A Model-Based Framework for Exploring Conflict Dynamics

Fazle Rabbi
fazle.rabbi@uib.no
Information Science and Media Studies.
University of Bergen
Norway

ABSTRACT

This paper introduces a novel framework for conflict analysis that leverages advanced visual modeling techniques. By employing comparative analysis, key variables influencing armed conflicts are identified and analyzed. The framework includes a meta-model representing domain concepts such as the goals and strategies of conflicting parties, escalating stages, and impacts of conflicts.

Conflict escalation is a complex process characterized by interactions between opposing parties. This paper presents a structured model that outlines how conflicts evolve and intensify over time. We adapt a meta-modeling framework called the Diagram Predicate Framework (DPF) to represent conflict-related concepts and extend it to support abstract view generation. This framework facilitates the analysis of conflict trends and the study of dynamics across various levels of abstraction.

A computational model based on category theory is proposed for trend analysis, enabling the extraction of patterns of conflict evolution and the comparison of strategies and goals at different escalation stages. Categorical operations such as pullback and limit construction are employed to compute conflict evolution and identify common structures among conflict instances, providing insights into conflict dynamics across diverse zones.

CCS CONCEPTS

• **Human-centered computing** → **Information visualization**; • **Computing methodologies** → **Knowledge representation and reasoning**; • **Software and its engineering** → **Dynamic analysis**; **Model-driven software engineering**.

KEYWORDS

Conflict analysis, Computational journalism, Category theory, Metamodeling

ACM Reference Format:
Fazle Rabbi. 2024. A Model-Based Framework for Exploring Conflict Dynamics. In *ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24), September 22–27, 2024, Linz, Austria.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3652620.3688206

MODELS Companion '24, September 22–27, 2024, Linz, Austria

ACM ISBN 979-8-4007-0622-6/24/09
https://doi.org/10.1145/3652620.3688206

1 INTRODUCTION

Media is one of the most important source to collect information for many people in modern society and journalists play the role of gatekeepers for their audiences. Journalists need to keep track of the ongoing events all around the world and analyze the events carefully as they need to inform their audience about the changing world. Journalists also have the responsibility to make politicians responsible about their actions. Therefore, journalists play an important role to shape our societies.

Since there is an abundance of news articles being published all over the world by several news media outlets, journalists would be benefited by techniques for systematically analyzing events. In addition to journalists, political scientists, UN peacekeepers are also in critical need for understanding trends in conflicts. In this paper we present a model based framework that employs graph theory and category theory for representing the knowledge in the domain and for representing computational methods for the analysis of events related to armed conflicts.

Conflicts are a common phenomenon in our society and they arise due to our nature as complex, multipurpose systems [7]. There are numerous reasons for different parties to have conflict with each other. While conflict is part of a democratic process they can turn violent specially in situations when the parties are polarized. When conflict becomes violent, it needs to be resolved to minimize catastrophes in the society. The formation of United Nations is to address these issues. There is no doubt that the United Nations has played an important role in resolving conflicts specially in situations when it came into the question of using Nuclear weapons. For instance, the Treaty on the Non-Proliferation of Nuclear Weapons (NPT) [2] is a key international agreement facilitated by the UN to prevent the spread of nuclear weapons and promote disarmament. However, there are also many cases where UN could not play an effective role.

Conflict analysis is a complex and dynamic process involving multiple stakeholders with divergent objectives. While covering conflicts, journalists report about the opinions of different groups involving conflicting parties as well as international organizations. Traditional approaches often fall short in capturing the evolving nature of conflicts. This paper presents an innovative method that synergistically combines multi-dimensional comparative analysis to enhance the strategic modeling of conflicts.

In [7], Castelfranchi formulated social or interpersonal conflicts those arise between two agents (e.g., two persons, two groups or teams, two companies, two states, or two institutions) with their own goals. He defined conflict as follows: *Any "entity" that is defined in terms of certain goals and functions in terms of goals and is aimed at realizing those goals can be in conflict with other analogous entities.* In his paper, Castelfranchi formulated social conflicts between agents in the following way:

> *If X simply has the goal that P, but in order to achieve P she must use resource R or achieve or maintain the condition that Q, but in the same world (interference) Y exists, who has the goal that W (or that Not P, or W that implies Not P) or needs to exploit R, which is not enough for both, or needs to eliminate or avoid that Q, perhaps they will ignore each other, but they will actually fight one other, they will compete for R or for (Not) Q. Either X will win, that is, she will simply realize her goal.*

Although this formulation captures the fundamental relationship of conflicting parties, it does not provide any computational method to analyze the dynamic nature of conflicts. In this paper we propose to use category theory for analyzing conflict escalation paths. Category Theory provides tools to map out the dynamics of a conflict by considering it as a series of transformations within a categorical structure. This helps in understanding how conflicts evolve over time and how different interventions might alter their trajectory. Our main contributions in this paper include:

- a domain ontology for representing conflict related terminologies;
- a formal approach for specifying abstraction level for trend analysis;
- a computational method for analysing conflict trends from various perspective using category theory.

We assume the reader is comfortable with the basics of category theory [5]. The paper is organized as follows. In section 2 we present a domain ontology and a meta-modeling formalism to represent conflicting situation at various abstraction level. We present a transformation mechanism that allows us to study conflicting situations from various perspective. In section 3 we present a computational model based on category theroy for trend analysis. In section 4 we provide a discussion about the proposed method and provide a comparison with existing works.

2 MODELING CONFLICT

We use Diagram Predicate Framework (DPF) [17] which is an extension of the Generalised Sketches formalism originally developed by Diskin et al. [8]. DPF provides a meta-modeling language to construct a metamodelling hierarchy in which a model at any level can be considered as a metamodel wrt. the models at the level below it. Models at each level in a metamodelling hierarchy are specified by a modelling language at the level above and conform to the corresponding metamodel. We use a 3 level metamodeling hierarchy (see Figure 1) where M_i at a certain level conforms to a metamodel M_{i+1} at the level above and the model at level M_2 has itself as metamodel (i.e., reflexive model).

In the DPF approach, models at any level are formalised as diagrammatic specifications which consist of type graphs annotated with diagrammatic constraints. Since conflicts involve various concepts that include hierarchical relationships we need to use a formalism that supports inheritacne relationships. To support inheritance relationships in diagrammatic specifications we adopt the use of graphs with inheritance relationships to be used as type graphs. We use the concept of $\mathfrak{I}$-graph as defined in [14].

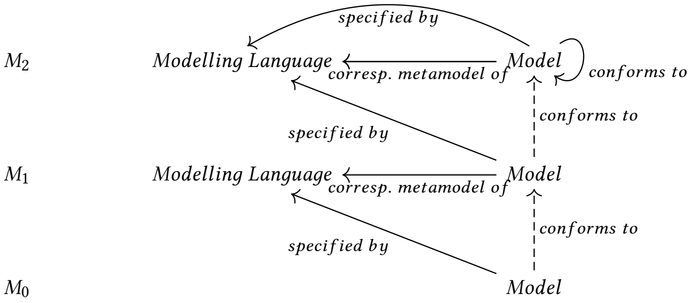

Figure 1: Generic pattern: modeling languages and metamodels (adapted from [17])

Definition 2.1 ($\mathfrak{I}$-graph). *A graph with inheritance, short $\mathfrak{I}$-graph, is given by $GI = (G, I)$. It consists of graph G and inheritance relation $I \subseteq G_V \; x \; G_V$, where for $v \in G_V$ $clan_I(v) = \{v' \in G_V \mid (v', v) \in I^*\}$ with I^* being the reflexive and transitive closure of I. A vertex $v' \neq v$ with $v' \in clan(v)$ is called a sub-vertex of v or more specifically as v $(v' \leq v)$, vertex v we call a super vertex of v'.*

Figure 2 shows the model at level M_2 which provides the typing information for the model at level M_1. The model consists of a shape graph G_2.

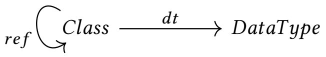

Figure 2: Model at level M_2

Table.1 shows a predicate signature $(P^{\Sigma_1}, \pi^{\Sigma_1})$ with two predicates. The predicates are typed by graph G_2 from metalevel M_2. The typing of the predicates are provided by a map $\tau^{\Sigma_1} : \Sigma_1 \longrightarrow G_2$. The notation $\Sigma_1 \triangleright G_2$: or $(P^{\Sigma_1}, \pi^{\Sigma_1}) \triangleright_{\tau^{\Sigma_1}} G_2$ will be used to the signature Σ_1 typed by G_2.

The $[mult(n, m)]$ predicate can be used to specify multiplicity constraints where n and m are used to define the lower and upper bound on the relationship (represented as a multi-valued function) between X and Y. The typing of the predicate specifies that the predicate can be used to add constraints to arrows which are typed by any of the arrows of type ref or dt. The $[COMP]$ predicate expresses compositional property among X, Y and Z. The semantic interpretation of $[COMP]$ is stated using set builder notation in the table.

We present a diagrammatic specification $\mathfrak{S}_{Conflict} = (S_{Conflict}, C^{\mathfrak{S}_{Conflict}} : \Sigma_1)$ for the domain model to represent conflicts. $(S_{Conflict})$ is an $\mathfrak{I}$-graph which represents the structure of the model. The predicates from a diagrammatic predicate signature Σ_1 are used to add constraints to this structure. Figure 3 shows the $\mathfrak{I}$-graph $S_{Conflict}$ which consists of terminologies to represent situations involving conflict. The inheritance relationships are represented by $subclassof$ relationships (shown as dotted arrows) in the figure. The circle shape in Figure 3 represent nodes those are typed

Table 1: A predicate signature Σ_1

p	$\pi^{\Sigma_1}(p)$	**Typing** $\tau^{\Sigma_1}(p)$	**Semantic Interpretation**
[mult(n,m)]	$X \xrightarrow{f} Y$	$X : Class \xrightarrow{f:ref} Y : Class$ $X : Class \xrightarrow{f:dt} Y : DataType$	$\forall x \in X : m \leq \lvert f(x) \rvert \leq n$, with $0 \leq m \leq n$, and $n \geq 1$.
[COMP]	$\begin{tikzcd} & X \arrow[ld, "g"'] \arrow[dd, "f"] \\ Z \arrow[rd, "h"'] & \\ & Y \end{tikzcd}$	$\begin{tikzcd} & X : Class \arrow[ld, "g:ref"'] \arrow[dd, "f:ref"] \\ Z : Class \arrow[rd, "h:ref"] & \\ & Y : Class \end{tikzcd}$	$\forall x \in X : f(x) = \bigcup\{h(z) \mid z \in g(x)\}$

by *Class* from model M_2; the nodes with yellow color represent nodes those are typed by *DataType* from model M_2. The specification is typed by the graph G_2 and the typing relationships are given by a typing graph homomorphism $\iota^S : S_{Conflict} \rightarrow G_2$ which assigns each element of $S_{Conflict}$ a type in G_2 such that $\forall (p, \delta) \in C^{\mathfrak{S}_{Conflict}} : \delta; \iota^S = \tau^{\Sigma_1}(p)$. We write $\mathfrak{S}_{Conflict} \triangleright G_2$ or $(S_{Conflict}, C^{\mathfrak{S}_{Conflict}} : \Sigma_1) \triangleright_{\iota^S} G_2$ to denote a conflict specification $\mathfrak{S}_{Conflict}$ typed by G_2.

The $\mathfrak{I}$-graph $S_{Conflict}$ represents a domain ontology for representing concepts related to armed conflict. The model includes concepts from *game theory* [10, 16], *argumentation framework* [4] and Friedrich Glasls *conflict escalation model* [9, 11]. The class *ArmedConflict* has been placed at the center of the model which will be used to represent real life conflict scenarios. This central class is connected to subsidiary classes and relations derived from diverse theoretical frameworks, providing a comprehensive structure for understanding the complexities inherent in conflicts. $S_{Conflict}$ includes various terminologies such as *Party*, *ConflictCause*, *ConflictImpact*, *ConflictStage* and resolution of conflicts. The model includes relationships between *Party* class and *Goal*, *Strategy* and *ConflictImpact*. Most of these classes have relationship with subclasses. The list of subclasses in $S_{Conflict}$ does not provide an exhaustive list to represent all possible conflicting situations in real-life.

In real-life scenario, conflicting parties have goals and to acheive their goals, parties often utilize strategies. Parties also have allies who provide support in achieving their goals. In this model we have incorporated concepts from Friedrich Glasl’s conflict escalation model. Glasl’s Conflict Escalation Model is a theoretical framework that provides insights into the dynamics and stages of conflict escalation. The model outlines a structured approach to understanding how conflicts evolve and intensify over time. The model is often used in conflict resolution studies, mediation processes, and organizational conflict management. Glasl’s model outlines nine stages of conflict escalation which are shown in the model as subclasses of *ConflictStage* class. His conflict escalation model outlines the sequential progression of conflicts and factors contributing to their escalation. It aids mediators and practitioners in de-escalating tensions and facilitating resolution before conflicts become destructive. Glasl’s model categorizes conflicts into three levels: Disagreement (Win-Win), Contest (Win-Lose), and Fight (Lose-Lose). These levels are shown in the model as subclasses of *EscalationLevel* class. Each level has distinct features and dynamics, aiding individuals and organizations in assessing conflicts and employing tailored resolution strategies. These models emphasize early intervention and proactive conflict management to prevent escalation and promote peaceful resolution by addressing underlying issues and restoring communication.

To resolve conflicts, conflicting parties reach agreements of various kinds, including *PeaceTreaty*, *Ceasefire*, *Negotiations*. These agreements are often facilitated by mediator parties, such as the United Nations. Typically these agreements include many details about conflicting goals and negotiations among the parties. We propose to use argumentation framework to present key information from agreements which include conflicting arguments involved among the parties including *TerritorialClaim*, *SecurityConcern*, etc. The conflicting arguments of the parties are related to the parties goals and they need to be addressed by the agreements made to resolve conflicts. However the arguments in the agreements may attack or support other arguments which can provide an overall idea about the agreements. In $S_{Conflict}$, class *AttackRelation* has been incorporated to represented attack and support relationships among agreements in an agreement.

Here we present an example instantiation of a peace agreement [1] signed between the parties in the first conflict of Nagorno Karabackh. Below is a list of instances of the subclasses of *Argument* class accompanied by a short description:

- **Argument A: SecurityConcerns** The ceasefire is necessary to halt immediate hostilities and save lives.
- **Argument B: MaintainingRegionalStability** The ceasefire leaves the status of Nagorno-Karabakh unresolved, which could lead to future conflicts.
- **Argument C: ControlOverTerritories** Russian involvement as a mediator is essential for the ceasefire’s success.
- **Argument D: ReturnOfTerritories** The ceasefire maintains Nagorno-Karabakh as a de facto independent entity, which is a victory for Armenian forces.
- **Argument E: OppositionToControl** The ceasefire maintains the territorial integrity of Azerbaijan, as Nagorno-Karabakh remains internationally recognized as part of Azerbaijan.
- **Argument F:Long-termStability** The lack of a comprehensive peace treaty means the ceasefire is only a temporary solution.

Argument B attacks **Argument A** by highlighting a significant drawback of the ceasefire - it does not provide a long-term resolution to the conflict. **Argument B** is focused on the long-term implications of the ceasefire. **Argument A** is supported by **Argument C**, emphasizing the role of Russian mediation.

Table.2 shows a set of constraint $C^{\mathfrak{S}_{Conflict}}$ on the conflict specification $\mathfrak{S}_{Conflict} \triangleright G_2$. The constraints have been used to specify the following requirements:

- Any instance of ArmedConflict must have at least two parties;
- Any instance of ArmedConflict must be related to at least one ConflictZone
- Any instance of ArmedConflict must have at least one ConflictCause
- Any instance of ArmedConflict must have at most one EscalationLevel
- Any instance of AttackRelation must have at least one source Argument and at least one target argument.

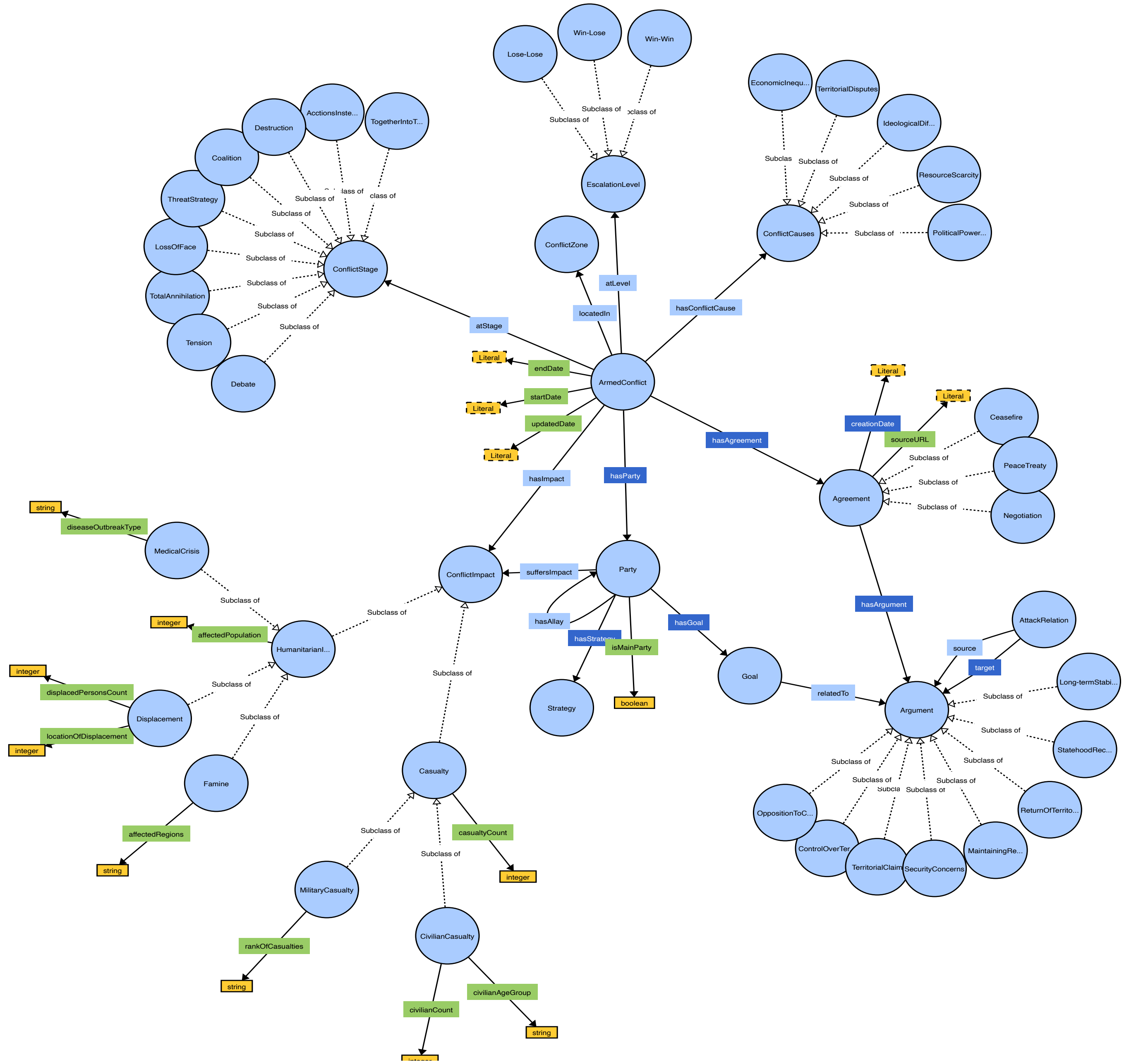

Figure 3: $\mathfrak{I}$-graph $S_{Conflict}$ representing concepts related to conflict

- The EscalationLevels are mutual exclusive
- Instances of ConflictImpact related to an ArmedConflict instance must be related to the Party instances of the ArmedConflict instance.

The shape graph $S_{Conflict}$ can be used to specify the typing of other models at a level below in the metamodeling hierarchy. We can use clan morphisms to specify the typing relationships between a graph at level M_0 in the metamodeling hierarchy to its model $S_{Conflict}$. In other words, any graph typed by the model $S_{Conflict}$ will be referred to as instance of $S_{Conflict}$. Clan morphisms are based on graph morphisms, but weaken the homomorphism condition. Hence, every graph morphism is also a clan morphism. In particular, clan morphisms allow to type edges connecting two vertices by edges connecting super vertices of their current source and target type. The definition of Clan morphism is given below [14]:

Definition 2.2 (Clan morphism). *Given graph $G1$ and $\mathfrak{I}$ − graph $GI2 = (G2, I2)$, a pair of mappings $f = (f_V, f_E) : G1 \rightarrow G2$*

Table 2: A set of Constraint $C^{\mathfrak{S}_{Conflict}}$

(p, δ)	$\pi^{\Sigma_1}(p)$	$\delta(\pi^{\Sigma_1}(p))$
$([mult(2,*)], \delta_1)$	$X \xrightarrow{f} Y$	$ArmedConflict \xrightarrow{hasParty} Party$
$([mult(1,*)], \delta_2)$	$X \xrightarrow{f} Y$	$ArmedConflict \xrightarrow{locatedIn} ConflictZone$
$([mult(1,*)], \delta_3)$	$X \xrightarrow{f} Y$	$ArmedConflict \xrightarrow{hasConflictCause} ConflictCause$
$([mult(0,1)], \delta_4)$	$X \xrightarrow{f} Y$	$ArmedConflict \xrightarrow{atLevel} EscalationLevel$
$([mult(1,*)], \delta_5)$	$X \xrightarrow{f} Y$	$AttackRelation \xrightarrow{source} Argument$
$([mult(1,*)], \delta_6)$	$X \xrightarrow{f} Y$	$AttackRelation \xrightarrow{target} Argument$
$([COMP], \delta_7)$	$\begin{tikzcd} & X \arrow[ld, "g"'] \arrow[dd, "f"] \\ Z \arrow[rd, "h"] & \\ & Y \end{tikzcd}$	$\begin{tikzcd} & ArmedConflict \arrow[ld, "hasParty"'] \arrow[dd, "hasImpact"] \\ Party \arrow[rd, "suffersImpact"'] & \\ & ConflictImpact \end{tikzcd}$

is called clan-morphism, written $f : G1 \longrightarrow GI2$, *if* $\forall e1 \in G1_E : (src^{G1}; f_V)(e1) \in clan_{I2}((f_E; src^{G2})(e1)) \wedge (trg^{G1}; f_V)(e1) \in clan_{I2}((f_E; trg^{G2})(e1))$.

Let us assume that $\iota : I \longrightarrow S_{Conflict}$ is an instance of $S_{Conflict}$ (also represented as (I, ι)) where $S_{Conflict}$ is the underlying graph of specification $\mathfrak{S}_{Conflict}$. In order to be a valid instance of $\mathfrak{S}_{Conflict}$, I must be typed by $S_{Conflict}$ and must satisfy all the constraints $C^{\mathfrak{S}_{Conflict}}$. The typing relationships ι is specified by clan morphism.

The constraint satisfaction relationships of an instance to its specification can be checked by using pullback operation for each constraints. Figure 4 shows an example of conducting pullback operation for constraint δ_1. By performing a pullback of $\pi^{\Sigma_1}([mult(2,*)]) \rightarrow S_{Conflict} \leftarrow I$, we extract the fragment of the graph that is affected by the multiplicity constraint δ_1. In order to be a valid instance of $\mathfrak{S}_{Conflict}$, the clan morphism ι^* (see Figure 4) needs to satisfy the semantic interpretation of $[mult(2,*)]$ e.g., all instances of ArmedConflict must have at least 2 relationships with the instances of Party.

$$\begin{tikzcd}
{[mult(2,*)]} \arrow[r, "\delta_1"] & S_{Conflict} \\
O^* \arrow[u, "\iota^*"] \arrow[r, "\delta_1^*"] & I \arrow[u, "\iota"']
\end{tikzcd}$$

Figure 4: Pullback operation for checking constraint satisfaction

By an *instance of conflict* we refer to an instance of $S_{Conflict}$ that consists of exactly one instance of *ArmedConflict* and its relationships. An *instance of conflict* encapsulates information concerning a conflictual situation involving various parties over a specific timeframe. For example, it may represent the strategies, goals and developmental stages characterizing a conflict at the [Win-Win] level. If the conflict is escalated or de-escalated, a new *instance of conflict* would be used to reflect the updated escalation level [Win-Lose] or [Lose-Lose], incorporating updated information about the stages, strategies, and the impact of the conflict. While it is not obligatory to create a new *instance of conflict* for every escalation in a conflict zone, it is essential to do so whenever a significant change occurs in the conflict situation.

A knowledge graph is used to store the instances of conflict. While storing information in the knowledge graph we label the nodes and arrows in such a way that the knowledge graph can be considered as a disjunctive union of conflict instances. Unique identifier is used for each conflict instance and it allows us to store and retrieve information from the knowledge graph. This is required as there are several concepts being reused over various conflicting situations. For example, a party ($P_1 : Party$) may be involved in two different conflicts while their strategies might be different in the conflicts where they are involved in. If there are n number of conflict instances $(I_1, \iota_1), (I_2, \iota_2), (I_3, \iota_3)..(I_n, \iota_n)$ in a knowledge graph, we use the following commutative diagram to represent the relationships among the instances to the knowledge graph. Using category theory concepts, the knowledge graph is an object $\mathfrak{K}_{[+]}$ together with injections $k_1 : I_1 \longrightarrow \mathfrak{K}_{[+]}, k_2 : I_2 \longrightarrow \mathfrak{K}_{[+]}, ..k_n : I_n \longrightarrow \mathfrak{K}_{[+]}$ is said to be a sum(coproduct).

$$\begin{tikzcd}
S_{Conflict} & \\
\mathfrak{K}_{[+]} \arrow[u, "\iota_{\mathfrak{K}}"] & I_i \arrow[l, "k_i"] \arrow[ul, "\iota_i"] \arrow[ul, phantom, "{[=]}"']
\end{tikzcd}$$

Figure 5: Knowledge graph represented as a sum(coproduct) of instances $\alpha_i (i \in \mathbb{N})$

2.1 Viewpoint Specification

We reuse the concept of diagrammatic specification to define viewpoints which will be used to generate abstract view from graphs. Viewpoint specifications include 4 predicates as shown in signature Σ_{View} in Table.3. A Model constrained with these predicates (will be referred to as viewpoint specification) are used by a transformation algorithm to carry out the transformation of conflict instances to appropriate view.

Table 3: A predicate signature $\Sigma_{View} \triangleright G_2$

p	$\alpha^{\Sigma_{View}}(p)$	**Proposed Visualization and Typing**
[View-Instance]	1	$X : Class^{[V-I]}$
$[View - ChildNode]$	1	$X : Class^{[V-C]}$
$[View - Node]$	1	$X : Class^{[V-N]}$
$[View - ParentNode]$	1	$X : Class^{[V-P]}$

The purpose of these predicates is to annotate nodes in a model to indicate how their instances will be treated for generating views. The semantic interpretation of the predicates in signature $\Sigma_{View} \triangleright G_2$ is given below:

(1) The predicate $[View - Instance]$ would indicate that actual instances will be included in the view.
(2) The predicate $[View - ChildNode]$ would indicate that the name of the child node will be used in the view instead of actual instances.

(3) The predicate $[View - ParentNode]$ would indicate that the name of the parent node will be used in the view instead of actual instances.
(4) The predicate $[View - Node]$ would indicate that the name of the current node will be used in the view instead of actual instances.

Figure.6 shows an example viewpoint specification $\mathfrak{S}^{View} = (G_1, C^{\mathfrak{S}^{View}} : \Sigma_{View})$ at the top where the nodes are annotated with predicates from Σ_{View}. Dotted arrows in the model represent inheritance relationships. A sample instance G_0 is shown in the bottom of the figure where curved dotted arrows represent typing relationships of the node instances. The purpose of these annotations are reflected in the sample view shown in Figure.7. In Figure.7, $a : A$, $b : B$, $h : H$, $f : F$ are included as A, B, H, F are annotated with predicate $[View - Instance]$. Instance $l : L$ has been replaced with a node (L') as class G in G_1 is annotated with predicate $[View - ChildNode]$. The node L' in the view instance has a typing morphism to the node L in the type graph. Instances $m : M$ and $n : N$ have been replaced with a node (J') (a copy of J) as node K in G_1 is annotated with predicate $[View - ParentNode]$ and the parent class is J; Instance $q : Q$ has been replaced with a node (O') (a copy of O) as node O in G_1 is annotated with predicate $[View - Node]$. The nodes J' and O' in G_0 have typing morphisms to corresponding nodes J and O in G_1. The edges in Figure.7 have also been updated to reflect the changed being made in the view. Any instance of an edge stays the same in the view if both of its source and target nodes are preserved in the view. However, if any of the source or target of an edge has been modified in the view by a copy of the type node from the model, then an edge is added in the view that connects the copy of the type node(s). Note that when we replace some of the instances with the type of instances or their super class name, the abstract view may consist of fewer number of nodes and edges, for example, in Figure.7, two instances $n : N$ and $m : M$ have been replaced with node J' and two edges $r31 : e3$ and $r32 : e3$ have been replaced with one edge $e3' : e3$ (a copy of $e3$) which connects $a : A$ and the newly created node $J' : J$.

Viewpoint specifications should provide unambiguous viewing constraints i.e., should not provide different kinds of viewing requirement in the same inheritance hierarchy. For example, if a class A is annotated with [View-Node] while its subclass B is annotated with [View-Instance], then an instance of $b : B$ would have 2 different viewing requirement: one requires to incorporate $b : B$ in the view, while the other requires to incorporate A' (copy of A) in the view. In order to prevent any unambiguous viewing constraint for viewpoint specifications, we apply negative universal constraints.

Definition 2.3 (Negative Universal Constraints). *Given a typed signature $\Sigma \triangleright G = ((P^{\Sigma}, \alpha^{\Sigma}) \triangleright_{\tau^{\Sigma}} G)$, a negative universal constraint is a typed specification morphism $n : \mathfrak{L} \triangleright G \longrightarrow \mathfrak{R} \triangleright G$ with $\mathfrak{L} \triangleright G = ((L, C^{\mathfrak{L}} : \Sigma) \triangleright_{\iota^L} G)$ and $\mathfrak{R} \triangleright G = ((N, C^{\mathfrak{R}} : \Sigma) \triangleright_{\iota^N} G)$.*

Definition 2.4 (Satisfaction of Negative Universal Constraints). *A typed specification $\mathfrak{S} \triangleright G = ((S, C^{\mathfrak{S}} : \Sigma) \triangleright_{\iota S} G)$ satisfies a negative universal constraint $n : \mathfrak{L} \triangleright G \longrightarrow \mathfrak{R} \triangleright G$ iff for any injective typed specification morphism $m : \mathfrak{L} \triangleright G \longrightarrow \mathfrak{S} \triangleright G$ there does not exist an injective typed specification morphism $q : \mathfrak{R} \triangleright G \longrightarrow \mathfrak{S} \triangleright G$ such that $n; q = m$.*

$$\begin{tikzcd}
 & G & \\
 & S \arrow[u, "\iota_S"'] & \\
 & \neq & \\
L \arrow[uuur, bend left, "\iota_L"] \arrow[uur, "m"] \arrow[rr, "n"] & & N \arrow[uuul, bend right, "\iota_N"'] \arrow[uul, "q"']
\end{tikzcd}$$

Table.4 shows a universal constraint for the inheritance hierarchies in specification $\mathfrak{S}_{View}$. $[p_i]$ and $[p_j]$ represent predicates from signature Σ_{View}.

Table 4: Universal constraint for inheritance hierarchies

$\mathfrak{L} \triangleright G_2$	$\mathfrak{R} \triangleright G_2$
$1 : Class$ ↑ $2 : Class^{[p_i]}$	$1 : Class^{[p_j]}$ ↑ $2 : Class^{[p_i]}$

Definition 2.5 (Conforms to Viewpoint). *Given a viewpoint specification $\mathfrak{S}_{View} \triangleright G_2 = ((S, C^{\mathfrak{S}_{View}} : \Sigma_{View} \triangleright G_2) \triangleright_{\iota S} G_2)$ and an instance (I_i, ι_i) of S, (I_i, ι_i) conforms to $\mathfrak{S}_{View}$ if ι_i is a clan morphism and (I_i, ι_i) respects the constraints in $C^{\mathfrak{S}_{View}}$ according to the semantic interpretations of the view predicates in $\Sigma_{View} \triangleright G_2$.*

2.2 Transformation to abstract view

The purpose of the transformation is to transform an instance of a conflict to an appropriate abstraction level. Below is an algorithm that processes an instance '(I, ι)' of a graph S according to a view specification $\mathfrak{S}_{View} = (S, C^{\mathfrak{S}_{View}} : \Sigma_{View})$. The goal is to transform '(I, ι)' into '(I^{View}, ι^{View})', a graph where some nodes and edges are replaced with types or supertypes from S according to the abstraction level specified in $\mathfrak{S}_{View}$. We present a model transformation algorithm in Algorithm.1 which takes a view specification $\mathfrak{S}_{View} = (S, C^{\mathfrak{S}_{View}} : \Sigma_{View})$ and an instance '(I, ι)' of S as input. Here ι is a clan morphism from I to $\mathfrak{I}$-graph S. The algorithm produces an abstract view (I^{View}, ι^{View}) where the nodes and edges are represented according to $\mathfrak{S}_{View}$.

Theorem 1. *Let (I^{View}, ι^{View}) is an abstract view resulted by the processing of Algorithm.1 from an instance (I, ι) of an $\mathfrak{I}$-graph S and a view specification $\mathfrak{S}_{View} = (S, C^{\mathfrak{S}_{View}} : \Sigma_{View})$, then ι^{View} is a clan morphism from I^{View} to S.*

Proof. The algorithm produces a pair of mappings $\iota^{View} = (\iota_V^{View}, \iota_E^{View})$. In the following paragraphs we discuss about the validity of these two mappings and the justification of ι^{View} for a clan-morphism.

About the validity of ι_V^{View}: Let us assume that node $m \in I_V^{View}$ represents a view of a node $n \in I_V$ where $n \in I_V$ is typed by $t \in S$. In that case, node m must be typed by either t or any of its super

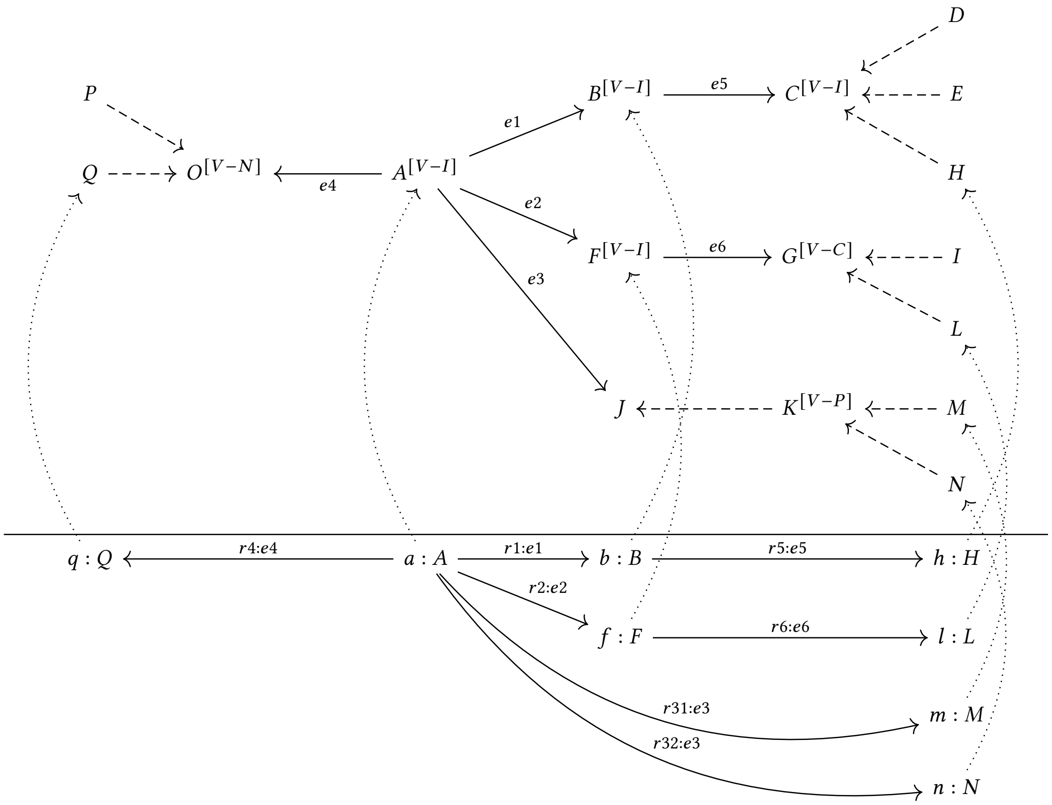

Figure 6: (Top) An example viewpoint specification $\mathfrak{S}^{View} \triangleright G_2 = (G_1, C^{\mathfrak{S}^{View}} : \Sigma_{View} \triangleright G_2)$; (Bottom) A sample instance G_0

$$\begin{tikzcd}
O' : O & a : A \arrow[l, "r4':e4"'] \arrow[r, "r1:e1"] \arrow[dr, "r2:e2"] \arrow[ddr, "e3':e3"'] & b : B \arrow[r, "r5:e5"] & h : H \\
 & & f : F \arrow[r, "r6':e6"] & L' : L \\
 & & J' : J &
\end{tikzcd}$$

Figure 7: A transformed model from Figure.6 representing a viewpoint of the sample instance G_0 according to the viewpoint specification $\mathfrak{S}^{View} \triangleright G_2$

vertex from S_V (according to the view specification $\mathfrak{S}^{View}$ and the implementation in the first for loop in the Algorithm). Since the view specification $\mathfrak{S}^{View}$ is constrained by a universal negative application condition to prevent any ambiguous view constraint to be assigned on the same inheritance hierarchy, node m can only be typed by any node from S_V which is a super vertex of $\iota^I(n)$.

About the validity of ι_E^{View}: Let us assume that edge $(m_1 \xrightarrow{h} m_2) \in I_E^{View}$ represents a view of an edge $(n_1 \xrightarrow{f} n_2) \in I_E$ where f is typed by $e \in S_E$. In that case, h is also typed by e and $\iota^{View}(m_1) \in clan(src^{I_E}(e))$ and $\iota^{View}(m_2) \in clan(trg^{I_E}(e))$ (according to the Algorithm). Therefore, ι^{View} for a clan-morphism.

□

3 TREND ANALYSIS OF CONFLICTS

In this section, we present computational models based on category theory for trend analysis of conflicts which include extracting patterns of conflict evolution. Category theory allows us to understand the interdependencies of conflicts from a higher level of abstraction. Using category theory, the complex interdependencies and interactions between different stakeholders' arguments and strategies can be modeled more abstractly and comprehensively which helps us in identifying key elements of conflicts and categorizing them. We present a categorical approach to compare and correlate various trends of patterns of conflicts which will essentially allow us to get an insightful knowledge about conflict resolution strategies as well as investigate coherent resolution plans. Below is a list of use cases for conflict analysis which is intended to be supported by our proposed computational models:

Example use-cases:

- Given a goal g of a party p, what strategies are commonly played by p and its opponents.
- Comparison of escalations among conflicts where party p_1 has a goal g_1 and party p_2 has a goal g_2.
- What are the common and unique strategies of a conflict zone?

```
Data: A view specification $\mathfrak{S}_{View} = (S, C^{\mathfrak{S}_{View}} : \Sigma_{View})$
      and an instance '$(I, \iota)$' of $S$ where $\iota$ is a clan
      morphism from $I$ to $\mathfrak{I}$-graph $S$
Result: Transformed Graph $(I^{View}, \iota^{View})$ where the nodes
      and edges are represented according to $\mathfrak{S}_{View}$
Initialize $I^{View}$: Create an empty graph that will be the
  transformed version of $(I, \iota)$.
for $n \in I_V$ do
    $T \leftarrow \iota(n); x \leftarrow copy(n); t \leftarrow copy(T)$;
    if $T$ or any of its super vertex is constraint with predicate
      [View-Instance] then
        /* add a copy of $n : T$ from $I$ to $I^{View}$ */
        $I_V^{View} \leftarrow I_V^{View} \cup \{x\}; view(n) \leftarrow x$;
        $\iota^{View}(x) \leftarrow T$;   /* The image is in $S$ */
    end
    if $T$ has a super vertex $T_i$ and $T_i'$s supervertex is
      constraint with predicate [View-ChildNode] then
        /* add $copy(T_i)$ (with typing morphism to
           node $T_i$ in $S$) in $I^{View}$ */
        $t_i' \leftarrow copy(T_i); I_V^{View} \leftarrow I_V^{View} \cup \{t_i\}$;
          $view(n) \leftarrow t_i; \iota^{View}(t_i) \leftarrow T_i$
    end
    if $TP$ is a super vertex of $T$ and $TP$ is constraint with
      predicate [View-Node] then
        /* add $copy(TP)$ (with typing) in $I^{View}$ */
        $tp \leftarrow copy(TP); I_V^{View} \leftarrow I_V^{View} \cup \{TP\}$;
          $view(n) \leftarrow tp; \iota^{View}(tp) \leftarrow TP$
    end
    if $T$ or any of its super vertex is constraint with predicate
      [View-ParentNode] and $TP$ is the parent node of that
      constraint node then
        /* add $copy(TP)$ (with typing) in $I^{View}$ */
        $tp \leftarrow copy(TP); I_V^{View} \leftarrow I_V^{View} \cup \{tp\}$;
          $view(n) \leftarrow tp; \iota^{View}(tp) \leftarrow TP$
    end
end
for $f \in I_E$ do
    $e \leftarrow \iota(f); n_1 \leftarrow src^I(f); n_2 \leftarrow trg^I(f)$;
    $m_1 \leftarrow view(n_1); m_2 \leftarrow view(n_2)$ if an edge
      $(m_1 \xrightarrow{:e} m_2) \notin I_E^{View}$ then
        /* Check for typing */
        if $\iota^{View}(m_1) \in clan(src^{I_E}(e))$ and
          $\iota^{View}(m_2) \in clan(trg^{I_E}(e))$ then
            /* add a copy of $f$ in $I^{View}$. Include a
               typing morphism to $e \in S_E$ */
            $h \leftarrow copy(f); I_E^{View} \leftarrow I_E^{View} \cup \{h\}$;
              $src^{I^{View}}(h) \leftarrow m_1; trg^{I^{View}}(h) \leftarrow m_2$;
            $\iota^{View}(h) \leftarrow e$;   /* The image is in $S$ */
        end
    end
end
```

Algorithm 1: Model Transformation Algorithm to Generate Abstract View

- Comparison between strategies of involved parties at different stages of escalation.
- Which strategies commonly lead to high number of casualties?
- Given a goal g of a party p and an agreement d, how the strategies of involved parties evolve over time?
- What are the common mediating patterns when the involved parties have a certain goal g and certain background of using strategy s?

The variables g, p, s, d for goals, parties, strategies, arguments used in the use-case scenarios may refer to concrete real life instances (i.e., instances of *Goal*, *Strategy*, etc) or the name of classes and subclasses from $S_{Conflict}$. In our approach, we propose to encode the requirements from the use-case scenarios into view specifications using predicates from Σ_{View}. In order to study the dynamic nature of conflicts we present a notion of state space for conflicts:

Definition 3.1 (Conflict State Space). *A conflict state space is defined by a tuple $CS = (\mathfrak{C}, R)$ where:*

- *$\mathfrak{C}$ is a non-empty set of conflict states. Each state ($cs \in \mathfrak{C}$) is an instance of conflict and represents a distinct scenario involved in a conflict.*
- *R is a binary relation over $\mathfrak{C}$, represents the evolution of a conflict state. The evolution relation captures the possible transitions or evolutions between conflict states.*

The conflict state space definition allows us to model conflicts as dynamic systems with different possible states and transitions between these states. The evolution relation (R) reflects how conflict states can evolve over time. The evolution relation (R) is transitive, therefore, if an instance of conflict (aka. conflict state) α_3 has evolved from α_2 and α_2 has evolved from α_1, then α_3 has evolved from α_1. If α_2 has evolved from α_1 (i.e., $(\alpha_1, \alpha_2) \in R$), we can say that α_2 is a direct successor of α_1. Conflict states can be also be transformed into abstract viewpoints based on view specifications. Figure 8 illustrates the evolution of a conflict state from α_i to α_j and their transformation into viewpoints α_i^{View1} and α_j^{View2}. There exists injective morphisms from α_i and α_j to the knowledge graph $\mathfrak{K}_{[+]}$. If we consider a view specification $\mathfrak{S}_{View1}$ and transform $\mathfrak{K}_{[+]}$ into a viewpoint $\mathfrak{K}_{[+]}^{View1}$, then $\mathfrak{K}_{[+]}^{View1}$ would be a sum(coproduct) of all the abstract views of conflict instances from $\mathfrak{K}_{[+]}$. $\mathfrak{K}_{[+]}^{View} = \alpha_1^{View} + \alpha_2^{View} + \alpha_3^{View} .. + \alpha_n^{View}$

To calculate the similarities between two conflict instances α_i and α_j, we propose to use pullback operation as shown in Figure 9. $\mathfrak{K}_U$ represents a union of all the conflict instances from knowledge graph $\mathfrak{K}_{[+]}$. Same entities from conflict instances would appear only once in $\mathfrak{K}_U$. Transformation of the objects to their abstract viewpoints with respect to a view specification are represented by wiggly arrows in the figure. The transformed abstract viewpoint of $\mathfrak{K}_U$ essentially represent an union of all the abstract viewpoint of conflict instances as shown below:

$$\mathfrak{K}_U^{View} = \alpha_1^{View} \cup \alpha_2^{View} \cup \alpha_3^{View} .. \cup \alpha_n^{View}$$

Figure 9 shows two pullback objects $c = \alpha_i \; X_{\mathfrak{K}_\cup} \; \alpha_j$ and $c' = \alpha_i^{View} X_{\mathfrak{K}_\cup^{View}} \; \alpha_j^{View}$ and their relationships. The figure also shows that there must exist an unique morphism k from c' to the abstract viewpoint of c.

$$\begin{tikzcd}
 & & \mathfrak{S}_{View1} \arrow[rrr, Rightarrow, "view\text{–}update"] & & & \mathfrak{S}_{View2} \\
 & \mathfrak{R}^{View1}_{[+]} \arrow[ur, dashed, "conformsTo"] & & & \mathfrak{R}^{View2}_{[+]} \arrow[ur, dashed, "conformsTo"] & \\
\mathfrak{R}_{[+]} \arrow[ur, squiggly, "transformation"] \arrow[rrr, "id_{\mathfrak{R}}"] & & & \mathfrak{R}_{[+]} \arrow[ur, squiggly, "transformation"] & & \\
 & & \alpha^{View1}_j \arrow[uul, hook] \arrow[uuu, "conformsTo"] \arrow[rrr, Rightarrow, "evolve"] & & & \alpha^{View2}_j \arrow[uul, hook] \arrow[uuu, "conformsTo"] \\
\alpha_i \arrow[uu, hook] \arrow[urr, squiggly, "transformation"] \arrow[rrr, Rightarrow, "evolve"] & & & \alpha_j \arrow[uu, hook] \arrow[urr, squiggly, "transformation"] & &
\end{tikzcd}$$

Figure 8: Conflict evolution and its transformation to abstract representation

$$\begin{tikzcd}
 & \alpha^{View}_i \arrow[dl, hook, "g'"'] & & \\
\mathfrak{R}^{View}_{\cup} & & c' \arrow[ul, "q'_2"'] \arrow[dl, "q'_1"] \arrow[r, "\exists!k"] \arrow[l, phantom, "<"] & c^{View} \arrow[ull, "p'_2"'] \arrow[dll, "p'_1"] \\
 & \alpha^{View}_j \arrow[ul, hook', "f'"] & & \\
 & \alpha_i \arrow[uuu, squiggly, bend left] \arrow[dl, hook, "g"'] & & \\
\mathfrak{R}_{\cup} \arrow[uuu, squiggly, bend left] & & & c \arrow[uuu, squiggly, bend right] \arrow[ull, "p_2"'] \arrow[dll, "p_1"] \arrow[l, phantom, "<"] \\
 & \alpha_j \arrow[uuu, squiggly, bend right] \arrow[ul, hook', "f"] & &
\end{tikzcd}$$

Figure 9: Finding commonality between conflict instances and their viewpoints using pullback

$$\begin{tikzcd}
 & & c_{m_{11}} \arrow[dl, bend right, "q_0"'] \arrow[dr, bend left, "p_0"] \arrow[d, phantom, "\vee"] & & \\
 & \Xi^{View}_1 \arrow[ddr, hook, "g_1"] & {} & \Gamma^{View}_1 \arrow[ddl, hook', "f_1"'] & \\
c_{l_{12}} \arrow[ur, "q_1"] \arrow[dr, "q_2"'] \arrow[r, phantom, ">"] & {} & c_{m_{22}} \arrow[dl, bend right, "q_3"'] \arrow[dr, bend left, "p_3"] \arrow[d, phantom, "\vee"] & {} & c_{r_{12}} \arrow[ul, "p_1"'] \arrow[dl, "p_2"] \arrow[l, phantom, "<"] \\
 & \Xi^{View}_2 \arrow[r, hook, "g_2"] & \mathfrak{R}^{View}_{\cup} & \Gamma^{View}_2 \arrow[l, hook', "f_2"'] & \\
c_{l_{23}} \arrow[ur, "q_4"] \arrow[dr, "q_5"'] \arrow[r, phantom, ">"] & {} & & {} & c_{r_{23}} \arrow[ul, "p_4"'] \arrow[dl, "p_5"] \arrow[l, phantom, "<"] \\
 & \Xi^{View}_3 \arrow[uur, hook, "g_3"] & {} & \Gamma^{View}_3 \arrow[uul, hook', "f_3"'] & \\
 & & c_{m_{33}} \arrow[ul, bend left, "q_6"] \arrow[ur, bend right, "p_6"'] \arrow[u, phantom, "\wedge"] & &
\end{tikzcd}$$

Figure 10: Evolution of two conflicting situations

We present another computational model in Figure 10 using categorical pullback operation to compute the evolution of conflicts in different conflict zones. Let us assume that Γ_1^{View}, Γ_2^{View}, Γ_3^{View} represent the unions of conflict instances in the first week, second week and third week respectively of a conflicting situations in conflict Zone-1. For example, if $\alpha_a^{View}, \alpha_b^{View}, \alpha_c^{View}$ are three conflict instances representing the situation in the first week of Zone-1, then $\Gamma_1^{View} = \alpha_a^{View} \cup \alpha_b^{View} \cup \alpha_c^{View}$. Similarly, Ξ^{View}, Ξ^{View}, Ξ^{View} represent the unions of abstract viewpoints of conflict instances in the first week, second week and third week of a conflicting situations in conflict Zone-2. Figure 10 illustrates a computational model to compare the development of these two different conflict zones over the weeks. With this computational model we are able to provide a comparison between strategies and/or goals of involved parties at different stages of escalation.

To find out common structures among different conflict zones, for example, information about commonly played strategies and goals in different conflicting zones we propose to apply limit operation. A specific analysis could entail identifying typical negotiation arguments employed when parties harbor security-related goals. Figure 11 illustrates two different conflicting situations represented as $\alpha_i^{View}(i \in \mathbb{N})$ and $\beta_j^{View}(j \in \mathbb{N})$ where the limit objects are shown as ϕ_1 and ϕ_2 and co-limit objects are shown as σ_1 and σ_2. With the limit construction we extract the common structure among the conflict instances. The pullback object $\Theta = \phi_1 X_{\mathfrak{R}_{\cup}^{View}} \phi_2$ represent commonality between the conflict instances. The overall structural differences between the conflict instances is computed by means of comparing the co-limit objects as shown in the figure with objects ψ_1 and ψ_2. With the structural differences we are able to extract information about unique structures (e.g., unique strategies played by parties) in different conflict zones. This computation model also allows to compute common mediating patterns (i.e., agreements) when the involved parties have a certain goal and certain background of using strategies.

4 DISCUSSION

In this paper, we introduce a formal method for conflict analysis that diverges significantly from conventional conflict analysis techniques. This innovative approach addresses longstanding challenges through a visual modeling language.

In computational journalism, various methods and systems have been developed for the analysis of news content such as GDELT[3], Event Registry[13], News Hunter[6]. These platforms excel in identifying and organizing news events from vast data streams in structured formats. However, despite these advancements in leveraging semantic knowledge bases like Wikidata for news content analysis, several significant challenges within the domain of news reporting and analysis remain unaddressed. To tackle these issues in a comprehensive manner, we propose a formal framework for modeling events related to armed conflict which aims to provide solutions to some of the open problems in the field and contributes to the ongoing development of robust techniques for computational journalism and conflict analysis.

Conflict analysis is a complex task, as it involves many parties with various kinds of interests. Formal methods enable us to capture, represent, and analyze these subtleties more effectively. While argumentation framework has been applied for conflict analysis [15, 18] using logic based approach, it has not been used for analyzing trends of armed conflict supporting multiple abstraction

$$\begin{tikzcd}
 & & & & & \Theta \arrow[dlll, "m"'] \arrow[drrr, "m'"] \arrow[d, phantom, "\vee"] & & & & & \\
 & & \phi_1 \arrow[dll, "u_1"'] \arrow[dl, "u_2"] \arrow[d, "u_3"] \arrow[dddrrr, "v"] & & & {} & & & \phi_2 \arrow[d, "u'_1"] \arrow[dr, "u'_2"] \arrow[drr, "u'_3"] \arrow[dddlll, "v'"'] & & \\
\alpha_1^{View} \arrow[drrrrr, hook, bend right=10, "f_1"] \arrow[ddr, "g_1"] \arrow[ddrrrrr, hook, bend right=10, "h_1"] & \alpha_2^{View} \arrow[drrrr, hook, bend right=10, "f_2"] \arrow[dd, "g_2"] \arrow[ddrrrr, hook, bend right=10, "h_2"] & \alpha_3^{View} \arrow[drrr, hook, bend right=10, "f_3"] \arrow[ddl, "g_3"] \arrow[ddrrr, hook, "h_3"] & & & & & & \beta_1^{View} \arrow[dlll, hook', bend left=10, "f'_1"'] \arrow[ddr, "g'_1"] \arrow[ddlll, hook', "h'_1"'] & \beta_2^{View} \arrow[dllll, hook', bend left=10, "f'_2"'] \arrow[dd, "g'_2"] \arrow[ddllll, hook', bend left=10, "h'_2"'] & \beta_3^{View} \arrow[dlllll, hook', bend left=10, "f'_3"'] \arrow[ddl, "g'_3"] \arrow[ddlllll, hook', bend left=10, "h'_3"'] \\
 & & & & & \mathfrak{R}^{View}_{[+]} \arrow[d, "\exists! k"] & & & & & \\
 & \sigma_1 & & \psi_1 \arrow[ll, "n"'] \arrow[rr, "p"', "|" marking] & & \mathfrak{R}^{View}_{U} & & \psi_2 \arrow[ll, "p'", "|" marking] \arrow[rr, "n'"] & & \sigma_2 &
\end{tikzcd}$$

Figure 11: Limit and co-limit construction. Here α_i^{View} $(i \in \{1, 2, 3, ..\})$ and β_j^{View} $(j \in \{1, 2, 3, ..\})$ represent conflicts in two different conflict-zones

veiwpoints. In contrast, our modeling approach provides a comprehensive framework for analyzing conflicts from various abstraction level which provides trends of event progression. It offers a unified solution that can address multiple aspects of conflict analysis.

We have presented a model transformation approach to prepare multiple viewpoints. However, by leveraging languages and frameworks such as ATL [12], MTBE [19, 20], one can automate the process of transforming models into visual formats, thus facilitating data visualization in different domains. These methodologies can be exploited for making data more accessible and comprehensible through visual means.

The model-based approach proposed in this paper enables the analysis of conflicts from multiple perspectives and levels of abstraction, enhancing transparency and traceability in the analysis process and thereby improving accountability in conflict analysis. We presented an ontology to capture key concepts related to conflicts. However, there are many other aspects that might be important to capture such as political and economic issues, etc. Future works include an adaptive method to update the domain ontology using generative AI.

ACKNOWLEDGEMENT

This research is funded by SFI MediaFutures partners and the Research Council of Norway (grant number 309339).

REFERENCES

[1] [n. d.]. Bishkek Protocol. https://en.wikipedia.org/wiki/Bishkek_Protocol. Accessed: 2024-03-12.

[2] 1995. Treaty on the Non-Proliferation of Nuclear Weapons (NPT). https://disarmament.unoda.org/wmd/nuclear/npt/. Accessed: 2024-03-12.

[3] 2023. GDELT. https://www.gdeltproject.org/data.html. Accessed: 2023-09-12.

[4] Martin Bal\'a\v{z}, Jozef Frt\'us, and Martin Homola. 2014. Conflict Resolution in Structured Argumentation. In *LPAR-19. 19th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (EPiC Series in Computing, Vol. 26)*, Ken Mcmillan, Aart Middeldorp, Geoff Sutcliffe, and Andrei Voronkov (Eds.). EasyChair, 23–34. https://doi.org/10.29007/brgz

[5] Michael Barr and Charles Wells. 1990. *Category Theory for Computing Science.* Prentice-Hall, Inc., USA.

[6] Arne Berven, Ole A. Christensen, Sindre Moldeklev, Andreas L. Opdahl, and Kjetil J. Villanger. 2020. A knowledge-graph platform for newsrooms. *Computers in Industry* 123 (2020), 103321. https://doi.org/10.1016/j.compind.2020.103321

[7] Cristiano Castelfranchi. 2015. *The Cognition of Conflict: Ontology, Dynamics, and Ideology.* Springer International Publishing, Cham, 3–32. https://doi.org/10.1007/978-3-319-14081-0_1

[8] Zinovy Diskin and Uwe Wolter. 2007. A Diagrammatic Logic for Object-Oriented Visual Modeling. In *Proceedings of the Second Workshop on Applied and Computational Category Theory, ACCAT@ETAPS 2007, Braga, Portugal, March 25, 2007 (Electronic Notes in Theoretical Computer Science, Vol. 203)*, Hartmut Ehrig, Jochen Pfalzgraf, and Ulrike Prange (Eds.). Elsevier, 19–41. https://doi.org/10.1016/J.ENTCS.2008.10.041

[9] Friedrich Glasl. 2011. *Konfliktmanagement.* VS Verlag für Sozialwissenschaften, Wiesbaden, 125–145. https://doi.org/10.1007/978-3-531-92789-3_4

[10] Edwin Ho, Arvind Rajagopalan, Alex Skvortsov, Sanjeev Arulampalam, and Mahendra Piraveenan. 2022. Game Theory in Defence Applications: A Review. *Sensors* 22, 3 (2022). https://doi.org/10.3390/s22031032

[11] Thomas Jordan. 2000. Glasl's Nine-Stage Model Of Conflict Escalation. (01 2000).

[12] Frédéric Jouault and Ivan Kurtev. 2006. Transforming Models with ATL. In *Satellite Events at the MoDELS 2005 Conference*, Jean-Michel Bruel (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 128–138.

[13] Gregor Leban, Blaz Fortuna, Janez Brank, and Marko Grobelnik. 2014. Event Registry: Learning about World Events from News. In *Proceedings of the 23rd International Conference on World Wide Web* (Seoul, Korea). ACM, 107–110. https://doi.org/10.1145/2567948.2577024

[14] Florian Mantz. 2014. Coupled Transformations of Graph Structures applied to Model Migration. https://doi.org/10.17192/z2014.0783

[15] Pradeep K. Murukannaiah, Anup K. Kalia, Pankaj R. Telangy, and Munindar P. Singh. 2015. Resolving goal conflicts via argumentation-based analysis of competing hypotheses. In *2015 IEEE 23rd International Requirements Engineering Conference (RE).* 156–165. https://doi.org/10.1109/RE.2015.7320418

[16] Roger B Myerson. 2013. *Game theory.* Harvard university press.

[17] Adrian Rutle. 2010. Diagram predicate framework: A formal approach to MDE.

[18] John A.A. Sillince. 1994. Multi-agent conflict resolution: a computational framework for an intelligent argumentation program. *Knowledge-Based Systems* 7, 2 (1994), 75–90. https://doi.org/10.1016/0950-7051(94)90021-3

[19] Michael Strommer and Manuel Wimmer. 2008. A Framework for Model Transformation By-Example: Concepts and Tool Support. In *Objects, Components, Models and Patterns*, Richard F. Paige and Bertrand Meyer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 372–391.

[20] Manuel Wimmer, Michael Strommer, Horst Kargl, and Gerhard Kramler. 2007. Towards Model Transformation Generation By-Example. In *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07).* 285b–285b. https://doi.org/10.1109/HICSS.2007.572