Mathematical Foundations for Cross-Standard Requirements Decomposition

Seven established mathematical frameworks can rigorously formalize the decomposition of heterogeneous safety standards into atomic requirements ("Forms") and enable cross-standard comparison. The most applicable combination is **Formal Concept Analysis** for discovering requirement clusters and implications, **Description Logics** for computing subsumption hierarchies and equivalences, and **Institution Theory** for preserving semantics across heterogeneous formalisms. Each framework offers distinct formal machinery; this report provides the core definitions, foundational citations, and precise applicability assessments.

Formal Concept Analysis provides the structural backbone

Rudolf Wille's Formal Concept Analysis (FCA) offers the most directly applicable framework for decomposing standards into atomic requirements and discovering their relationships. The foundational work is **Wille (1982), "Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts," in Ordered Sets, NATO Science Series C, Vol. 83, pp. 445–470, Reidel**, Grokipedia with the definitive treatment in **Ganter & Wille (1999), Formal Concept Analysis: Mathematical Foundations, Springer**. SCIRP

A **formal context** is a triple **$\mathbb{K} := (G, M, I)$** where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is an incidence relation. Grokipedia For standards decomposition, G could be safety standards or integrity levels (IEC 61508, ISO 26262, ASIL-A through ASIL-D), and M the atomic requirements. The derivation operators $A' := \{m \in M \mid gIm \text{ for all } g \in A\}$ and $B' := \{g \in G \mid gIm \text{ for all } m \in B\}$ Wikipedia form an **antitone Galois connection** between power sets.

A **formal concept** is a pair (A, B) where A' = B and B' = A—the extent A contains all objects sharing exactly the attributes in intent B. Wikipedia The set of all concepts forms a **complete lattice** under the ordering $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 \Leftrightarrow B_1 \supseteq B_2$. Grokipedia This lattice structure directly captures requirement hierarchies: concepts at higher levels represent more general requirement clusters shared across more standards. IEEE Xplore

What FCA captures: Object-attribute relationships (standard-requirement incidence), hierarchical concept structure, attribute **implications** (if all standards requiring X also require Y, then $X \to Y$ holds), and maximal clusters of co-occurring requirements. The **canonical basis** (Guigues-Duquenne basis) provides an irredundant set of implications from which all valid dependencies can be derived.

What FCA does not capture: Subsumption between requirements themselves (only object-attribute relations), temporal/versioning aspects, quantitative thresholds (ASIL levels require conceptual scaling), and causal dependencies. Requirements must be preprocessed into binary form—"ASIL-D requires technique T with ≥99% coverage" must be scaled.

Description Logics formalize subsumption and equivalence

Description Logics (DL), the formal foundation of OWL, provide the precise semantics for requirement subsumption and equivalence relations. The authoritative reference is **Baader, Calvanese, McGuinness, Nardi & Patel-Schneider (eds.), The Description Logic Handbook, Cambridge University Press, 2003/2007**. OWL 2 DL corresponds to **SROIQ**, formalized in **Horrocks, Kutz & Sattler (2006), "The Even More Irresistible SROIQ," KR 2006, AAAI Press**. [W3C]

An **interpretation** I = (Δ^I, ·^I) consists of a non-empty domain and an interpretation function mapping concept names to subsets of Δ^I and role names to binary relations. Core constructors include conjunction (C ⊓ D)^I = C^I ∩ D^I, disjunction (C ⊔ D)^I = C^I ∪ D^I, existential restriction (∃R.C)^I = {x | ∃y: (x,y) ∈ R^I ∧ y ∈ C^I}, and universal restriction (∀R.C)^I = {x | ∀y: (x,y) ∈ R^I → y ∈ C^I}.

The critical relations for requirements are:

- **Subsumption:** C ⊑_T D iff C^I ⊆ D^I for every model I of TBox T
- **Equivalence:** C ≡_T D iff C^I = D^I for every model I of T

For requirements, if $R_1 \sqsubseteq R_2$, then any system satisfying R_1 automatically satisfies R_2—R_2 is a "weaker" requirement. This directly models "one requirement subsumes another." A knowledge base with **TBox** (terminological axioms like ASIL_D ⊑ SafetyRequirement) and **ABox** (assertions about specific requirements) enables automated **classification** (computing the subsumption hierarchy) and **consistency checking**.

What DL captures: Hierarchical classification, subsumption (directly models "R_1 implies R_2"), equivalence, automated reasoning for consistency and classification, and formal disambiguation of natural language requirements.

What DL does not capture: The **Open World Assumption** (OWA) means absence of information is not evidence of absence—problematic when "not specified means not required." **SHACL** (W3C 2017) provides closed-world validation to complement OWL. Computational complexity is **N2ExpTime-complete** for SROIQ; even ALC is **ExpTime-complete**. Quantitative constraints and temporal aspects require extensions. The W3C OWL 2 Direct Semantics specification (December 2012) provides the normative reference. [w3]

Institution Theory handles heterogeneous formalisms

For comparing standards expressed in different logical frameworks, Institution Theory provides the mathematically cleanest abstraction. The foundational reference is **Goguen & Burstall (1992), "Institutions: Abstract Model Theory for Specification and Programming," Journal of the ACM 39(1):95–146**, [utm] with comprehensive treatment in **Diaconescu (2008), Institution-independent Model Theory, Birkhäuser**.

An **institution** I = (Sign, Sen, Mod, ⊨) consists of:

- **Sign:** A category of signatures (vocabularies)
- **Sen:** Sign → **Set**, a functor assigning sentences to signatures

- **Mod:** $\mathrm{Sign}^{op} \to$ **Cat**, a contravariant functor assigning models
- **⊨:** A family of satisfaction relations $\models_\Sigma \subseteq |\mathrm{Mod}(\Sigma)| \times \mathrm{Sen}(\Sigma)$

The key coherence requirement is the **Satisfaction Condition**: for any signature morphism $\varphi: \Sigma \to \Sigma'$, any Σ-sentence ρ, and any Σ'-model M':

> $$\mathbf{Mod(\varphi)(M') \models_\Sigma \rho \iff M' \models_{\Sigma'} Sen(\varphi)(\rho)}$$

This states that **truth is invariant under change of notation**— utm translating a sentence forward and evaluating equals reducing the model backward and evaluating the original.

Institution comorphisms ($I \to I'$) enable embedding one logic into another while preserving satisfaction: a functor $\Phi: \mathrm{Sign} \to \mathrm{Sign}'$ with natural transformations translating sentences forward and models backward. utm The **Grothendieck construction** flattens a diagram of institutions connected by comorphisms into a single "heterogeneous institution." utm **CASL** (Common Algebraic Specification Language, Astesiano et al. 2002) and the **Heterogeneous Tool Set (Hets)** implement this for practical specification. utm

What institutions capture: Multiple logical systems with different syntaxes/semantics, signature translation preserving vocabulary mappings, semantic preservation via the satisfaction condition, and modular composition of specifications.

What institutions do not capture: Institutions are highly abstract—no built-in distinction between requirements, specifications, or standards. They require formalization of prose standards into logical sentences. Partial or fuzzy mappings are not native; morphisms are exact mathematical translations. The framework provides the mathematical foundation for heterogeneous comparison but significant formalization effort is required.

Galois Connections formalize abstraction hierarchies

The Galois connection framework, applied to program analysis by **Cousot & Cousot (1977), "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs," POPL, pp. 238–252**, provides rigorous machinery for abstraction-concretization relationships. The mathematical foundation dates to **Ore (1944), "Galois connexions," Trans. AMS 55, pp. 493–513**.

A **monotone Galois connection** between posets $(C, \sqsubseteq)$ and $(A, \sqsubseteq)$ consists of functions $\alpha: C \to A$ (abstraction) and $\gamma: A \to C$ (concretization) satisfying:

> $$\boldsymbol{\alpha(c) \sqsubseteq a \iff c \sqsubseteq \gamma(a)}$$

Key properties include: **inflationarity** ($c \sqsubseteq \gamma(\alpha(c))$—abstraction loses information), **deflationarity** ($\alpha(\gamma(a)) \sqsubseteq a$), and that $\gamma \circ \alpha$ is a **closure operator**. Crucially, **soundness** follows: any property proved at the abstract level holds at the concrete level.

For requirements, if IEC 61508 represents an abstraction of ISO 26262: α maps automotive-specific requirements to generic functional safety concepts, and γ interprets generic requirements in automotive context. Compliance with α(ISO 26262) implies compliance with the abstracted IEC 61508 requirements—the soundness property.

The **Knaster-Tarski theorem** (**Tarski 1955, Pacific J. Math. 5:285–309**) guarantees that monotone functions on complete lattices have complete lattices of fixed points, with constructively computable least and greatest fixed points. This enables iterative computation of stable requirement sets.

FCA internally uses Galois connections: the derivation operators form an antitone Galois connection, and concept lattices are the complete lattices of closed elements. Wikipedia

What Galois connections capture: Abstraction-concretization relationships, soundness preservation across abstraction levels, hierarchical refinement structures, and composable multi-level abstractions.

What they do not capture: Heterogeneous formalisms require compatible poset structures; there's no natural ordering between incomparable standards without explicit mapping. Non-monotonic relationships (exception handling, conditional requirements) and quantitative aspects (coverage percentages) are not modeled.

Feature Models decompose requirements with Boolean semantics

Feature modeling, introduced in **Kang et al. (1990), "Feature-Oriented Domain Analysis (FODA) Feasibility Study," SEI Technical Report CMU/SEI-90-TR-021**, provides a well-established framework for variability modeling with precise propositional semantics. The formal connection to propositional logic is established in **Batory (2005), "Feature Models, Grammars, and Propositional Formulas," SPLC 2005, LNCS 3714, pp. 7–20**.

A **feature diagram** FD = (F, r, DE, CE, λ) consists of features F, root r, decomposition edges DE, cross-tree constraints CE, and type function λ assigning mandatory/optional/or/xor. The **semantics** is the set of valid configurations: each feature f corresponds to Boolean variable f, the diagram encodes as propositional formula φ_FM, and a valid configuration is a satisfying assignment.

Translation rules map primitives to propositional logic:

- Mandatory child ($p \rightarrow c$): $p \leftrightarrow c$
- Optional child: $c \rightarrow p$
- OR-group: $p \leftrightarrow (c_1 \vee ... \vee c_n)$
- XOR-group: $p \leftrightarrow$ exactly-one(c_1,...,c_n)
- Requires: $A \rightarrow B$
- Excludes: $\neg(A \wedge B)$

Benavides, Segura & Ruiz-Cortés (2010), "Automated Analysis of Feature Models 20 Years Later," Information Systems 35(6):615–636 surveys automated analysis: void detection (is $[[FM]] = \emptyset$?), dead feature detection, product counting (#SAT), and commonality analysis—all reducible to SAT/SMT solving.

What feature models capture: Hierarchical decomposition, mandatory/optional distinctions, dependencies (requires) and conflicts (excludes), valid configuration enumeration, and consistency checking via SAT solvers.

What they do not capture: Semantic equivalence between requirements (no native support for $Req\_A \equiv Req\_B$ meaning "same thing"), subsumption relations, partial satisfaction, multi-standard composition, or versioning. Extension strategies include multi-root models with mapping constraints and compositional operators (merge, slice, diff) from the FAMILIAR framework.

Category Theory provides the mathematical meta-framework

Category theory underlies institution theory [utm] and provides the abstract framework for specification composition. The key reference for computing applications is **Goguen (1991), "A Categorical Manifesto," Mathematical Structures in Computer Science 1(1):49–67**, articulating five "dogmas" including: "To each species of mathematical structure corresponds a category whose objects have that structure, and whose morphisms preserve it."

A **category** C has objects Ob(C), morphisms Hom(A,B) for each pair, associative composition, and identity morphisms. **Functors** $F: C \rightarrow D$ map objects and morphisms, preserving composition and identities. **Natural transformations** $\eta: F \Rightarrow G$ relate functors via component morphisms satisfying naturality.

Colimits—especially **pushouts**—compute specification combination. Per Goguen's Fifth Dogma: "Given a category of widgets, the operation of putting a system of widgets together to form some super-widget corresponds to taking the colimit." [ox] For specifications sharing common signatures, the pushout computes the unified specification.

Adjunctions $L \dashv R$ with $Hom\_D(L(C), D) \cong Hom\_C(C, R(D))$ capture translation-embedding relationships. **Kan extensions** (per Mac Lane: "All Concepts Are Kan Extensions") compute "best fit" translations between categories.

CASL (Astesiano et al. 2002, TCS 286:153–196) implements categorical specification with basic, structured, and architectural layers. The **Heterogeneous Tool Set (Hets)** provides multi-logic proof management using institution comorphisms. [utm]

What category theory captures: Compositional structure via colimits, morphisms as refinement/translation, universal constructions, and logic independence through institution parameterization.

What it does not capture: Category theory is too abstract for comparing informal requirements documents. Natural language requirements lack obvious categorical structure. No inherent notion of "degree of difference" between specifications—morphisms either exist or don't.

Existing work on safety standards comparison identifies key gaps

Academic work specifically on cross-standard safety requirements mapping remains nascent. **Okoh & Myklebust (2024)** provide the most systematic mapping from ISO 26262 to IEC 61508 for hardware and software, identifying six dimensions for comparison (from Machrouh et al. 2012). A critical finding: "Currently, no criteria have yet been set by consensus on how to requalify elements originally developed according to ISO 26262 but intended to be reused in relation to IEC 61508." Taylor & Francis Online

Antonino et al. (2014), "The Safety Requirements Decomposition Pattern" defines **Atomic Functional Safety Requirements (AFSR)** as FSRs with only one failure cause versus **Composite Functional Safety Requirements (CFSR)** that can be decomposed—this directly addresses the "Forms" concept.

SACM (Structured Assurance Case Metamodel, OMG) provides a metamodel for representing assurance cases, with **Wei et al. (2019)** providing transformations from Goal Structuring Notation (GSN) and Claims-Arguments-Evidence (CAE). **SafeML** (Biggs, AIST) and **EAST-ADL** model safety information in SysML/UML profiles supporting ISO 26262 and DO-178C.

EARS (Easy Approach to Requirements Syntax) from **Mavin et al. (2009)** provides structured patterns enabling machine processing: "While <precondition>, When <trigger>, the <system> shall <response>." Adopted by Bosch, Rolls-Royce, NASA, and Siemens.

Key gaps requiring new research: no comprehensive formal model mapping all IEC 61508 family requirements to atomic units, no automated tools for computing requirement-level deltas between standards, no standard ontology for safety requirements across domains, and cross-standard traceability remains largely manual.

Framework selection matrix for the Forms system

Framework	Decomposition	Equivalence	Subsumption	Cross-Standard	Deltas	Best For
FCA	✓ (as attributes)	Via implications	✗ (objects only)	Via shared context	Via lattice difference	Discovering requirement clusters
Description Logic	Via TBox	✓ ($C \equiv D$)	✓ ($C \sqsubseteq D$)	Requires ontology alignment	Via unsat cores	Computing subsumption hierarchies
Institution Theory	Via sentences	Via isomorphism	Via morphisms	✓ (comorphisms)	Via kernel	Heterogeneous formalism translation
Galois Connections	Via closure	Via bijectivity	Via ordering	Requires compatible posets	Via residual	Abstraction level relationships

Framework	Decomposition	Equivalence	Subsumption	Cross-Standard	Deltas	Best For
Feature Models	✓ (features)	Via equivalence	Via constraints	Via multi-root + mapping	Via formula diff	Configuration validity

Recommended architecture: Use **FCA** to discover natural clusters of co-occurring requirements across standards and extract implications. Use **Description Logics** (OWL 2 with SHACL validation) to formally define subsumption and equivalence between atomic requirements, enabling automated reasoning. Use **Institution comorphisms** to translate between standards expressed in different logical frameworks while preserving satisfaction. Use **Feature Model** semantics for configuration validity checking and delta computation via propositional formula differencing.

The mathematical foundations are mature and well-cited. The primary engineering challenge is the formalization of prose safety standards into these frameworks—Antonino's AFSR/CFSR decomposition pattern and EARS structured syntax provide the bridge from natural language to formal representation.