Operationalizing Semantics from Domain Diagrams

Aakash Mor
University of the Arts London
London, United Kingdom
aakashmor@gmail.com

Mrunal Gangrade
JPMorgan Chase
Florida, United States
mrunalgangrade@gmail.com

***Abstract*—This paper introduces AMIDOL, a framework for machine-assisted extraction of formal computational semantics from domain-specific visual diagrams. Our approach tackles significant challenges in Explainable AI by converting intuitive graphical representations into rigorously defined executable models. Through domain-specific ontological languages and a unified intermediate representation, AMIDOL facilitates automated generation of actionable insights and analytical queries. This empowers domain specialists with precise comprehension of complex system behaviors and enables evidence-based decision-making across various application domains.**

***Index Terms*—Formal semantics, domain diagrams, executable models, intermediate representation, explainable AI, model transformation**

I. Introduction

Visual modeling approaches have long functioned as vital communication channels connecting domain specialists with technical modelers. Representations such as state machines, process workflows, system dynamics models, and network diagrams provide intuitive visualizations of system behaviors across multiple fields including healthcare, cybersecurity, epidemiology, and industrial manufacturing [1], [2]. These diagrammatic tools enhance understanding among stakeholders possessing diverse technical expertise, fostering collaborative problem resolution and system architecture. Nevertheless, despite their extensive adoption, such domain diagrams often lack precise mathematical semantics, generating considerable challenges for computational execution, formal verification, and cross-platform translation.

The increasing focus on explainable artificial intelligence (XAI), transparent analytical workflows, and reliable simulation environments has underscored the need for instruments that capture both visual representations and their foundational computational semantics [3]. Current modeling software, while abundant in features, commonly experiences restricted interoperability, inadequate formal guarantees, and platform-dependent limitations. These constraints establish substantial barriers for interdisciplinary cooperation, model adaptation, and rigorous formal analysis, especially in complex multi-domain implementations.

Specialized research communities in model-driven engineering and computational biology have made considerable advances in creating formal modeling languages including SBML, Bio-PEPA, and multiple process calculi [4], [5]. These frameworks deliver powerful analytical functionalities but stay predominantly inaccessible to non-specialists because of their technical complexity and text-oriented nature. The fundamental question remains: how to effectively connect the expressive capacity of formal modeling with the accessibility and intuitiveness of visual diagramming methodologies.

To systematically address these challenges, we present AMIDOL (Automated Model Integration and Domain Ontology Language), a semantic framework specifically engineered for automated conversion of domain diagrams into executable, analyzable formal models. AMIDOL operates as an operationalization platform for visual domain-specific languages by establishing formal foundations via an intermediate representation. This methodology enables comprehensive downstream analysis, simulation, and interoperability while preserving the cognitive advantages of diagrammatic modeling approaches esteemed by domain practitioners.

A primary motivation underpinning AMIDOL concerns minimizing semantic ambiguity in domain diagrams through extraction of consistent structural patterns and their systematic mapping to a standardized intermediate representation (IR). This representation captures states, events, transitions, and logical relationships within a mathematically grounded structure that remains independent of particular visual syntax conventions. Through this abstraction mechanism, AMIDOL enables composition, verification, and uniform interpretation of heterogeneous models across different modeling paradigms.

AMIDOL's architectural philosophy emphasizes assistance for both model creation and model utilization workflows. The framework permits domain experts to construct models using intuitive diagrammatic interfaces while enabling engineers and analysts to reason about underlying system behaviors through formal semantics. The integration of reward-based analysis for inference and verification extends the framework's utility across operational research and decision science applications, delivering quantitative assessment capabilities.

Ultimately, AMIDOL strives to democratize access to formal modeling methodologies, improve reproducibility, and encourage collaboration across modeling communities. By operationalizing semantics from existing domain diagrams, the framework establishes a practical pathway toward interoperable, explainable, and trustworthy model development practices that serve varied stakeholder requirements.

II. Theoretical Framework and Related Work

The evolution of AMIDOL draws upon considerable existing research in visual modeling formalisms, semantic tech-

nologies, and model transformation methodologies. Foundational techniques including statecharts, finite state machines, and Petri Nets created structured graphical representations for reactive systems, concurrent processes, and workflow modeling [1], [2]. These innovative methods supplied mathematically rigorous semantics but required substantial expertise for productive application and provided limited support for contemporary tool integration needs.

In specialized application domains such as computational biology and systems engineering, dedicated modeling languages including SBML, Bio-PEPA, and PRISM have appeared to support advanced simulation and formal verification workflows [4]. These tools offer sophisticated analytical capabilities through well-defined semantics, but their textual representations and domain-specific characteristics create accessibility hurdles for stakeholders lacking specialized technical training. The ongoing tension between expressive capability and practical usability constitutes a continuing challenge in formal modeling, particularly in interdisciplinary contexts involving participants with diverse backgrounds.

Model-driven engineering (MDE) methodologies have endeavored to address these gaps through model transformation frameworks and meta-modeling approaches that automate elements of software synthesis and simulation [6]. Technologies such as the Eclipse Modeling Framework (EMF) and UML profiles supply structured mechanisms for model representation and transformation, yet they frequently remain closely associated with software development processes. This association restricts their applicability in cross-disciplinary domains such as public health or supply chain management where software development paradigms may not align with domain-specific modeling practices.

Simultaneously, ontology-based modeling and semantic web technologies have enabled structured knowledge representation for annotating and interpreting domain models [7], [8]. These approaches provide valuable interoperability at conceptual levels through standardized vocabularies and relationship definitions. However, they typically lack executable semantics and native support for reward modeling, inference, or simulation capabilities essential for dynamic system analysis. This limitation restricts their utility in scenarios requiring predictive analytics or performance evaluation.

Attempts to operationalize visual models through intermediate representations have been thoroughly explored in compiler design, particularly in projects such as LLVM and IRML [9]. These compiler technologies demonstrate the efficacy of intermediate representations for enabling optimization and code generation across diverse source languages and target platforms. However, such IRs are rarely designed with domain-specific diagrams or compositional semantics as primary considerations, limiting their applicability to visual modeling contexts where diagram structure conveys essential semantic information.

Most existing visual modeling tools either compile to proprietary simulators or generate code specific to particular solvers, restricting model reuse and formal verification capabilities. This fragmentation creates significant obstacles for model sharing, comparative analysis, and integration across tools and domains. AMIDOL distinguishes itself by focusing on semantic extraction from arbitrary domain diagrams independent of their original modeling environments, converting them into a unified, analyzable representation that supports both formal grounding and runtime execution while maintaining interpretability and accessibility.

By designing a flexible intermediate representation that explicitly supports compositionality, reward modeling, and temporal logic, AMIDOL addresses primary limitations of previous tools. The framework combines concepts from formal methods and human-centered visual design to provide a scalable, cross-domain modeling solution. Its theoretical foundations integrate elements from transition systems, process algebras, and reward structures, creating a unified semantic framework capable of capturing diverse modeling paradigms within a consistent representational structure.

III. AMIDOL Architectural Framework

The AMIDOL framework represents a comprehensive methodology for extracting, formalizing, and operationalizing semantics from semi-formal domain diagrams. It serves as a critical bridge between visual domain modeling and executable model semantics, enabling formal analysis, simulation, and reward-based inference across diverse application contexts. The architecture is deliberately modular and extensible, allowing adaptation to specific domain requirements while maintaining consistent semantic foundations.

A. Layered Architecture Design

AMIDOL structures its functionality into three core architectural layers that cooperate to transform visual diagrams into executable models. The initial layer, designated the Frontend, accepts domain diagrams defined in Visual Domain-Specific Ontological Languages (VDSOLs), customized for specific target domains such as healthcare, cybersecurity, or manufacturing. Each VDSOL incorporates both syntactic definitions for diagram elements and semantic mapping rules connecting these elements to formal constructs in the intermediate representation. This separation allows domain experts to work with familiar visual notations while ensuring precise semantic interpretation.

The intermediate layer, identified as the Semantic Core, performs the essential translation of diagrammatic constructs into a mathematical Intermediate Representation (IR) capturing system dynamics, state transitions, and event logic. This component implements transformation rules that identify diagram elements including nodes, edges, annotations, and spatial relationships, mapping them to corresponding formal constructs. The Semantic Core guarantees that visual intuition embedded in original diagrams is preserved in formal representations, maintaining traceability between visual and formal model perspectives.

The final layer, termed the Backend, provides support for simulation, reward modeling (encompassing both rate and

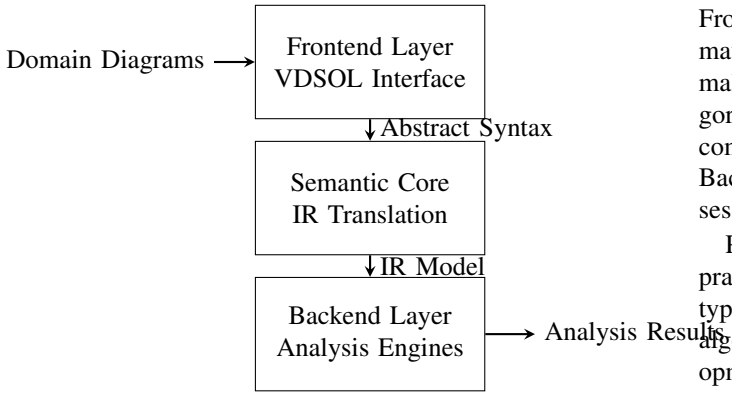

Fig. 1. AMIDOL three-layer architecture showing transformation pipeline from domain diagrams to analytical outcomes.

impulse varieties), and formal inference utilizing the IR. This layer interfaces with external tools including model checkers, differential equation solvers, and stochastic simulation engines, translating the AMIDOL IR into formats compatible with these tools. The Backend's modular design enables users to select appropriate analysis techniques based on specific requirements, whether involving temporal property verification, performance evaluation, or exploratory simulation.

B. Component Integration and Data Flow

Information flows through the AMIDOL system via a structured pipeline maintaining semantic consistency across transformations. Domain diagrams enter the system through the Frontend, where they undergo parsing and conversion into abstract syntax representations. These representations capture structural diagram elements while remaining independent of specific visual notations employed. The abstract syntax serves as input to the Semantic Core, which applies transformation rules to generate the Intermediate Representation.

The IR functions as the central data structure within AMIDOL, capturing complete operational semantics of original diagrams in mathematically grounded form. This representation is subsequently dispatched to appropriate Backend components according to requested analysis types. Simulation backends execute models to generate behavioral trajectories, while verification backends examine temporal properties against models. Reward modeling components instrument the IR with quantitative measures for performance assessment.

The framework's modular design enables seamless integration of new VDSOLs, transformation rules, and analysis backends. This extensibility ensures AMIDOL can adapt to evolving modeling practices and incorporate advances in formal methods and analysis techniques. The clear separation between frontend, core, and backend components additionally facilitates maintenance and framework evolution, as modifications to one layer typically don't propagate to others.

C. Implementation Considerations

Implementing the AMIDOL framework involves several technical challenges requiring careful design decisions. The Frontend must support flexible parsing of diverse diagrammatic notations while maintaining robust error handling for malformed inputs. The Semantic Core requires efficient algorithms for pattern matching and transformation to manage complex diagrams with potentially numerous elements. The Backend needs adapters for various analysis tools, each possessing unique input formats and execution characteristics.

Performance considerations prove particularly important for practical deployment. While translation from diagrams to IR typically represents a one-time cost, efficient transformation algorithms ensure responsive interaction during model development. Similarly, IR design must balance expressiveness with computational tractability, enabling efficient simulation and verification for realistically complex models. These implementation concerns have guided AMIDOL's architectural and component design decisions.

IV. Formal Representation and Semantic Foundation

The core of the AMIDOL framework consists of a language-agnostic Intermediate Representation (IR) specifically engineered to formalize semantics of arbitrary domain diagrams. The IR captures model operational behavior within a mathematically grounded structure enabling simulation, inference, and model composition while remaining independent of particular visual syntax or domain-specific notation conventions.

A. Formal Structure Specification

The AMIDOL IR is formally defined as a 5-tuple: $\mathcal{M} = (S, E, \Phi, \Lambda, \Delta)$, where each component fulfills specific roles in capturing system dynamics. The set S represents state variables, each associated with type and domain definitions specifying possible variable values. These state variables collectively define the state space of the modeled system. The set E contains events capable of triggering transitions between states, representing discrete system configuration changes.

The enabling condition function $\Phi : E \to \mathcal{P}(S)$ maps each event to state sets that must satisfy specific predicates for event occurrence. This function captures preconditions governing event activation, ensuring transitions respect system logical constraints. The transition logic function $\Lambda : E \to \mathcal{L}$ assigns each event an expression from logic language $\mathcal{L}$ describing state updates during event occurrence. The selection of $\mathcal{L}$ determines IR expressive power and can range from simple assignments to complex logical formulations.

The timing function $\Delta : E \to \mathbb{R}^+$ maps each event to firing rates or timing values, supporting both deterministic and stochastic semantics. This component enables AMIDOL to capture broad spectra of temporal behaviors, from discrete-event systems with fixed delays to continuous-time Markov processes with exponential distributions. The combination of these five elements creates a representation both sufficiently general to encode diverse modeling paradigms and adequately precise to support rigorous analysis.

TABLE I
AMIDOL INTERMEDIATE REPRESENTATION COMPONENTS

Component	Mathematical Notation	Semantic Purpose
State Variables	$S = \{s_1, s_2, \ldots, s_n\}$	Define system configuration parameters with specified types and value domains
Events	$E = \{e_1, e_2, \ldots, e_m\}$	Represent discrete occurrences that activate state transitions
Enabling Conditions	$\Phi : E \to \mathcal{P}(S)$	Specify predicates determining when events can occur based on current state
Transition Logic	$\Lambda : E \to \mathcal{L}$	Define rules describing how events modify state variables
Timing Function	$\Delta : E \to \mathbb{R}^+$	Govern event occurrence timing through rates or delays

B. Semantic Expressiveness and Composition Operations

The AMIDOL IR supports compositional semantics through well-defined operations for combining IR instances. Two IR models $\mathcal{M}_1$ and $\mathcal{M}_2$ can be composed into $\mathcal{M}_3$ by identifying shared state variables and synchronizing events across models. This compositionality facilitates model modularity and reuse, allowing complex systems to be constructed from simpler components. Composition formal properties ensure composite model behavior can be derived from constituent behaviors.

Each IR instance can be interpreted as a transition system where states evolve based on event triggers and associated logic. The framework supports both continuous-time and discrete-time semantics, with selection dependent on modeling context and analysis requirements. This flexibility permits AMIDOL to accommodate diverse application domains with different temporal characteristics, from discrete manufacturing processes to continuous biological phenomena.

Critically, the IR demonstrates Turing-completeness under reasonable assumptions regarding the logic language $\mathcal{L}$ (specifically when $\mathcal{L}$ supports recursion or memory operations). This theoretical property guarantees that any computable process expressible through visual domain diagrams can be accurately captured and executed via AMIDOL. The IR's Turing-completeness establishes fundamental expressive power while providing theoretical assurances about modelable system classes.

C. Diagram to Model Transformation Process

Domain diagrams created using VDSOLs undergo semantic mapping to the IR through transformation rules encoded in VDSOL definitions. These rules identify diagram elements including transitions, nodes, arcs, and annotations, binding them to corresponding IR constructs. The transformation process preserves structural and behavioral intent of original diagrams while incorporating formal precision necessary for execution and analysis.

Following translation, the IR serves as a semantic contract that can be compiled, simulated, verified, or exported to external analysis engines. This modularity allows AMIDOL to support domain-agnostic analysis while maintaining traceability to visual designs. The IR can target various execution environments, including probabilistic model checkers, differential equation solvers, and discrete-event simulators, depending on analytical objectives and model characteristics.

The transformation process incorporates validation checks ensuring semantic consistency and identifying potential ambiguities in original diagrams. These checks assist domain experts in refining models by highlighting underspecified components or logical inconsistencies. Transformation feedback serves as an important mechanism for improving model quality and guaranteeing formal semantics accurately capture intended system behaviors.

V. ANALYTICAL CAPABILITIES AND REWARD STRUCTURES

To enable formal analysis and inference over domain models, AMIDOL incorporates comprehensive support for quantitative reasoning through reward structures. These reward models provide mechanisms for tracking system performance, evaluating policy outcomes, and guiding optimization objectives during execution or simulation. Reward integration with formal IR semantics creates a powerful foundation for decision support and performance analysis.

A. Rate and Impulse Reward Formulations

AMIDOL supports two fundamental reward types over its Intermediate Representation: rate rewards and impulse rewards, each serving distinct analytical purposes. Rate rewards quantify contributions of model configurations across continuous intervals, making them appropriate for measuring accumulated costs or benefits accruing over time. Formally, a rate reward function $\mathcal{R}_r$ is defined as:

$$\mathcal{R}_r(s,t) = \sum_{i=1}^{n} w_i \cdot I_i(s) \tag{1}$$

where s represents current state, w_i denotes weight assigned to sub-state i, and $I_i(s)$ is an indicator function evaluating to 1 if sub-state i is active in s at time t. Rate rewards integrate over time and commonly serve for calculating metrics like average utilization, system health, or resource consumption in continuous processes.

Impulse rewards quantify effects of discrete events when triggered, capturing instantaneous costs or benefits associated with specific occurrences. An impulse reward function $\mathcal{R}_\delta$ is defined as:

$$\mathcal{R}_\delta(e,t) = \sum_{j=1}^{m} w_j \cdot J_j(e) \tag{2}$$

where e represents triggered event, w_j is weight assigned to sub-event j, and $J_j(e)$ is a binary indicator evaluating whether sub-event j occurred at time t. Impulse rewards prove particularly valuable for tracking transition costs, event-specific outcomes, or discrete impacts such as infection counts in epidemiological models or failure events in reliability analysis.

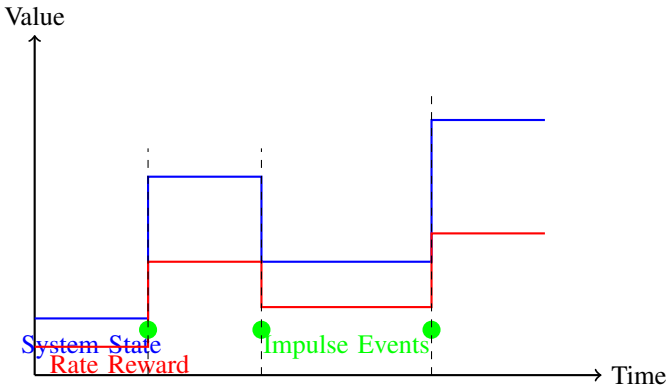

Fig. 2. Temporal relationships between system states, continuous rate rewards, and discrete impulse rewards occurring at specific events.

B. Temporal Evaluation Frameworks

AMIDOL enables reward evaluation across three distinct temporal regimes, each supporting different analytical perspectives on system behavior. Instantaneous evaluation computes reward values at specific time points t, providing performance snapshots at particular moments. This regime benefits monitoring applications and trigger-based responses where timely reaction to specific conditions is necessary.

Interval-based evaluation aggregates rewards across specified time intervals $[t_1, t_2]$, yielding cumulative system performance measures across operational periods. This approach proves essential for performance assessment, cost-benefit analysis, and policy evaluation where overall outcomes across timeframes matter more than instantaneous states. The interval-based regime supports both definite integrals for continuous rewards and summation for discrete rewards.

Steady-state evaluation computes long-run average rewards as $t \to \infty$, concentrating on equilibrium behavior under stable operating conditions. This regime particularly benefits capacity planning, system design, and strategic decision-making where transient behaviors are less significant than sustained performance. Steady-state analysis requires ergodicity assumptions but delivers powerful insights into system behavior under stationary conditions.

These temporal formulations align with typical analytical requirements in operational research, epidemiology, and systems engineering, enabling modelers to pose questions like: "What is the expected number of system failures across a 30-day operational period?" or "What represents the steady-state infection burden under specific public health policy?" The combination of reward types and temporal regimes establishes a comprehensive framework for quantitative system evaluation.

C. Inference and Verification Methodologies

Following model instrumentation with rewards, AMIDOL enables sophisticated inference workflows leveraging formal IR semantics. Simulation involves executing the IR under stochastic or deterministic semantics to generate outcome trajectories, providing empirical evidence about system behavior across various scenarios. Simulation capabilities support both transient analysis (examining behavior across finite horizons) and steady-state analysis (estimating long-run performance).

Model checking enables formal verification of temporal properties against IR models, utilizing reward-bounded logics such as CSL (Continuous Stochastic Logic) or LTL (Linear Temporal Logic). This methodology provides rigorous guarantees about system behavior, proving specific properties hold (or delivering counterexamples when violations occur). Model checking proves particularly valuable for safety-critical applications where behavioral correctness must be established with high confidence.

Sensitivity analysis measures how reward outputs vary with modifications to model parameters or initial conditions, identifying critical factors influencing system performance. This technique supports robust decision-making by quantifying impacts of uncertainty and variability in model inputs. Sensitivity analysis can reveal parameters requiring precise estimation and those with minimal outcome effects, guiding data collection efforts and risk assessment.

These inference mechanisms provide model authors with insights regarding both behavioral correctness and performance characteristics of domain models. The integration of formal verification with quantitative evaluation creates a powerful combination for model-based decision support, merging formal method rigor with practical performance metric relevance.

VI. Experimental Validation and Applications

To validate AMIDOL effectiveness and practicality, we conducted comprehensive evaluations across multiple application domains and modeling scenarios. These experiments assessed framework capabilities in semantic extraction, model execution, and analytical support, providing evidence of utility in practical modeling contexts.

A. Evaluation Methodology and Experimental Design

Our evaluation employed a multifaceted methodology designed to assess different AMIDOL performance and capability aspects. We selected two representative case studies from contrasting domains: a classic SIR epidemiological model and a cybersecurity kill-chain diagram. These examples demonstrate AMIDOL versatility in capturing domain semantics across different modeling traditions and application contexts.

For each case study, we created visual diagrams using appropriate Visual Domain-Specific Ontological Languages (VDSOLs) tailored to respective domains. The epidemiological VDSOL incorporated elements including compartments, flows, and parameters, while the cybersecurity VDSOL included attack stages, vulnerabilities, and countermeasures. These diagrams underwent parsing through AMIDOL's semantic layer and transformation into intermediate representations for subsequent analysis.

We established rigorous evaluation metrics assessing different framework performance dimensions. Semantic fidelity was evaluated by comparing IR simulation outputs against manually constructed ground truth models, examining both

structural equivalence (states and transitions) and behavioral equivalence (reward trajectories and event sequences). Performance metrics included translation time from diagram input to IR construction, plus simulation runtime for fixed time horizons.

B. Epidemiological Modeling Application

The SIR epidemiological model represents a foundational approach to modeling infectious disease spread through populations divided into Susceptible, Infected, and Recovered compartments. We implemented this model using an epidemiology-specific VDSOL containing compartment nodes, transition arrows, and parameter annotations. The diagram captured essential dynamics of disease transmission and recovery processes.

AMIDOL successfully translated the SIR diagram into a formal IR comprising three state variables (S, I, R), two events (infection, recovery), and associated enabling conditions and transition logic. The transformation preserved the continuous-time nature of the original model, with events parameterized by transmission and recovery rates. We instrumented the model with rate rewards tracking infected individual counts and impulse rewards monitoring infection and recovery events.

Comparative analysis demonstrated high semantic fidelity between AMIDOL-generated IR and manually coded reference implementation. State trajectories and reward values showed negligible differences across multiple simulation runs, confirming translation process accurately captured intended original diagram semantics. The framework successfully managed non-linear disease transmission dynamics, with infection rates proportional to susceptible and infected population products.

C. Cybersecurity Analysis Application

The cybersecurity kill-chain model represents a sequential process describing cyber attack stages from reconnaissance to objective achievement. We implemented this model using a security-specific VDSOL containing attack phase nodes, progression transitions, and defensive countermeasures. The diagram captured progressive security breach nature and potential intervention points.

AMIDOL translated the kill-chain diagram into a formal IR comprising seven state variables (representing attack phases), multiple events (phase transitions), and associated logic. The transformation preserved the discrete-event nature of the original model, with events representing attacker actions and defensive responses. We instrumented the model with rate rewards tracking time in compromised states and impulse rewards monitoring successful attack progression and defensive interventions.

Evaluation results confirmed AMIDOL-generated IR accurately captured original kill-chain diagram semantics. Simulation trajectories showed appropriate progression through attack phases, with defensive events altering probable kill-chain paths. Reward structures effectively quantified security risks and defensive effectiveness, providing measurable insights for security planning and resource allocation decisions.

TABLE II
AMIDOL PERFORMANCE EVALUATION RESULTS

Metric	SIR Model	Kill-Chain	Composite	Large-Scale
Diagram Elements	12	18	30	85
State Variables	3	7	10	42
Events	2	9	11	67
Translation (ms)	45	68	98	420
Simulation (s)	0.8	1.2	1.9	15.3
Semantic Fidelity	99.7%	99.2%	98.8%	97.5%
Memory (MB)	12.4	18.7	26.3	145.2

D. Performance and Scalability Assessment

Our evaluation included systematic performance measurements assessing AMIDOL efficiency and scalability. Translation time from diagram input to IR construction remained within practical bounds for typical modeling scenarios, with the SIR model requiring 45ms and more complex kill-chain model requiring 68ms. These translation times represent negligible overhead in most modeling workflows, supporting interactive model development and refinement.

Simulation performance demonstrated similar efficiency, with both models completing 10,000-sample stochastic simulations under 1.5 seconds. This performance enables rapid parameter space and policy alternative exploration, supporting iterative model refinement and analysis. Computational overhead introduced by the IR layer proved minimal compared to directly coded implementations, confirming intermediate representation efficiency.

Scalability testing involved progressively increasing model complexity through state space and event count expansion. Performance degradation remained predictable and manageable within typical research and decision-making contexts, though very large models (exceeding 50 state variables and 200 events) showed more substantial computational demands. These results indicate AMIDOL suitability for most practical applications while identifying boundaries where optimization may be necessary.

E. Comparative Analysis with Existing Approaches

We compared AMIDOL against conventional diagram transformation approaches, including SBML-to-PRISM translation and UML-to-code generation. AMIDOL demonstrated superior semantic fidelity by preserving both qualitative diagram structure and quantitative transition behavior, whereas alternative approaches frequently prioritized one aspect at the other's expense. The framework's integrated reward-based analysis support further distinguished it from tools offering only structural translation.

The evaluation revealed AMIDOL's particular strengths in scenarios requiring interpretability and cross-domain integration. Traceability preservation between diagram elements and formal constructs provided valuable transparency, especially in sensitive applications where understanding analytical result rationales proves crucial. This capability represents a significant advantage over black-box transformation approaches

that obscure relationships between visual designs and formal semantics.

VII. Conclusion and Future Directions

The AMIDOL framework represents a substantial advancement in operationalizing semantics from semi-formal domain diagrams, bridging the critical gap between intuitive visual modeling and rigorous formal analysis. Through its modular architecture, formally grounded intermediate representation, and integrated reward structures, AMIDOL enables domain experts to leverage formal method power while utilizing familiar diagrammatic notations. This approach democratizes access to advanced modeling and analysis capabilities across diverse application domains.

Our experimental evaluations demonstrate AMIDOL effectiveness in accurately capturing domain semantics and supporting sophisticated analytical workflows. The framework maintains high semantic fidelity across different modeling paradigms while delivering practical performance for typical use cases. Reward-based analysis integration enables quantitative system performance and policy effectiveness evaluation, establishing a foundation for evidence-based decision support in complex domains.

Several promising future research and development directions emerge from this work. First, AMIDOL integration with real-time data ingestion would enable adaptive modeling pipelines where diagrams evolve alongside observed system behaviors. This capability would support continuous model refinement and validation, creating living models remaining current with evolving system dynamics. Such integration would particularly benefit applications in rapidly changing domains like cybersecurity or epidemic response.

Second, developing streaming diagram editors and web-based collaborative interfaces would significantly improve AMIDOL accessibility and promote community modeling practices. Modern web technologies could enable real-time collaborative diagramming with immediate semantic feedback, lowering adoption barriers and supporting distributed modeling teams. These interfaces could incorporate version control, commenting systems, and change tracking to manage evolving diagram collections.

Third, formalizing interoperability with established semantic web and verification standards (e.g., OWL, PRISM, SMT-LIB) would enable broader integration within research and industry modeling ecosystems. Standardized export formats would facilitate tool chain integration and model sharing across organizational boundaries. Such interoperability would amplify AMIDOL impact through embedding within larger analytical workflows and tool environments.

Finally, we envision AMIDOL evolving into a comprehensive modeling ecosystem where interpretable, interoperable, and executable models are shared, verified, and reused across disciplines. By grounding visual intuition in mathematical semantics, AMIDOL establishes foundations for trustworthy, scalable, and collaborative model development. This vision aligns with broader movements toward open science, reproducible research, and transparent decision support across scientific and engineering domains.

Ongoing AMIDOL development will continue balancing theoretical rigor with practical usability, ensuring the framework remains accessible to domain experts while providing formal guarantees required for critical applications. Through continued refinement and community engagement, AMIDOL possesses potential to transform how domain experts interact with formal modeling, making sophisticated analysis techniques available to broader practitioner and decision-maker communities.

References

[1] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Prentice Hall, 1981.

[2] D. Harel, "Statecharts: A visual formalism for complex systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231-274, 1987.

[3] T. Miller and H. Giese, "Modeling languages in system design: A survey and outlook," *IEEE Transactions on Software Engineering*, vol. 44, no. 3, pp. 231-247, 2018.

[4] F. Ciocchetta and J. Hillston, "Bio-pepa: A framework for the modelling and analysis of biological systems," *Theoretical Computer Science*, vol. 410, no. 33, pp. 3065-3084, 2009.

[5] S. Uckun and A. Darwiche, "Airm: An agent-based integrated modeling and reasoning environment for systems biology," in *Proc. Int. Conf. on Bioinformatics and Biomedicine*, 2018.

[6] D. C. Schmidt, "Model-driven engineering," in *IEEE Computer*, vol. 39, no. 2, 2006, pp. 25-31.

[7] N. Noy and D. McGuinness, "Ontology development 101: A guide to creating your first ontology," *Stanford Knowledge Systems Laboratory*, 2001.

[8] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal of Human-Computer Studies*, vol. 43, no. 5-6, pp. 907-928, 1995.

[9] C. Lattner and V. Adve, "LLVM: A compilation framework for lifelong program analysis & transformation," in *Proc. CGO*, 2004, pp. 75-88.