UniLog: A Unified Logic Framework for Constraint Representation and Handling in Intelligent Systems

Harris Wang *

Posted Date: 26 March 2026

doi: 10.20944/preprints202603.2104.v1

Keywords: UniLog; constrained object hierarchies; constrained zone-object architecture; constraint specification; modal logic; temporal logic; modular logic; conservativity

 doi:10.20944/preprints202603.2104.v1

Article

UniLog: A Unified Logic Framework for Constraint Representation and Handling in Intelligent Systems

Harris Wang

School of Computing and Information Systems, Athabasca University, Athabasca, Canada;
harrisw@athabascau.ca

Abstract

This paper presents UniLog, a unified logic framework designed to integrate the expressive power and reasoning capabilities of diverse logical systems into a single coherent formalism. UniLog is specifically developed to serve as the constraint representation language for the Constrained Object Hierarchies (COH) and Constrained Zone-Object Architecture (CZOA) frameworks, enabling the specification of identity constraints, trigger constraints, goal constraints, and access constraints in a unified manner. The core novelty of UniLog lies not in any single logic but in its architecture: a modular, fibred system that combines a core modal-temporal substructure with conservatively extended modules for probabilistic, non-monotonic, epistemic, and domain-specific reasoning. We provide formal syntax, semantics, and inference rules for UniLog, demonstrate its embedding of major logic families, and illustrate its application to complex organizational intelligent information systems. In addition, we establish key meta-theoretic properties: conservativity of modular extensions (Theorem 3), soundness of the inference system, decidability of useful fragments, and soundness and relative completeness of a translation from a bounded fragment of UniLog to SMT-LIB. These results provide a rigorous foundation for using UniLog in formal verification and for integrating it with automated theorem provers.

Keywords: UniLog; constrained object hierarchies; constrained zone-object architecture; constraint specification; modal logic; temporal logic; modular logic; conservativity

---

1. Introduction

The **Constrained Object Hierarchies (COH)** framework [1, 2] and its enterprise-oriented extension, the **Constrained Zone-Object Architecture (CZOA)** [3], rely on constraints as first-class citizens for specifying system behavior. Constraints in COH/CZOA are categorized into four types:

- **Identity constraints (I):** invariants that must hold in all reachable states
- **Trigger constraints (T):** event–condition–action rules for reactive behavior
- **Goal constraints (G):** optimization objectives driving purposive behavior
- **Access constraints (C):** runtime checks for permission decisions

In the current CZOI implementation [4], constraints are represented as Python expressions evaluated in a context dictionary. While practical, this approach lacks formal semantics, proof-theoretic guarantees, and integration with automated reasoning tools.

Different constraint types naturally correspond to different logical paradigms: identity constraints align with first-order and description logics; trigger constraints with temporal and dynamic logics; goal constraints with preference and optimization logics; and access constraints with deontic and epistemic logics. Real-world organizational systems further require reasoning under uncertainty (probabilistic logic), handling exceptions (non-monotonic logic), and modeling agent knowledge (epistemic logic).

 doi:10.20944/preprints202603.2104.v1

These observations motivate the need for a **unified logical framework** that can integrate heterogeneous reasoning modes while preserving formal rigor. This paper introduces **UniLog**, a modular logic framework designed to serve precisely this role.

While UniLog is motivated by COH/CZOA, it is important to emphasize that all formal definitions, semantics, inference rules, and meta-theoretic results presented in this paper are fully self-contained. The COH/CZOA frameworks provide an application context and design motivation, but the logical framework developed here stands independently as a general solution to the problem of disciplined logic combination.

UniLog is designed as a **modular, fibred logic system**, inspired by Gabbay's fibring approach [38], in which a core modal–temporal logic is extended conservatively by specialized modules for probabilistic, non-monotonic, epistemic, deontic, description, dynamic, and fuzzy reasoning. This architecture enables UniLog to integrate over eight major logic families without collapsing them into a monolithic or semantically fragile system.

Contributions

The main contributions of this work are:

1. **A Modular Unified Logic.**
   We define UniLog as a logic with a core modal–temporal substructure and a set of conservatively extending modules, enabling principled integration of classical, modal, temporal, epistemic, deontic, probabilistic, non-monotonic, description, dynamic, and fuzzy logics.
2. **Layered Semantic Framework.**
   We provide a Kripke-style semantics layered with probability measures, preference relations, and graded truth interpretations, while preserving the semantics of the core logic.
3. **Meta-Theoretic Guarantees.**
   We prove soundness and relative completeness of a bounded translation from UniLog to SMT-LIB (Theorems 1–2), conservativity of modular extensions (Theorem 3), and soundness of the inference system (Proposition 4).
4. **Constraint-Oriented Interpretation.**
   We give a unified logical interpretation of identity, trigger, goal, and access constraints, bridging foundational logic with intelligent information system specification.
5. **Practical Feasibility.**
   Through SMT-based case studies using Z3, we demonstrate that bounded fragments of UniLog can be effectively analyzed using existing automated reasoning tools.

2. Related Work

The quest for a unified logical framework has a long history, from Leibniz's dream of a *characteristica universalis* to modern combinations of modal, temporal, and other logics. This section reviews the most influential contributions that inform UniLog's design.

2.1. Foundational Semantics: Kripke and Tarski

Kripke's possible worlds semantics [5] provided the pivotal insight that alethic, epistemic, deontic, and temporal modalities can all be interpreted through accessibility relations on possible worlds. Tarski's model-theoretic semantics [6] established the mathematical foundation for first-order logic (FOL), which remains the core of classical reasoning. Gödel's completeness theorem [7] and Herbrand's theorem [8] underpin modern automated theorem proving.

2.2. Modal and Temporal Unification

The combination of modal and temporal logics was explored by Thomason [9] and Emerson & Halpern [10]. Pnueli's introduction of LTL [11] and the subsequent development of CTL [12] provided

 doi:10.20944/preprints202603.2104.v1

the temporal backbone for program verification. Fagin et al. [13] integrated epistemic and temporal logic for distributed systems, directly influencing UniLog's treatment of knowledge and time.

2.3. Deontic and Epistemic Logics

Von Wright [14] formalized standard deontic logic (SDL), while Hintikka [15] established epistemic logic with possible worlds. Anderson's reduction [16] connected deontic to alethic modalities, and Meyer [17] integrated deontic concepts with dynamic logic. Multi-agent epistemic logic was systematized by Fagin et al. [13].

2.4. Probabilistic and Fuzzy Logics

Nilsson [18] introduced probabilistic logic, later extended by Halpern [19] and Fagin et al. [20]. For graded concepts, Zadeh's fuzzy sets [21] and Hájek's mathematical fuzzy logic [22] provide the foundation. Markov logic networks [23] combine FOL with probabilistic graphical models.

2.5. Non-Monotonic and Default Reasoning

Reiter's default logic [24] and McCarthy's circumscription [25] pioneered non-monotonic reasoning. Nute's defeasible logic [26] and Pollock's argumentation theory [27] offered computationally oriented frameworks.

2.6. Description Logics

The KL-ONE family [28] and the subsequent Description Logic Handbook [29] established DLs as the basis for ontological reasoning. Horrocks et al.'s work on OWL [30] and tableau-based reasoners [31] demonstrated practical applicability.

2.7. Dynamic Logic

Pratt [32] and Fischer & Ladner [33] developed dynamic logic for reasoning about programs. Harel, Kozen, and Tiuryn [34] provided a comprehensive treatment, connecting to Hoare logic [35] and Dijkstra's weakest preconditions [36].

2.8. Hybrid and Combined Logics

Hybrid logics [37] add explicit world references to modal logic. Gabbay's fibring approach [38] offers a systematic method for combining logics, influencing UniLog's modular design.

2.9. Practical Unified Frameworks

Common Logic (ISO 24707) [39] and KIF [40] provide interchange formats. SWRL [41] combines OWL with rules. Constraint logic programming [42] and Constraint Handling Rules [43] demonstrate integration of constraints with logic.

2.10. Gaps and Opportunities

Despite extensive work, existing frameworks typically combine at most three or four logic families. UniLog's integration of eight major families (classical, modal, temporal, epistemic, deontic, probabilistic, non-monotonic, DL) with fuzzy and dynamic extensions, its explicit mapping to COH/CZOA constraint types, and its commitment to conservative modular extension address these gaps. Furthermore, the explicit meta-theoretic results—especially the proof of conservativity—distinguish UniLog from many pragmatic combinations.

 doi:10.20944/preprints202603.2104.v1

3. Design Principles and Requirements

3.1. Core Requirements

UniLog must satisfy the following requirements derived from COH/CZOA and general AI reasoning:

Requirement	Description	Corresponding Logic Families
R1: Invariant specification	Express properties that must hold in all states	First-order logic, temporal logic (□)
R2: Event-condition-action	Model reactive behavior	Dynamic logic, temporal logic (until, next)
R3: Optimization objectives	Specify goals with quantitative measures	Linear programming, modal logic with preferences
R4: Access control	Express permissions, obligations, prohibitions	Deontic logic, epistemic logic
R5: Uncertainty handling	Reason with probabilistic knowledge	Probabilistic logic, Markov logic
R6: Default reasoning	Handle exceptions and non-monotonicity	Default logic, circumscription
R7: Hierarchical structure	Reason about zones and role hierarchies	Description logics, modal logic with nested modalities
R8: Temporal reasoning	Model system evolution over time	LTL, CTL, MTL
R9: Knowledge representation	Model what agents know	Epistemic logic
R10: Computational efficiency	Allow tractable fragments for practical use	Description logics (ALC, SHOIN)

3.2. Design Principles

1. **Modularity**: UniLog is built as a core logic with pluggable modules. The core provides basic propositional, modal, and temporal operators. Modules add specialized reasoning (probabilistic, non-monotonic, etc.). This follows the spirit of Gabbay's fibring [38], where logics are combined by fusing their semantic structures, but with a stronger emphasis on conservative extension.

 doi:10.20944/preprints202603.2104.v1

2. **Conservative extension**: Each module extends the core conservatively—formulas that do not use module-specific operators retain their original semantics. This is a critical meta-theoretic property (proved in Theorem 3) that ensures the framework is truly modular.
3. **Unified syntax**: All modules share a common syntactic framework with a uniform notation for modalities, quantifiers, and connectives.
4. **Layered semantics**: The semantics is defined in layers, with the core semantics (Kripke-style possible worlds) extended by module-specific structures (probability measures, preference orders, etc.).
5. **Compositional reasoning**: Inference rules are designed to be compositional — reasoning can be performed within modules and combined via interface rules. The system does not aim for global completeness, but preserves completeness for each module when they are used in isolation.
6. **Tractable fragments**: For practical implementation, UniLog identifies syntactic fragments that correspond to decidable logics (e.g., description logics, propositional LTL).

4. The UniLog Framework

4.1. Syntax

UniLog formulas are built from the following components:

4.1.1. Core Vocabulary

- **Atomic propositions**: $p, q, r, \ldots$ (from a countable set $\mathcal{P}$)
- **Constants**: $c, d, \ldots$ (individuals)
- **Variables**: $x, y, z, \ldots$
- **Functions**: $f, g, \ldots$ (mapping terms to terms)
- **Predicates**: $P, Q, R, \ldots$ (including equality $=$)

4.1.2. Logical Connectives (Classical Core)

- ¬ (negation)
- ∧ (conjunction)
- ∨ (disjunction)
- → (implication)
- ↔ (equivalence)
- ∀ (universal quantification)
- ∃ (existential quantification)

4.1.3. Modal Operators (Core Modalities)

Operator	Meaning	Logic Family
$\Box\phi$	Necessarily ϕ	Alethic modal
$\Diamond\phi$	Possibly ϕ	Alethic modal
$\mathbf{K}_a\phi$	Agent a knows ϕ	Epistemic

 doi:10.20944/preprints202603.2104.v1

$\mathbf{B}_a\phi$	Agent a believes ϕ	Doxastic
$\mathbf{O}\phi$	It is obligatory that ϕ	Deontic
$\mathbf{P}\phi$	It is permitted that ϕ	Deontic
$\mathbf{Forb}\phi$	It is forbidden that ϕ	Deontic
$\mathbf{X}\phi$	Next state: ϕ	Temporal (LTL)
$\mathbf{G}\phi$	Globally: ϕ always	Temporal (LTL)
$\mathbf{F}\phi$	Finally: ϕ eventually	Temporal (LTL)
$\phi\mathbf{U}\psi$	ϕ until ψ	Temporal (LTL)
$\mathbf{A}\phi$	All paths: ϕ	Temporal (CTL)
$\mathbf{E}\phi$	Exists path: ϕ	Temporal (CTL)
$[\alpha]\phi$	After program α, ϕ	Dynamic logic
$\langle\alpha\rangle\phi$	Program α can execute and result in ϕ	Dynamic logic

4.1.4. Probabilistic Operators

- $\mathbb{P}_{\geq r}(\phi)$: Probability that ϕ holds is at least r
- $\mathbb{E}[t]$: Expected value of term t

4.1.5. Non-Monotonic Operators

- $\phi \Rightarrow \psi$: Default rule (if ϕ then typically ψ)
- $\mathbf{Ab}_p$: Abnormality predicate for circumscription

4.1.6. Preference/Optimization Operators

- $\phi \prec \psi$: ϕ is preferred over ψ
- $\mathbf{Opt}(\phi)$: ϕ is optimal (with respect to some goal)

4.1.7. Description Logic Constructs

- $C \sqcap D$: Concept conjunction
- $C \sqcup D$: Concept disjunction
- $\neg C$: Concept negation
- $\exists R.C$: Existential restriction
- $\forall R.C$: Universal restriction
- $\geq nR.C$: Number restriction (at least n)

 doi:10.20944/preprints202603.2104.v1

- $\leq nR.C$: Number restriction (at most n)

4.2. Well-Formed Formulas

The set of well-formed formulas is defined inductively:

1. Atomic propositions and predicate applications are formulas.
2. If ϕ and ψ are formulas, so are $\neg\phi,\ \phi \wedge \psi,\ \phi \vee \psi,\ \phi \rightarrow \psi,\ \phi \leftrightarrow \psi$.
3. If ϕ is a formula and x is a variable, $\forall x\phi$ and $\exists x\phi$ are formulas.
4. If ϕ is a formula, then for any modal operator M (from the tables above), $M\phi$ is a formula.
5. If ϕ and ψ are formulas, $\phi \mathbf{U} \psi$ is a formula.
6. If ϕ is a formula and $r \in [0,1]$, $\mathbb{P}_{\geq r}(\phi)$ is a formula.
7. If ϕ and ψ are formulas, $\phi \Rightarrow \psi$ is a formula.
8. If ϕ and ψ are formulas, $\phi \prec \psi$ is a formula.
9. If C and D are concepts (defined from predicates), the description logic constructs are formulas.
10. Nothing else is a formula.

4.3. Semantics

UniLog semantics is defined over **UniLog structures** – a generalization of Kripke models that incorporate multiple dimensions of modality.

4.3.1. Core Structure

A **UniLog frame** is a tuple:

$$\mathcal{F} = (W, \{R_M\}_{M \in \mathcal{M}}, V, \{\Pr_w\}_{w \in W}, \{\prec_w\}_{w \in W})$$

where:

- W is a non-empty set of possible worlds (states).
- For each modal operator M, $R_M \subseteq W \times W$ is an accessibility relation.
- $V: W \times \mathcal{P} \rightarrow \{0,1\}$ is a valuation function for atomic propositions.
- $\Pr_w$ is a probability measure on subsets of W (for probabilistic reasoning).
- $\prec_w$ is a preference relation on W (for optimization).

4.3.2. Satisfaction Relation

The satisfaction relation $\mathcal{M}, w \vDash \phi$ (where $\mathcal{M}$ is a model based on frame $\mathcal{F}$) is defined inductively:

Classical connectives: Standard Tarskian semantics.

Modal operators:

- $\mathcal{M}, w \vDash \Box\phi$ iff for all v with $(w, v) \in R_{\Box}$, $\mathcal{M}, v \vDash \phi$.
- $\mathcal{M}, w \vDash \Diamond \phi$ iff there exists v with $(w, v) \in R_{\Diamond}$ such that $\mathcal{M}, v \vDash \phi$.

Epistemic:

- $\mathcal{M}, w \vDash \mathbf{K}_a\phi$ iff for all v with $(w, v) \in R_{K_a}$, $\mathcal{M}, v \vDash \phi$.

(Typically R_{K_a} is an equivalence relation.)

Temporal (LTL):

- For a path $\pi = w_0, w_1, \ldots$:
  - $\pi \vDash \mathbf{X}\phi$ iff $\pi_1 \vDash \phi$
  - $\pi \vDash \mathbf{G}\phi$ iff for all $i \geq 0$, $\pi_i \vDash \phi$
  - $\pi \vDash \mathbf{F}\phi$ iff exists $i \geq 0$ such that $\pi_i \vDash \phi$
  - $\pi \vDash \phi \mathbf{U} \psi$ iff exists $i \geq 0$ such that $\pi_i \vDash \psi$ and for all $j < i$, $\pi_j \vDash \phi$

 doi:10.20944/preprints202603.2104.v1

Probabilistic:

- $\mathcal{M}, w \vDash \mathbb{P}_{\geq r}(\phi)$ iff $\Pr_w(\{v \mid \mathcal{M}, v \vDash \phi\}) \geq r$

Non-monotonic:

- $\mathcal{M}, w \vDash \phi \Rightarrow \psi$ iff in all most-preferred worlds (by $\prec_w$) where ϕ holds, ψ holds.

Optimization:

- $\mathcal{M}, w \vDash \phi \prec \psi$ iff for all v with $\mathcal{M}, v \vDash \phi$ and all u with $\mathcal{M}, u \vDash \psi$, $v \prec_w u$.

Description logic constructs: Standard DL semantics extended with possible worlds.

Fuzzy logic: For fuzzy interpretations, the satisfaction relation is replaced by a graded truth function (see §5.10).

4.4. Inference System

UniLog provides a modular inference system. The core consists of:

1. **Axioms for classical first-order logic** (all tautologies, quantifier axioms).
2. **Modal axioms** depending on the properties of accessibility relations:
   - **K**: $\Box(\phi \to \psi) \to (\Box\phi \to \Box\psi)$
   - **T**: $\Box\phi \to \phi$ (reflexive)
   - **4**: $\Box\phi \to \Box\Box\phi$ (transitive)
   - **5**: $\Diamond\phi \to \Box\Diamond\phi$ (Euclidean)
3. **Temporal axioms** (from LTL/CTL).
4. **Probabilistic axioms** (from probability theory).
5. **Default logic rules** (e.g., Reiter's default rules).
6. **Description logic axioms** (from ALC, SHOIN, etc.).

Inference rules:

- Modus ponens: $\frac{\phi, \phi \to \psi}{\psi}$
- Necessitation: $\frac{\phi}{\Box\phi}$
- Temporal generalization: $\frac{\phi}{\mathbf{G}\phi}$
- Probability preservation: $\frac{\phi \equiv \psi}{\mathbb{P}_{\geq r}(\phi) \equiv \mathbb{P}_{\geq r}(\psi)}$

A note on completeness: The UniLog inference system is not designed to be globally complete (the combination of its components would likely be incomplete). Instead, it is structured to preserve completeness for each module when formulas are confined to that module's language. This modular approach to completeness reflects the fibred nature of the framework and is a common trade-off in combined logics.

4.5. Modular Structure

UniLog is organized into modules that can be combined:

Module	**Operators**	**Semantics Extension**	**Axioms**
Core	$\neg, \wedge, \vee, \to, \forall, \exists$	Classical Tarski	FOL axioms
Modal	$\Box, \Diamond$	Kripke frames	K, T, 4, 5, etc.
Epistemic	$\mathbf{K}_a, \mathbf{B}_a$	Epistemic frames (S5)	S5 axioms

 doi:10.20944/preprints202603.2104.v1

Deontic	$\mathbf{O}, \mathbf{P}, \mathbf{Forb}$	Deontic frames (serial)	D, etc.
Temporal	$\mathbf{X}, \mathbf{G}, \mathbf{F}, \mathbf{U}$	Temporal frames	LTL axioms
Dynamic	$[\alpha], \langle\alpha\rangle$	Program semantics	PDL axioms
Probabilistic	$\mathbb{P}_{\geq r}$	Probability measures	Probability axioms
Non-monotonic	$\Rightarrow$	Preference orders	Default rules
Optimization	$\prec, \mathbf{Opt}$	Preference orders	Preference logic
Description	$\sqcap, \sqcup, \exists R.C, \forall R.C$	DL interpretations	DL axioms

5. Embedding Existing Logics into UniLog

5.1. Classical First-Order Logic

FOL is directly embedded in the Core module. Every FOL formula is a UniLog formula with no modal operators.

5.2. Modal Logics (K, T, S4, S5)

Each modal logic corresponds to a specific choice of axioms in the Modal module, with semantics grounded in Kripke frames [5]:

- **K**: No additional axioms beyond K and necessitation
- **T**: Add axiom T ($\Box\phi \rightarrow \phi$)
- **S4**: Add T and 4 ($\Box\phi \rightarrow \Box\Box\phi$)
- **S5**: Add T, 4, and 5 ($\Diamond\phi \rightarrow \Box\Diamond\phi$)

5.3. Temporal Logics (LTL, CTL, CTL)*

Temporal operators are directly provided in UniLog following Pnueli [11] and Emerson & Halpern [10]. For example:

- LTL formula $\mathbf{G}(p \rightarrow \mathbf{F}q)$ becomes $\mathbf{G}(p \rightarrow \mathbf{F}q)$ in UniLog.
- CTL formula $\mathbf{AG}(p \rightarrow \mathbf{EF}q)$ becomes $\mathbf{AG}(p \rightarrow \mathbf{EF}q)$.

5.4. Epistemic Logic

Multi-agent epistemic logic with common knowledge, as per Hintikka [15] and Fagin et al. [13].:

- $\mathbf{K}_a\phi$ as above.
- $\mathbf{C}_G\phi$ (common knowledge among group G) can be defined as the fixed point $\mathbf{E}_G(\phi \wedge \mathbf{C}_G\phi)$ where $\mathbf{E}_G\phi = \bigwedge_{a\in G} \mathbf{K}_a\,\phi$.

5.5. Deontic Logic

Standard deontic logic (SDL) with axioms:

- $\mathbf{O}(\phi \rightarrow \psi) \rightarrow (\mathbf{O}\phi \rightarrow \mathbf{O}\psi)$
- $\mathbf{O}\phi \rightarrow \neg\mathbf{O}\neg\phi$ (consistency)
- $\mathbf{P}\phi \equiv \neg\mathbf{O}\neg\phi$

 doi:10.20944/preprints202603.2104.v1

- **Forb**$\phi \equiv \mathbf{O}\neg\phi$

  According to von Wright [14] and Anderson [16].

5.6. Probabilistic Logic

According to Nilsson [18], Halpern [19], and Fagin et al. [20], probabilistic formulas $\mathbb{P}_{\geq r}(\phi)$ satisfy:

- $\mathbb{P}_{\geq 0}(\phi)$ is a tautology
- $\mathbb{P}_{\geq r}(\phi) \rightarrow \mathbb{P}_{\geq s}(\phi)$ for $r \geq s$
- $\mathbb{P}_{\geq r}(\phi) \wedge \mathbb{P}_{\geq s}(\psi) \wedge \neg(\phi \wedge \psi) \rightarrow \mathbb{P}_{\geq \min(1, r+s)}(\phi \vee \psi)$

5.7. Default Logic

Reiter's default rules $\frac{\alpha:\beta}{\gamma}$ are encoded as:

$$(\alpha \wedge \Diamond\, \beta) \Rightarrow \gamma$$

where $\Rightarrow$ is the non-monotonic implication [24].

5.8. Description Logics

Description logic [29] concepts map to UniLog formulas with one free variable:

- $C \sqcap D$: $C(x) \wedge D(x)$
- $\exists R.C$: $\exists y(R(x,y) \wedge C(y))$
- $\forall R.C$: $\forall y(R(x,y) \rightarrow C(y))$
- $\geq nR.C$: $\exists_{\geq n} y(R(x,y) \wedge C(y))$ (using counting quantifiers)

5.9. Dynamic Logic

Dynamic Logic (DL) [33,34] extends modal logic with modalities that describe the behaviour of programs or actions. It provides a formal framework for reasoning about state transitions caused by executing an action or program. In UniLog, Dynamic Logic is integrated via two core operators:

- **Box modality** $[\alpha]\phi$: "after every terminating execution of program α, ϕ holds".
- **Diamond modality** $\langle\alpha\rangle\phi$: "program α can execute and terminate in a state where ϕ holds".

Programs α are built from atomic actions using standard program constructs: sequence (;), choice (∪), iteration (∗), and test (?). Atomic actions correspond to operations in the system (e.g., database updates, user actions, system calls). The semantics is defined over a set of states W with a transition relation $R_\alpha \subseteq W \times W$ for each atomic action α; complex programs are interpreted via the usual relational composition.

Examples in UniLog

Example 1: Order entry in a trading system

unilog

[enter_order] (order_entered ∧ balance_updated)

This formula asserts that after every execution of the atomic action enter_order, both order_entered and balance_updated hold.

Example 2: Conditional prescription in healthcare

unilog

⟨prescribe_med; dispense_med⟩ (medication_given)

There exists an execution path where after prescribing and then dispensing, the medication is given.

Example 3: Safety property (no double spending)

unilog

[(enter_order)*] (¬(balance < 0))

 doi:10.20944/preprints202603.2104.v1

No matter how many times enter_order is executed (zero or more), the balance never becomes negative.

Example 4: Using test for access control

unilog

[(has_permission? ; perform_action)] success

If the user has permission, then performing the action leads to success.

Relationship to COH/CZOA

In CZOA, operations (O) are exactly the atomic actions of Dynamic Logic. The permission engine determines whether an operation is allowed in a given state. Dynamic Logic formulas can express pre- and post-conditions of operations, and can be used to specify system behaviour at a high level. For example, an identity constraint might require that after any sequence of operations, certain invariants hold (expressed with the box modality over Kleene star). Trigger constraints can be seen as dynamic formulas where an event (test) triggers an action.

5.10. Fuzzy Logic

Fuzzy Logic [21, 22] generalises classical logic by allowing truth values in the continuous interval **[0, 1]**. In UniLog, it is embedded through the probabilistic module extended with graded modalities and fuzzy predicates.

A fuzzy model $\mathcal{M}$ assigns to each world **w** a fuzzy interpretation: each predicate **P** of arity **n** is interpreted as a function

$$\mathbf{P}^{(\mathcal{M}, w)}: \mathbf{D}^{n} \rightarrow [0, 1].$$

The truth value of a formula is computed recursively using a chosen *t-norm* (e.g., Gödel, Łukasiewicz). Graded statements $\mathbb{T}_r(\varphi)$ are defined as follows:

$$\mathcal{M}, w \models \mathbb{T}_r(\varphi) \text{ iff } [\varphi]^w \geq r.$$

Fuzzy logic is invaluable for handling graded concepts such as *"high risk"* and for integrating neural-network outputs into symbolic reasoning.

Examples in UniLog

Example 1: Fuzzy medical diagnosis

unilog

high_fever(p) ∧ cough(p) → likely_flu(p)

If we interpret the connectives with Gödel semantics, the truth of the antecedent is min (high_fever(p),cough(p)), and the implication (as in residuated fuzzy logic) yields a truth value that can be used for decision support.

Example 2: Graded access control

unilog

risk_level(u) > 0.8 → block_access(u)

Here risk_level(u) is a fuzzy predicate computed by a neural component. The constraint fires when the risk exceeds a threshold.

Example 3: Goal constraint with fuzzy target

unilog

G(patient_satisfaction ≥ 0.9)

Using the graded modality $\mathbb{T}_{0.9}$(patient_satisfaction), this asserts that globally, patient satisfaction is at least 0.9.

Example 4: Fuzzy preference for optimization

unilog

 doi:10.20944/preprints202603.2104.v1

(throughput = t1) ≺ (throughput = t2) ← t1 < t2

This expresses that higher throughput is preferred – a fuzzy ordering can be defined directly on numerical attributes.

Integration with Probabilistic Module

The probabilistic module already provides $\mathbb{P}_{\geq r}(\phi)$, which is a graded statement about probability. Fuzzy logic can be seen as a different interpretation of graded truth, where the grade is not a probability but a degree of membership or truth. In UniLog, both are available, and they can be combined: e.g., $\mathbb{P}_{\geq 0.9}(\text{high\_risk}(u))$ means the probability that user u is (fuzzily) high-risk is at least 0.9 – a hybrid statement.

Application in COH/CZOA

Fuzzy logic is invaluable for:

- **Goal constraints**: many business objectives are inherently graded (e.g., "maximize customer satisfaction").
- **Trigger conditions**: conditions like "patient is very ill" are naturally fuzzy.
- **Access control**: risk-based access control uses fuzzy thresholds.
- **Neural component output**: neural networks produce continuous values that can be directly interpreted as fuzzy truth degrees.

By embedding fuzzy logic, UniLog allows constraints to reason with the continuous outputs of learned components, closing the loop between symbolic reasoning and machine learning.

6. Formal Properties of UniLog

This section establishes the core meta-theoretic properties of UniLog. We first prove that the translation from a bounded fragment of UniLog to SMT-LIB is sound (Theorem 1) and relatively complete for finite models (Theorem 2). We then show that the modular extension of UniLog is conservative (Theorem 3), meaning that adding modules does not affect the meaning of core formulas. Finally, we discuss the soundness of the inference system and summarize the decidability and complexity of useful fragments.

6.1. Translation Soundness (Theorem 1)

To leverage the power of modern SMT solvers (e.g., Z3, CVC5), we define a translation $[\![\cdot]\!]$ from a bounded fragment of UniLog, called **UniLog-SMT**, into SMT-LIB formulas. The fragment includes quantifier-free first-order formulas with equality, uninterpreted functions, and linear arithmetic; temporal and modal operators are handled by bounded unrolling up to a fixed depth k. This translation is used in the case studies of Section 9.

Definition (UniLog-SMT).

A UniLog-SMT formula Γ is built from:

- Atomic predicates over terms, including equality.
- Boolean connectives $\neg, \wedge, \vee, \rightarrow$.
- Quantifiers (optional; in practice we often restrict to quantifier-free).
- Modal operators $\Box$ and $\Diamond$ applied to quantifier-free formulas, interpreted over a fixed set of states $\{0, \ldots, k\}$ with a given accessibility relation.
- Temporal operators $\mathbf{X}, \mathbf{G}, \mathbf{F}, \mathbf{U}$ bounded to a finite horizon k (e.g., $\mathbf{F}_{\leq k}$).
- Probabilistic operators $\mathbb{P}_{\geq r}(\phi)$ where ϕ is quantifier-free and r is a rational constant; these are translated into arithmetic constraints on indicator variables.

The translation $[\![\Gamma]\!]$ produces an SMT-LIB formula by:

- Introducing a copy of all variables and predicates for each time step / state.
- Encoding the accessibility relation as a Boolean function or explicit transitions.

 doi:10.20944/preprints202603.2104.v1

- Replacing each modal or temporal operator with the corresponding constraints on the state copies (e.g., $[\![\Box\phi]\!]$ becomes $\bigwedge_{j\in R(i)}[\![\phi]\!]^{(j)}$ where $[\![\phi]\!]^{(j)}$ is ϕ evaluated at state j).
- Translating probabilistic statements into linear inequalities over the probabilities of state formulas (with probabilities given as constants or variables).

Theorem 1 (Translation Soundness).

Let Γ be a UniLog-SMT formula with bound k (i.e., all modalities are unrolled to depth k). If an SMT model $\mathcal{M}_{\text{SMT}}$ satisfies the translation $[\![\Gamma]\!]$, then there exists a UniLog interpretation $\mathcal{I}$ with at most k states such that $\mathcal{I} \vDash \Gamma$.

Proof sketch.

The SMT model provides an assignment to every copy of each variable and predicate for each state index $0, \ldots, k$. Construct a Kripke structure $\mathcal{I}$ with states $W = \{0, \ldots, k\}$, valuation $V(i)$ defined by the assignments to the i-th copy, and accessibility relations derived from the encoded transition predicates. By induction on the structure of Γ, one shows that for any subformula ψ, $\mathcal{M}_{\text{SMT}}$ satisfies the translation of ψ at state i iff $\mathcal{I}, i \vDash \psi$. The base case (atomic formulas) holds by construction; the inductive steps follow because the translation mirrors the semantic clauses. Hence $\mathcal{I}$ models Γ. ■

6.2. Relative Completeness for Bounded Models (Theorem 2)

Theorem 2 (Relative Completeness).

For the quantifier-free fragment of UniLog-SMT without uninterpreted functions but with linear arithmetic and equality (denoted UniLog-SMT$_0$), if a formula Γ is satisfiable in a finite UniLog model whose number of states does not exceed the bound k, then the translation $[\![\Gamma]\!]$ is SMT-satisfiable.

Proof sketch.

Given a finite UniLog model $\mathcal{I}$ with states W (where $|W| \leq k$) and a distinguished world w_0 satisfying Γ, we construct an SMT assignment as follows: for each state $i \in W$, assign the values of variables and predicates according to $V(i)$; for indices beyond $|W|$, assign arbitrary values consistent with the translation's constraints (e.g., accessibility relations can be made false). The translation's encoding of the semantic conditions is exact, so this assignment satisfies every translated subformula. In particular, at the copy corresponding to w_0, all constraints hold, yielding an SMT model of $[\![\Gamma]\!]$. ■

These theorems justify the use of SMT solvers for bounded verification of UniLog constraints, as demonstrated in the case studies of Section 9.

6.3. Conservativity of Modular Extensions (Theorem 3)

Theorem 3 (Conservativity).

For any UniLog module M that extends the core (e.g., probabilistic, non-monotonic, description logic), the extension is conservative: if ϕ is a formula in the language of the core (i.e., containing only classical connectives, quantifiers, and the basic modal operators $\Box$,$\Diamond$ with no module-specific symbols), then ϕ is satisfiable in the core semantics if and only if it is satisfiable in the extended semantics of M. Moreover, for any such ϕ, the core proof system derives ϕ if and only if the extended system derives ϕ.

Proof sketch.

The semantics of each module is defined by enriching the core Kripke frame with additional structures (e.g., probability measures $\Pr_w$, preference relations $\prec_w$, or DL interpretations) while keeping the set of worlds W, the accessibility relations for core modalities, and the valuation V unchanged. The satisfaction of a core formula ϕ at a world w depends only on $V(w)$ and the core accessibility relations; it does not refer to the extra structures. Hence ϕ holds in the extended model exactly when it holds in the core reduct. For the proof system, module-specific inference rules apply only to formulas containing module operators; derivations of core formulas can use only core axioms and rules. Therefore any core derivation in the extended system is already a

 doi:10.20944/preprints202603.2104.v1

core derivation. Conversely, any core derivation remains valid in the extended system because the core rules are still sound. ■

This conservativity property guarantees that reasoning about pure core formulas is unaffected by the presence of modules, enabling modular development of theories and incremental verification.

6.4. Soundness of the Inference System

Proposition 4 (Soundness).

The inference system of UniLog (core plus all modules) is sound: if a formula ϕ is derivable from a set of axioms using the inference rules, then ϕ is valid in all UniLog models that satisfy the corresponding frame conditions (e.g., reflexivity for T, seriality for deontic D, etc.).

Proof sketch.

The proof proceeds by induction on the length of the derivation. Each axiom corresponds to a formula that is true in all models meeting the required conditions (e.g., the K axiom $\Box(\psi \to \chi) \to (\Box\psi \to \Box\chi)$ holds in any frame, the T axiom $\Box\psi \to \psi$ holds in reflexive frames, etc.). The inference rules preserve validity: modus ponens is standard; necessitation (from ψ infer $\Box\psi$) holds because if ψ is valid in all worlds, then in any world all accessible worlds satisfy ψ; temporal generalization similarly; probability preservation follows from the fact that equivalent formulas have the same truth set and hence the same probability. For module-specific rules (e.g., default rules, preference transitivity), we verify that they hold in the intended semantics. Thus any derivable formula is universally true. ■

A full completeness result for the entire UniLog is not possible because the combination of first-order quantification, modal operators, and fixed-point constructs (e.g., common knowledge, temporal until) leads to incompleteness phenomena in general. However, for each individual module that corresponds to a known complete logic (e.g., S5 epistemic logic, LTL, ALC description logic), the combined system inherits completeness when the modules do not interact. Investigating completeness for specific interacting fragments is an active research direction.

6.5. Decidability and Complexity of Fragments

While the full UniLog language is undecidable (it subsumes first-order logic), several practically relevant fragments are decidable and have well-understood complexity. Table 1 summarizes key fragments.

Table 1. Decidable Fragments of UniLog.

Fragment	Logic Family	Decidability / Complexity
UniLog-ALC	Description logic	ExpTime-complete [29]
UniLog-LTL	Propositional LTL	PSPACE-complete [44]
UniLog-CTL	Propositional CTL	EXPTIME-complete (satisfiability), P (model checking) [10,45]
UniLog-S5	Single-agent epistemic (S5)	PSPACE-complete [13]
UniLog-K	Basic modal K	PSPACE-complete [5]

 doi:10.20944/preprints202603.2104.v1

UniLog-SMT	Quantifier-free FO + theories	Decidable for certain theories (e.g., LIA, LRA) [42]
UniLog-prob	Probabilistic (linear inequalities)	Decidable via linear programming [20]
UniLog-DL (propositional)	Dynamic logic (PDL)	EXPTIME-complete [33]

These fragments can be used in the CZOI implementation by restricting input constraints to the appropriate sublanguage, enabling efficient automated reasoning. For example, identity constraints often fall into UniLog-ALC or UniLog-LTL, while access constraints may use UniLog-S5 or UniLog-DL.

7. UniLog for COH/CZOA Constraints

7.1 Constraint Types in UniLog

Identity Constraints (I)

Identity constraints are invariants that must hold in all states. In UniLog, they are expressed as:

$$\mathbf{G}\phi$$

where ϕ is a first-order or description logic formula. For example, the zone containment principle $U_z \subseteq U_{\text{parent}(z)}$ becomes:

$$\mathbf{G}\forall u(\text{inZone}(u,z) \rightarrow \text{inZone}(u,\text{parent}(z)))$$

Trigger Constraints (T)

Trigger constraints are event-condition-action rules. In UniLog:

$$\mathbf{G}(\text{event}(e) \wedge \text{condition} \rightarrow \mathbf{X}\text{action})$$

For example, a critical lab alert:

$$\mathbf{G}(\text{labResult}(p,r) \wedge \text{critical}(r) \rightarrow \mathbf{X}\text{alertPhysician}(p))$$

Goal Constraints (G)

Goal constraints are optimization objectives. They can be expressed as preference statements:

$$\mathbf{Opt}(\phi)\text{or}\phi \prec \psi$$

For example, minimize mortality rate:

$$\forall r_1, r_2(\text{mortalityRate}(r_1) \prec \text{mortalityRate}(r_2) \leftrightarrow r_1 < r_2)$$

Access Constraints (C)

Access constraints are runtime checks that involve permissions, obligations, and prohibitions:

$$\mathbf{P}\big(\text{do}(u,op)\big) \text{ or } \mathbf{O}\big(\text{do}(u,op)\big) \text{ or } \mathbf{F}(\text{do}(u,op))$$

For example, separation of duty:

$$\mathbf{F}(\text{prescribe}(u,m) \wedge \text{dispense}(u,m))$$

8. Implementation in CZOI

The UniLog framework can be integrated into the CZOI toolkit as a constraint specification language. UniLog formulas can be translated to the input languages of various solvers (e.g., SMT-LIB for Z3, NuSMV for temporal formulas, OWL reasoners for description logic). The permission engine can use UniLog to evaluate access constraints at runtime. The following are some examples.

```
# python
from czoi.unilog import UniLogFormula, UniLogModel
from czoi.constraint import Constraint, ConstraintType
```

 doi:10.20944/preprints202603.2104.v1


```
# Create a UniLog formula
formula = UniLogFormula("G(forall u (inZone(u, z) -> inZone(u, parent(z))))")

# Use it in a constraint
constraint = Constraint(
    name="ZoneContainment",
    type=ConstraintType.IDENTITY,
    target={"zones": "all"},
    condition=formula     # instead of string
)

# Evaluate in a model
model = UniLogModel(system)
result = model.check(formula)

# The permission engine uses UniLog to evaluate access constraints at runtime
def decide(self, user, operation, zone, context):
    # ... compute effective permissions
    # Check UniLog access constraints
    for constraint in self.get_access_constraints():
        formula = constraint.condition
        if not model.evaluate(formula, context):
            return False
    return True
```

9. Case Studies and Evaluation (Z3)

We conducted three small-scale case studies using Z3 to validate the SMT translation:

- **Case Study A (UNSAT + unsat core)**: Encoded zone-based access control with a policy contradiction. Z3 returned UNSAT and identified the minimal conflicting constraints.
- **Case Study B (SAT + counterexample model)**: Performed bounded model checking (k=2) on a trigger rule. Z3 returned SAT with a model showing a violation.
- **Case Study C (Soft constraints)**: Expressed goal constraints with soft preferences. Z3 returned an optimal assignment (x=7, y=0) satisfying the hard constraints.

These experiments demonstrate the feasibility of using off-the-shelf SMT solvers for bounded verification of UniLog constraints.

10. Five Case Studies Demonstrating UniLog in Real-World Reasoning

To further illustrate UniLog's expressive power, we present five detailed case studies across different domains. These are presented as **semantic witnesses**—demonstrations of how UniLog's modular combination of logics can capture complex real-world requirements in a single, coherent formalization.

Case Study 1: Healthcare -- Sepsis Alert System combines probabilistic risk assessment, temporal obligations, and deontic permissions to model a hospital sepsis alert system.

 doi:10.20944/preprints202603.2104.v1

Case Study 2: Finance -- Insider Trading Detection uses epistemic logic (for knowledge of non-public facts), deontic logic (for prohibitions), and temporal constraints to model insider trading detection.

Case Study 3: Smart City -- Traffic Incident Response employs dynamic logic (for actions), temporal logic (for time bounds), and modal operators to model a traffic management system.

Case Study 4: University -- Student Academic Probation demonstrates non-monotonic reasoning with default rules to handle exceptions in academic probation policies.

Case Study 5: Supply Chain -- Cold Chain Temperature Violation uses fuzzy logic to capture graded thresholds and temporal accumulation in temperature monitoring.

These case studies illustrate UniLog's ability to unify diverse logical paradigms within a single framework, enabling rigorous verification and automated reasoning for complex intelligent systems.

11. Comparison with Existing Unified Logics

Framework	Classical	Modal	Temporal	Epistemic	Deontic	Probabilistic	Non - monotonic	DL
First-order logic	✓							
Modal logic (S5)	✓	✓						
Temporal logic (LTL)	✓		✓					
Epistemic logic	✓			✓				
Deontic logic	✓				✓			
Probabilistic logic	✓					✓		
Default logic	✓						✓	
Description logic	✓							✓
Hybrid logic	✓	✓						

 doi:10.20944/preprints202603.2104.v1

Framework	Classical	Modal	Temporal	Epistemic	Deontic	Probabilistic	Non - monotonic	DL
UniLog (this paper)	✓	✓	✓	✓	✓	✓	✓	✓

UniLog uniquely combines all major reasoning modalities within a single, coherent framework, supported by meta-theoretic results on modularity and conservativity.

12. Conclusions

This paper has introduced **UniLog**, a unified logic framework designed to serve as the constraint specification language for COH and CZOA. UniLog integrates the expressive power of classical, modal, temporal, epistemic, deontic, probabilistic, non-monotonic, and description logics into a single formalism with modular syntax, layered semantics, and compositional inference rules. We have established key meta-theoretic properties: soundness, conservativity of modular extensions, and soundness and relative completeness of a translation to SMT-LIB for a bounded fragment. By providing a rigorous foundation for identity, trigger, goal, and access constraints, UniLog enables formal verification of adaptive intelligent systems and bridges the gap between theoretical logic and practical implementation in the CZOI toolkit. Future work includes developing efficient decision procedures for UniLog fragments, implementing a UniLog reasoner in CZOI, and applying UniLog to the formal verification of the case study systems.

References

1. H. Wang, "Constrained Object Hierarchies as a Unified Theoretical Model for Intelligence and Intelligent Systems," *Computers*, vol. 14, p. 478, 2025.
2. H. Wang, "The 9-Tuple Formation of Constrained Object Hierarchies: The Mathematical Foundation and Implications for Artificial Intelligence," preprint, 2026.
3. H. Wang, "Constrained Zone-Object Architecture (CZOA): A Unified Framework for Building Secure and Intelligent Integrated Organizational Systems," preprint, 2026.
4. H. Wang, *CZOI Python Toolkit: Technical Specification and API Reference*, Technical Report, 2026.
5. S. A. Kripke, "A Completeness Theorem in Modal Logic," *Journal of Symbolic Logic*, vol. 24, no. 1, pp. 1–14, 1959.
6. A. Tarski, *Logic, Semantics, Metamathematics*, Oxford University Press, 1956.
7. K. Gödel, "Die Vollständigkeit der Axiome des logischen Funktionenkalküls," *Monatshefte für Mathematik und Physik*, vol. 37, pp. 349–360, 1930.
8. J. Herbrand, *Recherches sur la théorie de la démonstration*, Doctoral dissertation, University of Paris, 1930.
9. A. Thomason, "Combinations of Tense and Modality," in *Handbook of Philosophical Logic*, vol. 2, 1984.
10. E. A. Emerson and J. Y. Halpern, "'Sometimes' and 'Not Never' Revisited," *Journal of the ACM*, vol. 33, no. 1, pp. 151–178, 1986.
11. A. Pnueli, "The Temporal Logic of Programs," in *Proc. FOCS*, 1977.
12. E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*, MIT Press, 1999.
13. R. Fagin et al., *Reasoning about Knowledge*, MIT Press, 1995.
14. G. H. von Wright, "Deontic Logic," *Mind*, vol. 60, pp. 1–15, 1951.
15. J. Hintikka, *Knowledge and Belief*, Cornell University Press, 1962.
16. A. R. Anderson, "A Reduction of Deontic Logic to Alethic Modal Logic," *Mind*, 1958.
17. J.-J. C. Meyer, "Deontic Logic as a Variant of Dynamic Logic," *Notre Dame Journal of Formal Logic*, 1988.

 doi:10.20944/preprints202603.2104.v1

18. N. J. Nilsson, "Probabilistic Logic," *Artificial Intelligence*, 1986.
19. J. Y. Halpern, *Reasoning about Uncertainty*, MIT Press, 2003.
20. R. Fagin, J. Y. Halpern, and N. Megiddo, "A Logic for Reasoning about Probabilities," *Information and Computation*, 1990.
21. L. A. Zadeh, "Fuzzy Sets," *Information and Control*, 1965.
22. P. Hájek, *Metamathematics of Fuzzy Logic*, Kluwer, 1998.
23. D. M. Gabbay, *Fibring Logics*, Oxford University Press, 1999.
24. R. Reiter, "A logic for default reasoning," *Artificial Intelligence*, vol. 13, nos. 1–2, pp. 81–132, 1980.
25. J. McCarthy, "Circumscription – A form of non-monotonic reasoning," *Artificial Intelligence*, vol. 13, nos. 1–2, pp. 27–39, 1980.
26. D. Nute, "Defeasible logic," in *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol. 3, D. Gabbay, C. Hogger, and J. Robinson, Eds. Oxford, U.K.: Oxford Univ. Press, 1994, pp. 353–395.
27. J. L. Pollock, "Defeasible reasoning," *Cognitive Science*, vol. 11, no. 4, pp. 481–518, 1987.
28. R. J. Brachman and J. G. Schmolze, "An overview of the KL-ONE knowledge representation system," *Cognitive Science*, vol. 9, no. 2, pp. 171–216, 1985.
29. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2003.
30. M. Dean and G. Schreiber, Eds., *OWL Web Ontology Language Reference*. W3C Recommendation, 2004.
31. D. Tsarkov and I. Horrocks, "FaCT++ description logic reasoner," in *Proc. IJCAR*, 2006, pp. 292–297.
32. V. R. Pratt, "Semantical considerations on Floyd–Hoare logic," in *Proc. FOCS*, 1976, pp. 109–121.
33. M. J. Fischer and R. E. Ladner, "Propositional dynamic logic of regular programs," *J. Comput. Syst. Sci.*, vol. 18, no. 2, pp. 194–211, 1979.
34. D. Harel, D. Kozen, and J. Tiuryn, *Dynamic Logic*. Cambridge, MA, USA: MIT Press, 2000.
35. C. A. R. Hoare, "An axiomatic basis for computer programming," *Commun. ACM*, vol. 12, no. 10, pp. 576–580, 1969.
36. E. W. Dijkstra, "Guarded commands, nondeterminacy and formal derivation of programs," *Commun. ACM*, vol. 18, no. 8, pp. 453–457, 1975.
37. P. Blackburn and J. Seligman, "Hybrid languages," *J. Logic, Language and Information*, vol. 4, no. 3, pp. 251–272, 1995.
38. D. M. Gabbay, *Fibring Logics*. Oxford, U.K.: Oxford Univ. Press, 1999.
39. ISO/IEC, *Information Technology – Common Logic (CL): A Framework for a Family of Logic-Based Languages*, ISO/IEC 24707:2007.
40. M. R. Genesereth and R. E. Fikes, *Knowledge Interchange Format (KIF), Version 3.0*, Tech. Rep., Stanford Univ., 1992.
41. I. Horrocks et al., "SWRL: A Semantic Web Rule Language combining OWL and RuleML," W3C Member Submission, 2004.
42. J. Jaffar and M. J. Maher, "Constraint logic programming: A survey," *J. Logic Programming*, vols. 19–20, pp. 503–581, 1994.
43. T. Frühwirth, *Constraint Handling Rules*. Cambridge, U.K.: Cambridge Univ. Press, 2009.
44. S. Demri and P. Schnoebelen, "The complexity of temporal logic model checking," in *Advances in Modal Logic*, 2002.
45. E. A. Emerson, *"Temporal and Modal Logic"*, in Handbook of Theoretical Computer Science, Vol. B, Elsevier, 1990, pp. 995–1072.