ErisML: A Unified Modeling Language for Pervasive AI Governance

Andrew Bond, Senior Member, IEEE
Department of Computer Engineering, San José State University
Email: andrew.bond@sjsu.edu

6 December 2025

SJSU SAN JOSÉ STATE UNIVERSITY

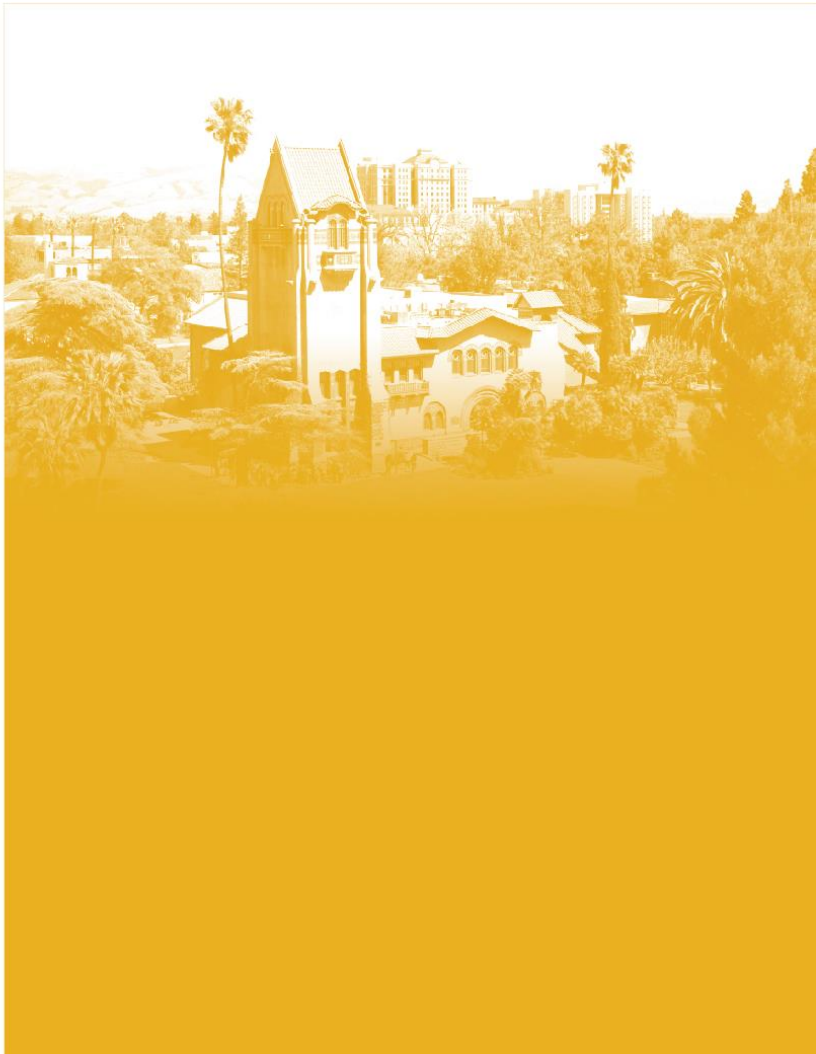

Table of Contents

Abstract

Foundation models are migrating from data centers to pervasive environments—homes, hospitals, factories, cities—where they encounter fundamental challenges: heterogeneous sensors, conflicting objectives, ambiguous regulations, and strategic multi-agent interaction. Current approaches fragment the problem: prompts encode goals, rules capture policies, code provides glue. We argue this fragmentation will not scale.

This paper presents **ErisML**, a vision for a unified modeling language that integrates environment dynamics, agent capabilities, multi-objective intents, normative structures, and strategic interaction into a single, machine-interpretable substrate. Rather than a finished solution, we articulate the design space, fundamental research challenges, and technical requirements for such a language. Through concrete examples in healthcare, mobility, and industrial settings, we illustrate both the promise and profound difficulties of formal governance in pervasive AI systems.

Index Terms: pervasive computing, foundation models, AI agents, modeling languages, multi-agent systems, normative systems, ambient intelligence

I. Introduction

A. The Challenge of Pervasive AI

Foundation models are escaping the data center. As they move into homes, hospitals, and factories, they face what we call the "golden apple problem"—when AI agents must balance patient comfort against energy costs, privacy regulations against clinical safety, or throughput against accessibility, who decides what is "fairest"?

Current pervasive AI systems exhibit four types of chaos:

Observational chaos: Motion sensors conflict with phone GPS. Healthcare monitors show vital sign spikes that could be distress or malfunction.

Intentional chaos: Energy management wants to shed load; medical monitors demand power for insulin pumps; humans want entertainment; policy says critical devices have priority—but what counts as "critical"?

Normative chaos: HIPAA demands privacy, public health law mandates reporting, advance directives limit interventions, and AI training objectives say "maximize health outcomes." Which norm takes precedence?

Temporal chaos: Distribution shifts are constant. Models trained on summer data face winter storms. Policies optimized for one resident must adapt when visitors arrive. Norms for routine operation must flex during emergencies.

Current approaches fragment the problem: - **Prompts** encode goals but are brittle and unverifiable - **Rules** capture norms but live separately from models - **Code** provides orchestration but obscures intent - **Planning languages** (PDDL) specify goals but lack multi-agent reasoning and normative structures

B. Vision: A Unified Substrate

We propose **ErisML** (named for the Greek goddess of discord) as a unified modeling language providing a single substrate for:

1. **Environment**: State spaces, dynamics, uncertainty, observability
2. **Agency**: Capabilities, beliefs, memory, decision interfaces
3. **Intent**: Multi-objective utilities, preferences, goal structures
4. **Norms**: Permissions, obligations, prohibitions, sanctions
5. **Dynamics**: Multi-agent interactions, strategic behavior, emergence

Why unification? Because these elements are inseparable in pervasive computing. Agent actions depend on environment state. Environment dynamics change based on multi-

agent interactions. Norms constrain which intents can be pursued. Chaos emerges from their interaction.

C. Contributions

This paper offers: - **Design space analysis** exploring fundamental trade-offs - **Research challenges** that must be addressed - **Technical requirements** derived from pervasive computing realities - **Formal semantics** showing what ErisML could provide - **Case study** demonstrating capabilities and limitations - **Research roadmap** for the community

We invite critique and collaborative refinement. The goal is to catalyze conversation about common substrates for pervasive, AI-enabled systems.

II. Design Space and Related Work

A. Fundamental Trade-offs

Expressiveness vs. Tractability: More expressive languages capture richer phenomena (continuous dynamics, strategic interaction, recursive norms) but become intractable. Our position: **stratified by complexity**—core constructs remain decidable; extension mechanisms allow expressing (but not automatically solving) complex scenarios.

Specification vs. Learning: We can't specify everything in advance. But unconstrained learning violates norms. Our position: **learning as constrained optimization**—objectives may be learned, but norms provide hard boundaries. Learning protocols become first-class objects.

Autonomy vs. Control: Users want autonomous handling of routine cases but must retain ultimate control. Our position: **mixed-initiative interaction** as first-class concept, with explicit policies for escalation, override, and explanation.

Centralized vs. Distributed: Pervasive systems face consensus problems, partial failures, and adversarial dynamics. Our position: **federation-native**—specifications scoped by agent/organization/jurisdiction, with explicit conflict resolution.

Human-Legibility vs. Machine-Optimality: Auditors need readable specifications; solvers need efficient encodings. Our position: **multiple representation levels**—canonical formal representation with projections to human-readable views.

B. Related Work and Positioning

Planning Languages: PDDL [5] and variants specify deterministic/stochastic planning problems but lack normative structures and multi-agent strategic reasoning.

Temporal Logics: LTL, CTL, and Signal Temporal Logic (STL) handle continuous-time dynamics with bounded constraints but don't integrate learning or multi-objective optimization.

Policy Languages: Rego (Open Policy Agent) and XACML provide industrial policy-as-code but lack environment modeling, dynamics, and planning integration.

Normative Multi-Agent Systems: Extensive work on deontic logic in MAS [8,31,32] provides foundational concepts but limited integration with modern learning-based agents and formal verification tools.

BDI Architectures: AgentSpeak and Jason [33] offer agent programming with beliefs/desires/intentions but lack formal semantics for learning and normative constraints.

Approach	Strengths	Limitations	ErisML Difference
PDDL	Mature tools, efficient	No norms, no multi-agent	Adds normative layer + strategic interaction
Temporal Logic	Expressive, verifiable	No learning, no utilities	Integrates with RL + multi-objective optimization
Policy Languages	Production-ready	No dynamics, no planning	Adds environment modeling + agent reasoning
Normative MAS	Rich deontic logic	Limited tool support, no learning integration	Executable specifications + learning backends

III. ErisML Architecture

A. Core Philosophy

Rather than treating uncertainty, conflicts, and strategic behavior as exceptions, ErisML makes them first-class constructs:

- **Observational**: Partial, noisy, conflicting sensor data
- **Intentional**: Multi-objective, often conflicting goals
- **Normative**: Ambiguous, contextual, sometimes contradictory rules
- **Interactive**: Strategic agents, emergent behaviors
- **Temporal**: Distribution shift, non-stationarity, adaptation

B. Illustrative Syntax

We present ErisML through a smart hospital room example:

C. Formal Semantics

ErisML specifications compile to a **Norm-Constrained Stochastic Game** (NCSG):

Definition: A tuple ⟨N, S, {A_i}, {Ω_i}, T, {U_i}, Φ, C⟩ where: - N = set of agents - S = state space (finite, continuous, or hybrid) - A_i = action space for agent i - Ω_i = observation space for agent i (Ω_i ⊇ S for partial observability) - T: S × A_1 × ... × A_n → Δ(S) = stochastic transition kernel - U_i: S × A → ℝ^k = multi-objective utility for agent i - Φ = {ϕ_j} = set of normative constraints - C: Φ × S → priority_ordering = conflict resolution function

Normative Constraints: Each ϕ_j: S × A × Time → {permit, prohibit, oblige} with: - **Temporal scope**: Obligations have deadlines; permissions may have duration limits - **Context**

dependence: Norms activated based on state predicates - **Jurisdictional scope**: Norms apply to subsets of agents/states

Norm-Gated Policy: Agent i's policy π_i: Δ(S) → Δ(A_i) must satisfy:

```
π_i(a | b) > 0 only if ∃s ∈ support(b):
  ∀ϕ ∈ Φ: ϕ(s, a, t) ∈ {permit, oblige} ∨ C(Φ, s) overrides ϕ
```

Multi-Objective Optimization: Vector utilities U_i ∈ $\mathbb{R}$^k handled via: - **Scalarization**: w · U_i for weights w ∈ Δ^{k-1} - **Lexicographic**: (U_i)_1 ≫ (U_i)_2 ≫ ... (strict priorities) - **Constrained**: max (U_i)_1 subject to (U_i)_j ≥ τ_j ∀j > 1

Strategic Equilibrium: Solution concept depends on assumptions: - **Cooperative**: Joint policy maximizes Σ_i U_i subject to norms - **Competitive**: Constrained Nash equilibrium - **Stackelberg**: Human as leader, AI agents as followers - **Open research problem**: What equilibrium concept is appropriate for human-AI teams under normative constraints?

D. Computational Complexity

Fragment	State Space	Norms	Agents	Verification	Planning
ErisML-Core	Finite, ≤10^6 states	Propositional	Single	O(S·N) model checking	NP-complete
ErisML-Multi	Finite, ≤10^6 states	Propositional	Multiple	PSPACE-complete	PPAD-complete (Nash)
ErisML-Temporal	Continuous time	Propositional LTL	Single	PSPACE-complete	Undecidable
ErisML-Full	Hybrid/infinite	First-order logic	Multiple	Undecidable	Undecidable

Design principle: Core language remains in decidable fragments. Extensions marked with complexity warnings.

E. Compilation Targets

ErisML must compile to diverse backends, each with semantic limitations:

Backend	Preserves	Loses	Verification
PDDL	Deterministic dynamics, goals	Norms → preconditions (provenance lost), strategic interaction	Classical planning guarantees
PRISM	Stochastic dynamics, probabilistic properties	Strategic interaction, multi-agent	Probabilistic model checking
Safe RL	Continuous optimization, learning	Logical norms → soft constraints (may violate during exploration)	Statistical testing + runtime monitors
Multi-agent RL	Strategic learning	Hard logical constraints difficult to enforce	Empirical testing, no guarantees

Key insight: Semantics-preserving compilation is impossible for some backends. We provide **best-effort approximation with explicit documentation of limitations**.

IV. Research Challenges

We identify five critical challenges (condensed from original ten):

Challenge 1: Specification-Reality Gap

Problem: How do we ensure ErisML specifications accurately capture behavior of learned models with billions of parameters?

Approaches: - Runtime conformance checking (behavioral types) - Statistical testing (does deployment match spec predictions?) - Formal synthesis (generate models from specs)

Open question: Is there a fundamental gap between symbolic specifications and learned representations?

Challenge 2: Norm Consistency and Conflict

Problem: Real normative systems contain contradictions and ambiguities. ErisML must handle this without undefined behavior.

Approaches: - Defeasible logic (norms have default priority, context overrides) - Probabilistic norms (obligations hold with varying certainty) - Meta-norms (rules about which norms apply when)

Open question: Is norm consistency decidable for realistic ErisML fragments? (For Core with propositional norms: yes, $O(N^2)$. For Full with first-order: no, undecidable.)

Challenge 3: Learning Under Normative Constraints

Problem: How do we ensure learned policies respect norms, especially when norms are complex and changing?

Approaches: - Constrained RL (Lagrangian methods, shield synthesis) - Norm-aware architectures (build normative reasoning into model structure) - Verification-in-the-loop (only deploy policies passing formal verification)

Open question: Can we prove a learning algorithm never violates norms in novel states?

Challenge 4: Distributed Specification and Governance

Problem: Pervasive systems span organizational boundaries. Who writes specifications? What happens when specifications conflict?

Approaches: - Federated specifications (local ErisML with conflict resolution protocols) - Jurisdictional layering (federal norms override state override local) - Negotiation protocols (automated contract negotiation)

Open question: How do we ensure global safety when no single entity controls all agents?

Challenge 5: The Value Alignment Problem

Problem: Even with perfect specifications and verification, whose values should the system optimize? When values conflict, who decides?

Approaches: - Explicit value pluralism (encode multiple value systems, let users choose) - Democratic processes (vote on trade-off weights) - Context-dependent ethics (different values in different situations)

Open question: Is value alignment solvable by better specifications, or is it inherently social/political?

V. Technical Requirements

Based on challenges, we derive requirements (condensed from original ten):

R1: Formal Semantics with Complexity Guarantees

Every construct has formal semantics with known computational complexity. Fragments marked by decidability: - **ErisML-Core**: Decidable, polynomial verification - **ErisML-Plus**: Expressive, NP-complete - **ErisML-Full**: Turing-complete, undecidable properties

Tools warn when crossing boundaries: *"Warning: First-order norms used. Verification may not terminate."*

R2: Compositional Semantics

Meaning of composite specifications determined from parts: - Environment dynamics compose via parallel (⊕) or sequential (⊙) composition - Utilities compose via weighted sum, lexicographic (≫), or Pareto - Norms compose via priority hierarchies (⊳)

Example: `norms Combined = HIPAA ⊳ HospitalPolicy ⊳ DeviceDefaults`

R3: Multi-Fidelity Modeling

Support multiple abstraction levels with verified relationships: - Abstract specifications (verifiable, coarse) - Concrete implementations (executable, detailed) - Abstraction functions mapping concrete → abstract - Refinement proofs ensuring concrete satisfies abstract

R4: Audit Trail and Provenance

Every action traceable to specification clauses permitting it:

R5: Graceful Degradation Under Uncertainty

Explicit uncertainty budgets as first-class constructs:

VI. Case Study: Smart Home Healthcare

Scenario: Margaret, 78, lives alone with AI care assistance. She has diabetes, mild cognitive impairment, and mobility issues. The system monitors vitals, manages medication reminders, coordinates with nurses, and alerts family during emergencies.

A. The Challenges

Observational chaos: Glucose monitor occasionally fails. Motion sensors have false negatives. Self-reports are inconsistent with cognitive state.

Intentional chaos: Margaret wants independence and privacy. Her daughter wants safety and monitoring. The AI optimizes health outcomes. Insurance wants cost containment.

Normative chaos: HIPAA limits data sharing, but elder abuse laws require reporting. Advanced directives say no hospitalization unless critical—but what is "critical"?

Temporal chaos: Cognitive state fluctuates daily. Winter brings different risks than summer. Care plan must adapt as she declines.

B. ErisML Solution (Excerpt)

C. What ErisML Enables

1. **Legible trade-offs**: Family can see autonomy is 2nd priority after safety
2. **Auditable decisions**: “Why didn’t system call 911?” → “Glucose was 55, above threshold of 40”
3. **Adaptive constraints**: Norms flex with cognitive state, time of day, emergency status
4. **Multi-stakeholder governance**: Margaret, daughter, care team, and regulators have voice
5. **Longitudinal validation**: Test adaptation over months, not just point-in-time

D. What ErisML Cannot Prevent (Failure Modes)

Gradual Cognitive Decline: - ErisML has discrete states (clear/confused/unresponsive) - Reality: Decline is gradual with good/bad days - Risk: System flips between autonomy and paternalism erratically - Residual risk: Requires human oversight during state transitions

Privacy-Safety Gaming: - Margaret might trigger false alarms to get attention - Daughter might engineer “emergencies” to get surveillance - Risk: Norm gaming by humans, not just AI - Mitigation: Partial (reputation systems, usage patterns analysis)

Sensor Failure Cascades: - If glucose monitor fails during actual emergency, system may not detect danger - False positives from noisy sensors may cause alert fatigue - Risk: Over-reliance on unreliable sensors - Mitigation: Multi-sensor fusion, uncertainty-aware decision making

VII. Research Roadmap

Phase 1: Foundations (Years 1-2)

Goals: Formal foundations, core language, proof-of-concept

Deliverables: 1. Formal semantics document with operational semantics, type system, norm interpretation 2. Complexity characterization for fragments (decidability, tight bounds) 3. Reference parser and interpreter for ErisML-Core 4. Compilers to PDDL, PRISM, Python simulation 5. Benchmark suite: 50+ scenarios across healthcare, mobility, industrial

Concrete Milestones: - M1.1: Inter-rater reliability on expressiveness: 3 experts, agreement > 0.8 (Krippendorff’s α) - M1.2: Verification time < 1 minute for specs with < 100

state variables (ErisML-Core) - M1.3: Compilation correctness: 100% of Core specs compile to PDDL/PRISM, verified via metamorphic testing

Phase 2: Integration and Learning (Years 2-4)

Goals: Integrate with foundation models, federated learning, edge deployment

Deliverables: 1. Foundation model integration layer (LLM-based agents with ErisML governance) 2. Constrained RL algorithms provably respecting ErisML norms 3. Federated learning protocols with differential privacy encoded as ErisML policies 4. Edge optimization toolkit with ErisML runtime monitors 5. Real-world pilots in 3 domains

Concrete Milestones: - M2.1: Healthcare pilot: 3 smart homes, 6 months, norm violation rate < 2%, patient satisfaction > 70% - M2.2: Campus mobility: 5 shuttles, 10k trips, zero safety incidents, throughput within 10% of baseline - M2.3: Edge deployment: Norm checking latency < 100ms on Raspberry Pi 4

Phase 3: Scaling and Standards (Years 4-5)

Goals: Scale to multi-stakeholder systems, develop standardization pathway

Deliverables: 1. Distributed ErisML with federation protocols, Byzantine tolerance 2. Visual modeling tools (IDE, graphical editors) 3. Conformance test suite for implementations 4. Standards proposal to IEEE P2863 / ISO working groups 5. Open-source ecosystem

Success Metrics: - 5+ independent implementations - 100+ public specifications, 10k+ downloads - IEEE/ISO working group formed with 20+ organizational members - Deployments span 3+ countries

VIII. Related Standards and Interoperability

ErisML must align with emerging regulatory and technical standards:

Standard	Requirement	ErisML Support	Gap
IEEE P7001 (Transparency)	Explainable decisions	Audit trail (R4), provenance	Need natural language summaries
IEEE P7003 (Algorithmic Bias)	Fairness metrics	Multi-objective intents encode fairness	No built-in bias detection tools
EU AI Act	High-risk system documentation	Formal specs provide documentation	Need certification pathway
ISO 23894 (AI Risk Mgmt)	Risk assessment, monitoring	Runtime monitoring, safety metrics	Need standardized risk scoring
NIST AI RMF	Trustworthy AI characteristics	Maps to requirements R1-R5	Need detailed guidance documents

IX. Limitations and Risks

A. Technical Limitations

Computational intractability: Full verification may be NP-hard or undecidable. We verify approximations, not actual systems (especially for learned components).

Specification incompleteness: Open-world environments prevent anticipating all states/events. Systems will encounter situations outside specifications.

Learning-specification mismatch: Gap between continuous neural embeddings and discrete symbolic logic is fundamental. No perfect bridge exists.

Scalability limits: As systems grow (millions of agents, billions of states), centralized specifications may become unmanageable.

B. Societal and Ethical Risks

Specification as control: ErisML could enable centralized control, with specifications written by the powerful and imposed on the vulnerable.

Complexity as opacity: If specifications are too complex to understand, they obscure rather than illuminate. May create false confidence.

Gaming and exploitation: Adversaries could craft malicious specifications that technically comply with norms but violate their spirit.

Power dynamics: Who writes specifications embeds values and power relations. Corporate-written specs encode corporate interests; government-written specs encode state priorities. We cannot solve this through better syntax—it requires transparency, contestation mechanisms, and accountability structures.

C. Environmental Costs

Formal verification, multi-agent simulation, and continuous monitoring are compute-intensive. At scale (smart cities, IoT), this could mean: - Billions of norm checks per day - Continuous verification as systems adapt - Significant energy consumption

We must weigh governance benefits against environmental costs. Prioritize high-stakes domains (healthcare, safety-critical) over convenience applications (smart lightbulbs).

X. Open Questions

Q1: Specification-Reality Gap: Can symbolic specifications ever fully capture learned model behavior? Or is there a fundamental representational mismatch?

Q2: Democratic Specification: How do we ensure specifications reflect diverse stakeholder values in public systems? Participatory design [34] provides methods, but pervasive systems blur stakeholder boundaries.

Q3: Equilibrium Concept: What is the right strategic equilibrium for human-AI teams under norms? Nash equilibria are chaotic; cooperative solutions require shared objectives.

Q4: Adaptation-Safety Trade-off: How do we maintain safety invariants across specification updates? Can we automatically propose specification changes based on failures?

Q5: Privacy-Transparency Paradox: Auditing requires transparency; privacy requires opacity. We cannot have perfect auditing and perfect privacy. Where should the balance be?

XI. Conclusion

As foundation models move from cloud to edge, from single-purpose to multi-agent, from controlled to pervasive environments, fragmented approaches to governance will fail. We risk accidents from misalignment, violations from ignorance, failures from brittleness, injustice from opacity, and erosion of trust.

ErisML proposes structured governance—not eliminating conflict (which is impossible in pluralistic systems) but making trade-offs explicit, enabling governance without tyranny, and supporting coexistence of agents with divergent goals.

This vision paper is an invitation, not a proclamation. The challenges are profound: - Fundamental gaps between symbolic specs and learned models - Undecidability for expressive fragments - Power dynamics in who writes specifications - Open questions about equilibrium concepts - Environmental costs of formal verification

But the alternative—continuing with prompts, rules, and ad-hoc code—is worse.

The path from vision to reality requires technical innovation, social negotiation, institutional change, and ongoing learning. We offer ErisML not as a finished solution but as a starting point—a design space, a set of challenges, and an invitation to collaborate.

The age of pervasive AI is here. The chaos is real. Will we face it with fragmented tools, or build common foundations for governance, safety, and trust?

Acknowledgments

The author thanks anonymous reviewers whose feedback shaped this vision. Discussions with colleagues in pervasive computing, AI safety, multi-agent systems, and HCI informed this work. Some editorial refinements were AI-assisted; all content was reviewed and approved by the author.

References

[1] M. Weiser, “The Computer for the 21st Century,” *Scientific American*, vol. 265, no. 3, pp. 94-104, 1991.

[2] M. Satyanarayanan, “Pervasive Computing: Vision and Challenges,” *IEEE Pervasive Computing*, vol. 1, no. 4, pp. 10-17, 2001.

[3] R. Bommasani et al., “On the Opportunities and Risks of Foundation Models,” *arXiv:2108.07258*, 2021.

[4] J. Achiam et al., “GPT-4 Technical Report,” *arXiv:2303.08774*, 2023.

[5] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*, Morgan Kaufmann, 2004.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018.

[7] M. Wooldridge, *An Introduction to MultiAgent Systems*, 2nd ed., Wiley, 2009.

[8] J. Carmo and A. J. I. Jones, “Deontic Logic and Contrary-to-Duties,” in *Handbook of Philosophical Logic*, vol. 8, pp. 265-343, 2002.

[9] D. Amodei et al., “Concrete Problems in AI Safety,” *arXiv:1606.06565*, 2016.

[10] L. Ouyang et al., “Training Language Models to Follow Instructions with Human Feedback,” *NeurIPS*, 2022.

[11] Y. Bai et al., “Constitutional AI: Harmlessness from AI Feedback,” *arXiv:2212.08073*, 2022.

[12] Anthropic, “Claude’s Constitution,” https://www.anthropic.com/index/claudes-constitution, 2023.

[13] D. Hadfield-Menell et al., “Inverse Reward Design,” *NeurIPS*, 2017.

[14] B. McMahan et al., “Communication-Efficient Learning of Deep Networks from Decentralized Data,” *AISTATS*, 2017.

[15] C. Dwork, “Differential Privacy,” *ICALP*, 2006.

[16] S. Amershi et al., “Guidelines for Human-AI Interaction,” *CHI*, 2019.

[17] P. Liang et al., “Holistic Evaluation of Language Models,” *arXiv:2211.09110*, 2022.

[18] D. Sculley et al., “Hidden Technical Debt in Machine Learning Systems,” *NeurIPS*, 2015.

[19] T. Gebru et al., “Datasheets for Datasets,” *Communications of the ACM*, vol. 64, no. 12, pp. 86-92, 2021.

[20] M. Mitchell et al., “Model Cards for Model Reporting,” *FAT**, 2019.

[21] N. Dalal et al., “Value-Sensitive Design and Information Systems,” in *The Handbook of Information and Computer Ethics*, Wiley, 2008.

[22] D. Ha and J. Schmidhuber, “World Models,” *arXiv:1803.10122*, 2018.

[23] D. Hafner et al., “Mastering Atari with Discrete World Models,” *ICLR*, 2021.

[24] A. Rai, “Explainable AI: From Black Box to Glass Box,” *Journal of the Academy of Marketing Science*, vol. 48, pp. 137-141, 2020.

[25] S. Russell, *Human Compatible: Artificial Intelligence and the Problem of Control*, Viking, 2019.

[26] V. Dignum, *Responsible Artificial Intelligence: How to Develop and Use AI in a Responsible Way*, Springer, 2019.

[27] I. Rahwan et al., “Machine Behaviour,” *Nature*, vol. 568, pp. 477-486, 2019.

[28] S. Yao et al., “ReAct: Synergizing Reasoning and Acting in Language Models,” *ICLR*, 2023.

[29] N. Shinn et al., “Reflexion: Language Agents with Verbal Reinforcement,” *arXiv:2303.11366*, 2023.

[30] G. Wang et al., “Voyager: An Open-Ended Embodied Agent with LLMs,” *arXiv:2305.16291*, 2023.

[31] F. Dignum et al., “Meeting the Deadline: Why, When and How,” in *Formal Approaches to Agent-Based Systems*, Springer, 2004.

[32] M. Sergot, “A Computational Theory of Normative Positions,” *ACM Trans. on Computational Logic*, vol. 2, no. 4, pp. 581-622, 2001.

[33] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason*, Wiley, 2007.

[34] J. Simonsen and T. Robertson, *Routledge International Handbook of Participatory Design*, Routledge, 2012.

[35] O. Maler and D. Nickovic, “Monitoring Temporal Properties of Continuous Signals,” in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Springer, 2004.

[36] A. Platzer, “Logical Foundations of Cyber-Physical Systems,” Springer, 2018.

[37] Y. Bengio et al., “A Meta-Transfer Objective for Learning to Disentangle Causal Mechanisms,” *ICLR*, 2020.

[38] D. Abel et al., “Agent-Agnostic Human-in-the-Loop Reinforcement Learning,” *arXiv:1701.04079*, 2017.

[39] C. Finn et al., “Model-Agnostic Meta-Learning for Fast Adaptation,” *ICML*, 2017.

[40] European Commission, “Proposal for a Regulation on Artificial Intelligence (AI Act),” 2021.

Appendix A: ErisML Grammar (Excerpt)

```
<model> ::= <environment> <agent>* <norms>? <dynamics>? <validation>?


<environment> ::= "environment" <id> "{" <env-body> "}"

<env-body> ::= <objects> <state> <observations>? <dynamics>


<agent> ::= "agent" <id> "{" <agent-body> "}"

<agent-body> ::= <capabilities> <beliefs> <intents> <constraints>?
```

```
<norms> ::= "norms" <id> "{" <norm-rule>* "}"
<norm-rule> ::= <permission> | <prohibition> | <obligation> | <sanction>

<permission> ::= "permission:" "{" <action-expr> <condition>? "}" ";"
<prohibition> ::= "prohibition:" "{" <action-expr> <condition>? "}" ";"
<obligation> ::= "obligation:" "{" <action-expr> <deadline>? "}" ";"

<intents> ::= "intents:" <intent-expr> ";"
<intent-expr> ::= <weighted> | <lexicographic> | <constrained>

<condition> ::= "when" <expr> | "unless" <expr>
```

Appendix B: Compilation Example (PDDL)

ErisML Input:

```
environment SimpleRoom {
  objects: Robot, Door;
  state:
    door.status: {open, closed};
    robot.location: {inside, outside};
  dynamics:
    open_door() ~> door.status = open;
    enter() ~> robot.location = inside if door.status == open;
}

agent Robot {
  intents: goal robot.location == inside;
}
```

```
norms Safety {
  prohibition: Robot.enter() unless door.status == open;
}
```

PDDL Output:

```
(define (domain simple-room)
  (:requirements :strips :typing)
  (:predicates (door-open) (door-closed) (robot-inside) (robot-outside))

  (:action open-door
    :parameters ()
    :precondition (door-closed)
    :effect (and (door-open) (not (door-closed))))

  (:action enter
    :parameters ()
    :precondition (and (robot-outside) (door-open))  ; Norm enforced
    :effect (and (robot-inside) (not (robot-outside)))))

(define (problem reach-inside)
  (:domain simple-room)
  (:init (door-closed) (robot-outside))
  (:goal (robot-inside)))
```

Note: The prohibition from `norms Safety` is compiled into the precondition of the `enter` action. This ensures the planner never generates norm-violating plans, but provenance information (that this precondition comes from a norm) is lost in translation.