certification occur together, or not at all.

3.3. Proof-Carrying World Model Interface

A world model operating over proof-carrying states must be designed so that:

1. *Observation closure*: applying the observation operator to an admissible state produces an admissible state.
2. *Imagination validity*: applying the imagination (rollout) operator to an admissible state produces a valid (non-undefined) state.

These are not runtime checks but proof obligations discharged at definition time. A world model that cannot satisfy them is, in a precise sense, *ill-typed*—it cannot be instantiated.

The following Rocq sketch captures the interface:

```
From mathcomp Require Import ssreflect ssrbool ssrnat seq.

Section TypedWorld.
Context {State Observation Action : Type}.
Context (Adm : State -> Prop).

Record CheckedState := {
  state_val : State;
  state_ok  : Adm state_val
}.

Record WorldModel := {
  observe : CheckedState -> seq Observation ->
     CheckedState;
  imagine : CheckedState -> seq Action -> option
     CheckedState
}.

End TypedWorld.
```

Listing 1: Proof-carrying world model interface in Rocq

This interface is intentionally schematic. The key point is that planning-time outputs are partial (`option`) and must justify admissibility on success. A JEPA-style model can be read through the same lens: context builds a latent state, prediction proposes a target representation, and typed obligations specify when

that representation is safe to use for downstream planning.

4. From Partial Views to Planning Contracts

4.1. JEPA and the Grammar of Missing Information

JEPA's conceptual strength is that it takes missing information seriously. It asks what can be inferred from what is present, without pretending that hidden structure has been directly observed [LeCun, 2022, Assran et al., 2023]. For autonomous systems operating in the wild, this is the norm rather than the exception: data arrive late, sensors disagree, and collaborators report through uneven trust channels.

Dependent types offer a useful complement: they let us distinguish between "predicted" and "safely actionable" in the type of an object, not only in a policy note outside the model. This creates a clean handoff: JEPA supplies statistical abstraction; typed contracts govern admissible use.

4.2. Planning Under Limited Knowledge

When knowledge is partial, planning should degrade gracefully rather than fail silently or overreach. A typed interface can enforce this behavior by requiring that high-impact actions carry explicit witnesses that their preconditions are met under the current evidence profile. If those witnesses are absent, the plan construction itself fails.

This is not merely elegant formalism. It is operationally relevant in settings where autonomous systems must choose under ambiguity. A planner that cannot prove enough should return an abstention, a request for more context, or a lower-risk fallback policy. Typed construction gives these behaviors first-class status.

4.3. A Collaborative Research Loop

Figure 2 sketches the intended workflow. Distributed research groups contribute partial-view datasets, local obligations, and refinement lemmas. Shared checks keep only artifacts that satisfy the agreed contracts. JEPA-style training then updates representations on this filtered corpus, and the next iteration tightens both statistical quality and formal clarity.

4.4. Immediate Alignment Value in Multi-Agent Settings

The safety advantage of this bridge is concrete. In multi-agent environments, failures often arise not from a single bad prediction but from a mismatch between what one agent inferred and what another agent was entitled to do. Typed world model interfaces make these mismatches visible at construction time. Provenance, permission scope, and uncertainty regime can all be made part of the data that planning consumes.

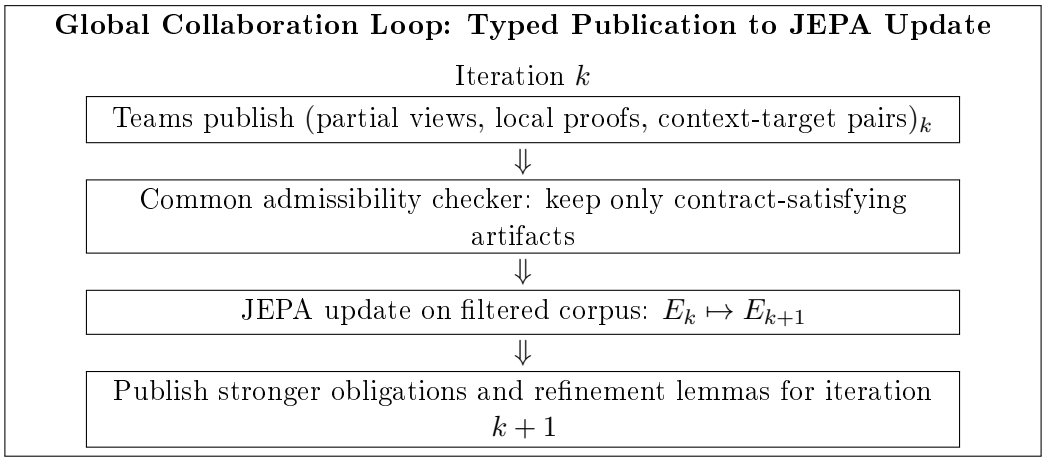

Figure 2: Indicative training-governance cycle. Representation quality improves statistically while admissibility guarantees improve constructively.

The resulting architecture remains learned and adaptive, but it becomes less willing to improvise beyond its evidence. That is precisely the behavior we want from autonomous systems that must act among other autonomous systems.

5. Illustrative Rocq Formalizations

5.1. A Tiny Example of Proof-Carrying State

Before introducing additional interface structure, we illustrate the simplest instance of proof-carrying state: a rollout record whose `horizon` field is certified by a length witness.

```
From Coq Require Import List Arith.
Import ListNotations.

Record CheckedRollout := {
  horizon         : nat;
  actions         : list nat;
  horizon_matches : length actions = horizon
}.

Definition single_action (a : nat) : CheckedRollout :=
  {| horizon         := 1;
     actions         := [a];
     horizon_matches := eq_refl |}.
```

Listing 2: Minimal proof-carrying rollout record

The field `horizon_matches` is not a runtime assertion but a proof term. It is discharged by `eq_refl` at the point of construction, confirming that the singleton list `[a]` has length 1. A `CheckedRollout` with mismatched horizon cannot be constructed; the attempt would produce a type error. This is the paper's basic

promise in miniature: if a state claims to summarize a horizon, that claim can be made part of the object rather than an informal comment about it.

5.2. Typed Context-Target Prediction

The following Rocq definition illustrates a proof-carrying context-target prediction pair. The key point is that the prediction is only usable when the context is known to be non-empty.

```
From Coq Require Import List Arith.
Import ListNotations.

Record PredictivePair := {
  context_tokens : list nat;
  target_tokens  : list nat;
  context_nonempty : length context_tokens > 0
}.

Definition safe_predict (p : PredictivePair) : option (
   list nat) :=
  if Nat.eqb (length (context_tokens p)) 0
  then None
  else Some (target_tokens p).
```

Listing 3: Typed context-target prediction record

The `safe_predict` function is deliberately modest: prediction is allowed only when the context witness says there is actually context to condition on. The point is not sophistication but shape. Even this tiny example reflects the design rule that planning-time prediction should expose admissibility directly.

5.3. Evidence-Scoped Action Proposals

A second pattern is to type action proposals by the evidence they require. Rather than "choose action," one defines "choose action under evidence profile e." When the profile is weak, the system can still reason, but only into action families allowed for weak evidence.

```
From Coq Require Import List Arith.

Inductive RiskLevel := Low | High.

Record Evidence := {
  confidence : nat;
  confidence_ok : confidence <= 100
}.

Definition can_take_high_risk (e : Evidence) : bool :=
  Nat.leb 80 (confidence e).
```

```
Definition propose_action (e : Evidence) : RiskLevel :=
  if can_take_high_risk e then High else Low.
```

Listing 4: Evidence-scoped action proposal

The program is simple, but the design lesson is durable: evidence quality becomes an explicit input to action construction. In richer systems, those thresholds are replaced by proofs over learned uncertainty estimates and provenance contracts.

5.4. Composing Multi-Agent Signals with Explicit Trust

The final pattern is trust-aware composition. In cooperative and adversarial settings alike, a planner must know not only *what* was observed, but *who observed it* and under what trust assumptions.

```
From Coq Require Import List Arith.

Inductive Trust := Unverified | Verified.

Record Observation := {
  payload : nat;
  trust   : Trust
}.

Definition usable_for_critical_plan (o : Observation) :
   bool :=
  match trust o with
  | Verified => true
  | Unverified => false
  end.
```

Listing 5: Trust-annotated observation packet

This sketch captures the core operational idea: critical planning paths should be typed to require stronger evidence than exploratory paths. The architecture can still learn from all signals, but it need not treat all signals as equally authoritative at decision time.

6. Discussion and Research Directions

6.1. Alignment-Aware Planning

The alignment problem for learned agents [Russell, 2019, Gabriel, 2020] is, in part, a problem of specification: how does an agent know that the actions it is considering are within the scope of what it is supposed to do? Current approaches rely heavily on reward shaping, RLHF [Christiano et al., 2017, Bai et al., 2022b], and constitutional constraints [Bai et al., 2022a], all of which are intrinsically

extrinsic—they surround the model with evaluative signals rather than embedding constraints within its representational structure.

A proof-carrying world model offers a complementary approach. By requiring that high-stakes action types carry typed witnesses of their preconditions, the planning procedure is structurally prevented from producing plans that violate stated constraints—not because a guard intercepts the output, but because the output type cannot be constructed without the required evidence. The difference is analogous to the difference between a runtime bounds check and a statically verified array access: the latter eliminates a class of error by construction, before execution begins.

We envision a planning module whose action type is parameterized by an evidence profile ϵ: an action of type $\mathsf{Action}(\epsilon)$ may only rely on assumptions justified by ϵ. Confidence escalation then becomes a type error.

The JEPA connection matters here as well. If a predictive model is trained to infer withheld structure from partial context, the question naturally follows: what kinds of context are sufficient for which kinds of action? Dependent types offer a way to state that question sharply. A latent prediction can be treated not as a free-floating hint, but as an object whose admissible uses are tied to the evidence from which it was formed.

6.2. Multi-Agent Trust and Fragment Provenance

In multi-agent settings, observations arrive from diverse sources with varying degrees of trust. A dependently typed world model can represent this by annotating observation histories with *provenance types*: a history of type $\mathsf{Hist}(\tau)$ was collected under trust level τ, and the type system can enforce that a planner operating under trust level τ' may only act on histories whose provenance satisfies $\tau' \preceq \tau$ for some trust ordering $\preceq$.

This makes cross-provenance contamination detectable at the type level—a property with direct practical significance whenever agents interact with adversarially constructed inputs or aggregated data of heterogeneous origin.

7. Related Work

World models. Ha and Schmidhuber's World Models [Ha and Schmidhuber, 2018] helped establish the "dream-then-act" paradigm for visual planning. The DreamerV1–V3 series [Hafner et al., 2020, 2021, 2023] demonstrated that continuous control and Atari tasks can be mastered through pure imagination. MuZero [Schrittwieser et al., 2020] showed that a latent model without ground-truth reconstruction suffices for search-based planning. EfficientZero [Ye et al., 2021] extended this to extreme sample efficiency. None of these architectures address typed invariants on the model interface.

JEPA and predictive representation learning. LeCun's program for autonomous machine intelligence [LeCun, 2022] argues that useful intelligence should

be grounded in predictive world models that operate in latent space rather than through exhaustive reconstruction. I-JEPA [Assran et al., 2023] makes this idea concrete by learning to predict target embeddings from context embeddings. Our use of typed predictive contracts is meant to be complementary: JEPA supplies a modern account of how predictive latent structure may be learned, while Rocq supplies a way to state, with precision, what such predictions are allowed to authorize in planning.

Formal methods and learning. Seshia et al. [Seshia et al., 2018] survey formal specification and verification techniques applicable to learning-enabled systems, emphasizing runtime monitors and assume-guarantee contracts. Neural network verification [Katz et al., 2019, Liu et al., 2021] focuses on robustness of fixed networks. CompCert [Leroy, 2009] and seL4 [Klein et al., 2009] demonstrate that large software systems can be given machine-checked correctness proofs; our work proposes that learned interfaces can participate in such developments.

Alignment and specification. Russell [Russell, 2019] frames alignment as a problem of misspecified objectives. Christiano et al. [Christiano et al., 2017] and Bai et al. [Bai et al., 2022b,a] address alignment through human feedback and constitutional constraints. Gabriel [Gabriel, 2020] provides a conceptual taxonomy. Our proposal is orthogonal: rather than specifying objectives more accurately, we propose to specify the operational interface of the model more precisely.

8. Conclusion

We have argued that world models become easier to reason about when they are understood as accountable constructions rather than opaque latent containers. JEPA reminds us that useful predictive structure can be learned from partial views. Dependent type theory adds a discipline for saying what those predictions license: when a state is admissible, when evidence is sufficient, and when a plan should abstain.

The proposal is not that proofs replace learning. A well-typed world model may still be empirically wrong about the world. The proposal is narrower and, for that reason, practical: that learned internal models can carry clearer commitments about admissibility, provenance, and use in multi-agent systems. If a system is going to imagine on our behalf, it is reasonable to ask that its imagination come with explicit terms.

References

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. *arXiv preprint arXiv:2301.08243*, 2023.

Yuntao Bai et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022a.

Yuntao Bai et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022b.

Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Thierry Coquand and Gérard Huet. The calculus of constructions. *Information and Computation*, 76(2–3):95–120, 1988.

Iason Gabriel. Artificial intelligence, values, and alignment. *Minds and Machines*, 30:411–437, 2020.

Georges Gonthier et al. A machine-checked proof of the odd order theorem. In *International Conference on Interactive Theorem Proving (ITP)*, pages 163–179, 2013.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations (ICLR)*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Guy Katz, Derek A. Huang, Duligur Ibeling, et al. The marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification (CAV)*, pages 443–452, 2019.

Gerwin Klein et al. seL4: Formal verification of an OS kernel. In *ACM Symposium on Operating Systems Principles (SOSP)*, pages 207–220, 2009.

Yann LeCun. A path towards autonomous machine intelligence. *OpenReview preprint*, 2022.

Xavier Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7):107–115, 2009.

Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, and Mykel J. Kochenderfer. Algorithms for verifying deep neural networks. *Foundations and Trends in Optimization*, 4(3–4):244–404, 2021.

Per Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

Christine Paulin-Mohring. Introduction to the calculus of inductive constructions. In David Delahaye and Bruno Woltzenlogel Paleo, editors, *All about Proofs, Proofs for All*. College Publications, 2015.

Stuart Russell. *Human Compatible: Artificial Intelligence and the Problem of Control*. Viking, 2019.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, et al. Mastering atari, go, chess and shogi by planning with a learned model. In *Nature*, volume 588, pages 604–609, 2020.

Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry. Formal specification for deep neural networks. In *Automated Technology for Verification and Analysis (ATVA)*, pages 20–34, 2018.

Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning (ICML)*, pages 216–224, 1990.

The Coq/Rocq Development Team. The rocq proof assistant reference manual. `https://rocq-prover.org/`, 2024.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.