

CAML: A Canonical Adventure Modeling Language for Tabletop Roleplaying Games

David R. Koepsell

January 5, 2026

Abstract

Tabletop roleplaying games (TTRPGs) occupy a distinctive space between narrative art, rule-based systems, and improvisational social play. Despite the increasing digitization of tabletop play and the proliferation of virtual tabletop platforms, adventures are still authored and distributed primarily as linear prose documents, limiting reuse, formal analysis, automation, and interoperability. This paper introduces the Canonical Adventure Modeling Language (CAML), a domain-specific modeling language designed to represent tabletop adventures as structured possibility spaces rather than scripted narratives. CAML provides a formal, machine-readable representation of locations, agents, items, encounters, conditions, and outcomes, enabling modular authoring, safe recombination, visualization, and export to diverse play environments. We argue that CAML functions as a bridge between narrative theory and computation by preserving human-authored meaning while enabling algorithmic reasoning over narrative structure. The paper situates CAML within existing work on interactive narrative, modeling languages, and game systems, and outlines its implications for tooling, procedural generation, and future research.

1 Introduction

Tabletop roleplaying games (TTRPGs) such as *Dungeons & Dragons* have long relied on written adventure modules as their primary medium of design, dissemination, and play preparation. These modules typically take the form of linear documents composed of descriptive prose, stat blocks, and maps. While effective for human consumption, such representations obscure the underlying structure of play: the conditional availability of encounters, the dependence of future events on player action, and the branching evolution of game state.

As tabletop play increasingly intersects with digital tooling, virtual tabletops, and artificial intelligence systems, the limitations of prose-centric adventure design become more pronounced. Linear documents are difficult to remix safely, impossible to analyze algorithmically without ad hoc interpretation, and tightly coupled to specific rule systems and platforms. This creates friction for designers, game masters, tool builders, and researchers alike.

This paper proposes an alternative representation: the Canonical Adventure Modeling Language (CAML). CAML is a domain-specific modeling language for tabletop adventures that treats an

adventure not as a story that will happen, but as a structured space of what *could* happen, subject to conditions and consequences. CAML is designed to be system-agnostic at its core, extensible to specific rulesets, and suitable as a canonical source for visualization, export, simulation, and analysis.

Availability. The Canonical Adventure Modeling Language (CAML) is released as an open-source project. The reference implementation, schemas, examples, and supporting tools are available at <https://github.com/dkoepsell/CAML5e>.

2 Background and Motivation

2.1 Adventures as Possibility Spaces

Narrative theorists have long distinguished between stories as fixed sequences of events and narratives as structures that support interpretation, agency, and contingency. In tabletop roleplaying games, this distinction is operational rather than theoretical. An adventure module does not determine what will occur; rather, it constrains and enables a range of possible developments shaped by player choice, chance, and interpretation.

Despite this, most published adventures are written as if they were linear stories with optional branches. This obscures the fact that many encounters may never occur, that certain outcomes are mutually exclusive, and that the same adventure can unfold in radically different ways across playthroughs.

CAML begins from the premise that an adventure is best understood as a *possibility space*: a structured set of potential events whose realization depends on evolving game state.

2.2 Limits of Existing Tools

Existing digital tools for tabletop play tend to fall into three categories:

- **Narrative editors** (e.g., interactive fiction tools), which emphasize branching stories but are poorly suited to tabletop rules and improvisation.
- **Worldbuilding and campaign managers**, which store notes and metadata but do not model conditional logic or outcomes.
- **Virtual tabletop platforms**, which focus on execution and automation rather than canonical representation.

What is notably absent is a shared, open, machine-readable representation of adventure structure itself. CAML is intended to fill this gap.

3 Design Goals

CAML was designed with the following goals:

1. **System agnosticism:** Core adventure structure should not depend on a specific ruleset.
2. **Explicit conditionality:** The conditions under which events may occur should be first-class elements.
3. **Modularity and reuse:** Components such as locations, encounters, and non-player characters should be independently reusable.
4. **Human readability:** Authors should be able to read and write CAML artifacts without specialized tools.
5. **Computational tractability:** Adventures should be analyzable, visualizable, and transformable by software.

These goals position CAML as a modeling language rather than a narrative scripting language.

4 The CAML Model

4.1 Core Entities

At its core, CAML defines a small set of entity types common to tabletop play:

- **Locations:** Places where events may occur.
- **Agents:** Player characters and non-player characters.
- **Items:** Artifacts that may affect state or unlock possibilities.
- **Encounters:** Potential events or scenes that may be realized.

These entities are represented as independent, addressable components.

4.2 Encounters, Gates, and Outcomes

The central abstraction in CAML is the *Encounter*. An encounter is not an event that necessarily occurs, but a disposition to occur if certain conditions are satisfied.

Each encounter may specify:

- **Gates:** Conditions that must hold for the encounter to be available.
- **Outcomes:** State changes that result from different resolutions of the encounter.

This structure makes conditionality explicit and separates availability from realization.

4.3 State and Tags

CAML tracks game state using lightweight, extensible state assertions such as tags, facts, and ownership relations. This allows encounters to depend on abstract conditions (e.g., an area being cleared, a faction being hostile) without encoding narrative assumptions.

5 CAML as a Domain-Specific Modeling Language

From a modeling perspective, CAML can be understood as a domain-specific modeling language (DSML) tailored to tabletop roleplaying games. It defines a constrained vocabulary and set of relations that capture the structural regularities of adventure design while excluding irrelevant details.

Unlike general-purpose programming or scripting languages, CAML is declarative. Authors describe what exists and under what conditions it may occur, rather than prescribing control flow. This makes CAML particularly suitable for analysis, visualization, and transformation.

6 Bridging Narrative Theory and Computation

CAML occupies a middle ground between narrative theory and computational representation. On the one hand, it preserves the interpretive openness and authorial intent characteristic of tabletop play. On the other hand, it exposes enough structure to enable algorithmic reasoning.

This duality allows CAML to serve as:

- a human-facing authoring format,
- an intermediate representation for tooling and export, and
- a research artifact for studying narrative possibility spaces.

In this sense, CAML does not replace narrative description but complements it with formal structure.

7 Applications

7.1 Visualization

Because encounters, conditions, and outcomes form an explicit graph, CAML adventures can be visualized as dependency networks. Such visualizations support design validation, debugging, and comprehension.

7.2 Procedural Remixing

CAML enables the safe recombination of adventure components by allowing tools to check whether encounter conditions remain satisfiable. This supports procedural generation and remixing without introducing unreachable content.

7.3 Interoperability

By serving as a canonical source representation, CAML can be exported to multiple virtual tabletop platforms, reducing authoring duplication and platform lock-in.

8 Limitations and Future Work

CAML is intentionally minimal. It does not attempt to encode narrative prose, perform automated storytelling, or replace the role of the game master. Future work includes richer state reasoning, formal semantics, and empirical evaluation of CAML-authored adventures in live play.

9 Conclusion

CAML proposes a shift in how tabletop adventures are represented: from linear documents to structured possibility spaces. By formalizing the conditional structure of play while remaining accessible to human authors, CAML offers a foundation for improved tooling, analysis, and interoperability in tabletop roleplaying games. More broadly, it demonstrates how domain-specific modeling languages can bridge narrative theory and computation in interactive systems.

References

- [1] Ryan, M.-L. (2001). *Narrative as Virtual Reality*. Johns Hopkins University Press.
- [2] Kelly, S., & Tolvanen, J.-P. (2008). *Domain-Specific Modeling*. Wiley.
- [3] Mateas, M., & Stern, A. (2005). Structuring content in the Faade interactive drama architecture. *Proceedings of Artificial Intelligence and Interactive Digital Entertainment*.

¹The CAML reference implementation and supporting materials are publicly available at <https://github.com/dkoepsell/CAML5e>.