Conflict Ontology

Technical Documentation

Silvia Bieliková, Miroslav Cibula,
Patrik Filipiak, Juraj Gavura

May 27, 2024

Contents

1 Introduction and scope

The purpose of this ontology is to represent historical events, chiefly violent conflicts, their belligerents, and related concepts such as the geopolitical, social, and cultural situation surrounding conflicts and other entities. Furthermore, ontology attempts to capture chronological relationships between represented concepts and their hierarchical structure.

This project's motivation is to create a structured, machine-readable historical data resource focused on conflicts. By limiting the scope of this ontology, we aim to design a more compact data structure compared to general-purpose ontologies and databases such as DBpedia or Wikidata.

2 Existing resources

The first step after defining the scope was to analyze existing resources. Initially, we searched for published ontologies focusing on conflicts, wars, or battles. The first valid result found was an image (Figure 1) linked to an article by Koho, which is, unfortunately, currently not available anywhere (Koho et al., 2017). However, this missing article brought our attention to its author and we found out, that more recently, the same author published another paper, where a knowledge graph containing information about World War II with a focus on Finnish military history is presented (Koho et al., 2021). Event-based modeling has been used since wars and war-related data are time-based and happen sequentially (Figure 2). This model is fine-grained in its time and chronology representation but does not contain everything we searched for, mainly the political and social concepts related to the situation surrounding the conflicts.

We then moved to Wikipedia[1], Wikidata[2], and DBpedia[3]. These projects all proved to be very useful for our task. The most important part of the whole process was analyzing large amounts of data – initially, we focused on standard data (the most famous conflicts) and later switched to wars and battles that are more specific, since we did not want to neglect exceptions and outliers.

[1] https://en.wikipedia.org/

[2] https://www.wikidata.org/

[3] https://www.dbpedia.org/

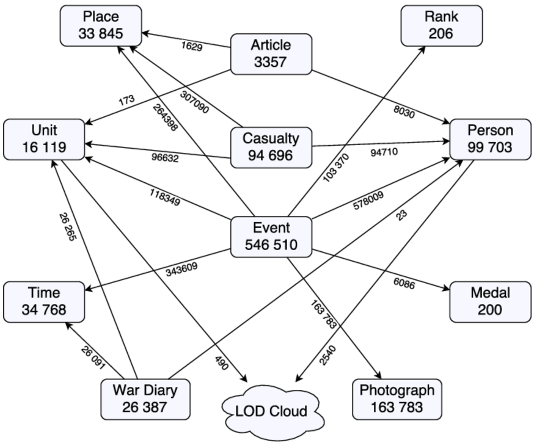


Figure 1

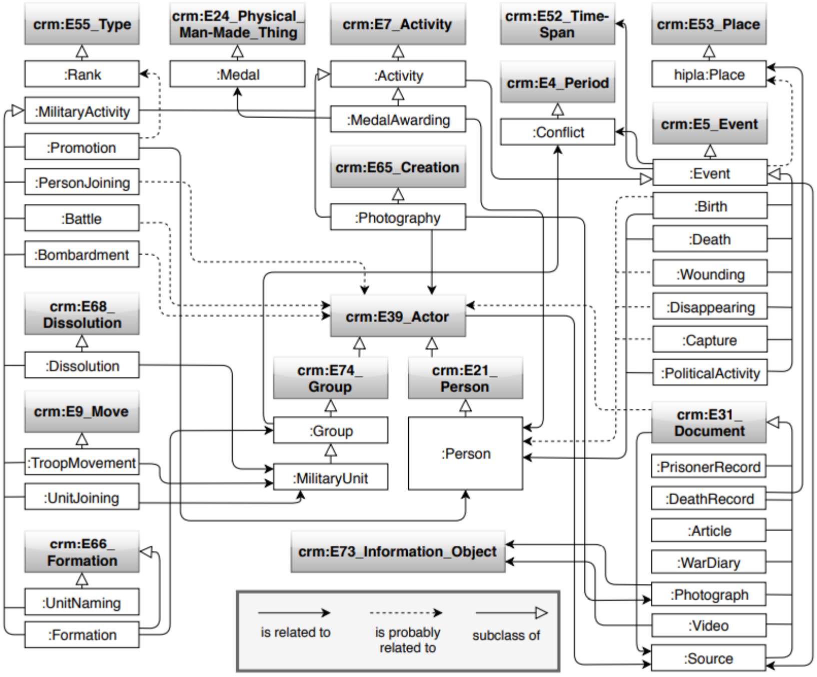
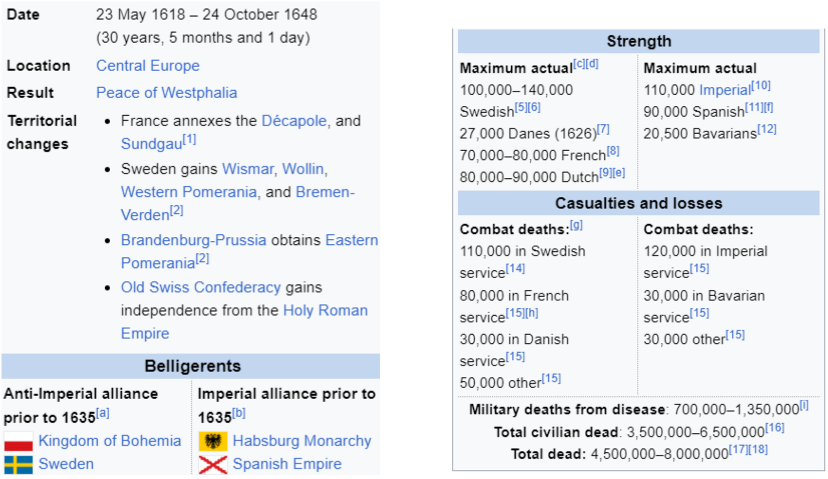


Figure 2



Date	23 May 1618 – 24 October 1648 (30 years, 5 months and 1 day)
Location	Central Europe
Result	Peace of Westphalia
Territorial changes	• France annexes the Décapole, and Sundgau[1] • Sweden gains Wismar, Wollin, Western Pomerania, and Bremen-Verden[2] • Brandenburg-Prussia obtains Eastern Pomerania[2] • Old Swiss Confederacy gains independence from the Holy Roman Empire

Belligerents

Anti-Imperial alliance prior to 1635[a]	Imperial alliance prior to 1635[b]
Kingdom of Bohemia Sweden	Habsburg Monarchy Spanish Empire

(a) World War II infobox

Strength

Maximum actual[c][d]	Maximum actual
100,000–140,000 Swedish[5][6] 27,000 Danes (1626)[7] 70,000–80,000 French[8] 80,000–90,000 Dutch[9][e]	110,000 Imperial[10] 90,000 Spanish[11][f] 20,500 Bavarians[12]

Casualties and losses

Combat deaths:[g]	Combat deaths:
110,000 in Swedish service[14] 80,000 in French service[15][h] 30,000 in Danish service[15] 50,000 other[15]	120,000 in Imperial service[15] 30,000 in Bavarian service[15] 30,000 other[15]

Military deaths from disease: 700,000–1,350,000[i]
Total civilian dead: 3,500,000–6,500,000[16]
Total dead: 4,500,000–8,000,000[17][18]

(b) Thirty Years' War infobox

Figure 3: Examples of Wikipedia infoboxes

We also found the Wikipedia page infoboxes highly practical for the initial stages of analyzing data (Figure 3).

Wikidata is backed up by an extensive general ontology, meaning no significant focus is present. It handles chronology by using properties `followedBy` and `follows`, the classes that represent time have a massive hierarchy. Classes representing different types of conflicts are also defined. Our main goal was to take inspiration from these projects – to specify, clean up, and enhance the part of the ontology provided by Wikidata that our project is concerned with, and to make it clearer, and easier to understand.

3 Outline and principal ideas

We developed ontology focusing on conflicts, chronology, and different types of changes. We will talk more about the structure in the next section, let us go through the principal ideas.

The core of the ontology is the class `Conflict` with its simple hierarchical structure comprised of two subclasses. It allows us to differentiate between wars and their battles. The class `Agent` and its subclasses represent individuals, nations, groups, or even countries. They are connected to `Conflict` via class `Belligerent` that serves as a wrapper. Chronology is present in several places in our ontology, mainly in the class `HistoricalPeriod`. The endings of conflicts are modeled using two classes: `Result` and `Aftermath`.

We have defined multiple enumeration classes used to tag objects or to connect them to predefined values that are not expected to change with new data.

Constraints are present primarily as domains, ranges and property characteristics.

4 Ontology structure

Most of our time was invested in the process of creating a suitable structure for our ontology. In this process, we were particularly mindful of generality, coherence, and readability. Thus, we had in mind a number of specific examples of different battles and wars so that this ontology could be used to represent any conflict. Therefore, the structure contains properties that not every conflict needs, and hence all properties are optional. At the same time, we were careful to make sure that the chosen structure, any connections between classes or subclasses, and the properties of those classes made sense for our purpose. Last but not least, it is important that the stored data is easy to navigate, i.e., that the whole structure is understandable. After many modifications, we have achieved a suitable structure that allows us to represent any armed conflict in certain detail. We will now look at the different interconnected parts of our structure.

The main class in this structure is the `Conflict` class (Figure 4). We can see that it contains two data properties, `startDate` and `endDate`, which assign a time period to the conflict. Furthermore, we can see the object properties `casusBelli`, `conflictType`, `location` and `conference`.

Using the `casusBelli` property, we can assign a cause to the conflict in the form of the enumeration class `Cause`. The possibilities are

- `economicCause`,
- `ideologicalCause`,
- `politicalCause`,
- `religiousCause`, and
- `territorialCause`.

Another enumeration class is the `ConflictType` class, which we associate with a conflict using the `conflictType` property. This class assigns a type to a conflict. The types to choose from are

- `aerial`,
- `biological`,
- `chemical`,
- `civil`,
- `cold`,
- `cyber`,
- `economic`,
- `hot`,
- `hybrid`,
- `informational`,
- `naval`,
- `proxy`,
- `psychological`, and
- `radiological`.

The `location` property refers to the `State` class that gives the conflict a location. A possible conference will use the `Conference` class and the `conference` property. This conference has its outcome in the `outcome` property and its venue in the `venueLocation` property. We also see here that the `Conflict` class is referenced by the `ConflictAftermath` class. This can be used if the result of a conflict is another conflict. The subclasses of the `Conflict` class are the `War` and `Battle` classes, which are linked to each other by the inverse properties `partOfWar` and `includesBattle`.

Other important classes that are linked to the main `Conflict` class are

- `Belligerent`,
- `Agent`,
- `HistoricalPeriod`,
- `Result`, and
- `Aftermath`.

The `Belligerent` class (Figure 5) represents any participant in the conflict. It is associated with the `Conflict` class by the `belligerent` property. The `strength` data property describes the military strength of a participant in a particular conflict. And the `casualties` data property gives the participant the number of lives lost, soldiers and civilians combined. A participant has its commanders in the `Person` class attached by the `leader` prop-

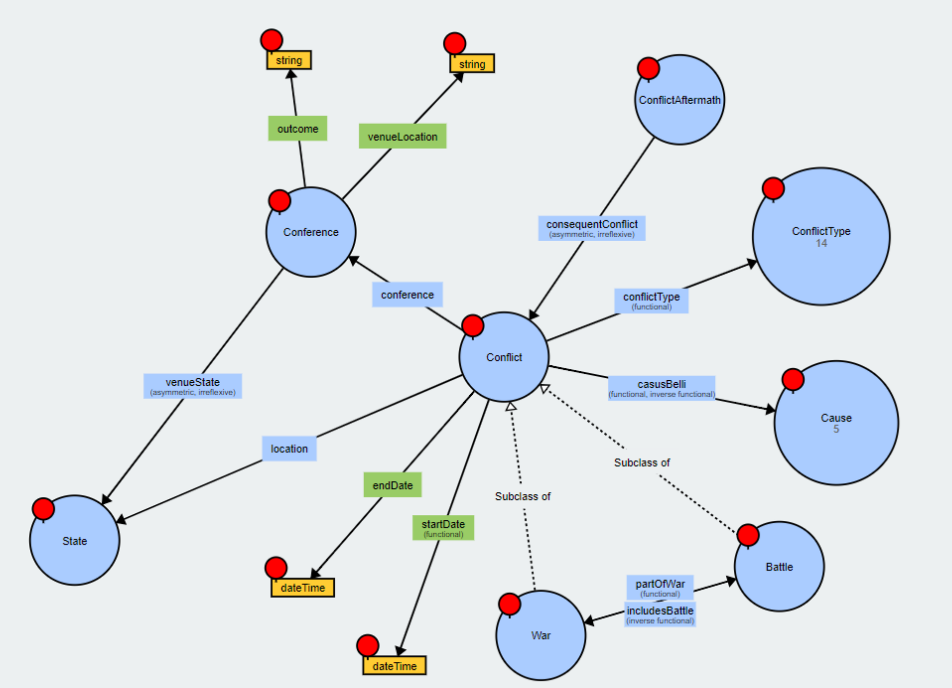


Figure 4: Conflict class

erty. The `Belligerent` class has a subclass `BelligerentGroup`, which is used to create allies. Individual allies are assigned to the group by the `includesBelligerent` object property. The `Person` class is a subclass of the `Agent` class, which is linked to the `Belligerent` class by the `isBelligerent` and `isAgent` properties.

The `Agent` class (Figure 6) is another important part of this structure. It is a superclass of the aforementioned `State` and `Person` classes, which are linked to other classes, as shown by the union. We can see that the `Person` class contains additional object properties for representing family members. These are the `spouse`, `sibling`, `parent` and `child` properties. The `State` class also has additional properties. The `includesState` property helps with countries that include other countries. The inverse properties `followedByState` and `precededByState` provide the chronological evolution of countries. The form of government of a state is described by the enumeration class `GovernmentForm` using the `governemntForm` property. Its options are

- `absoluteMonarchy`,

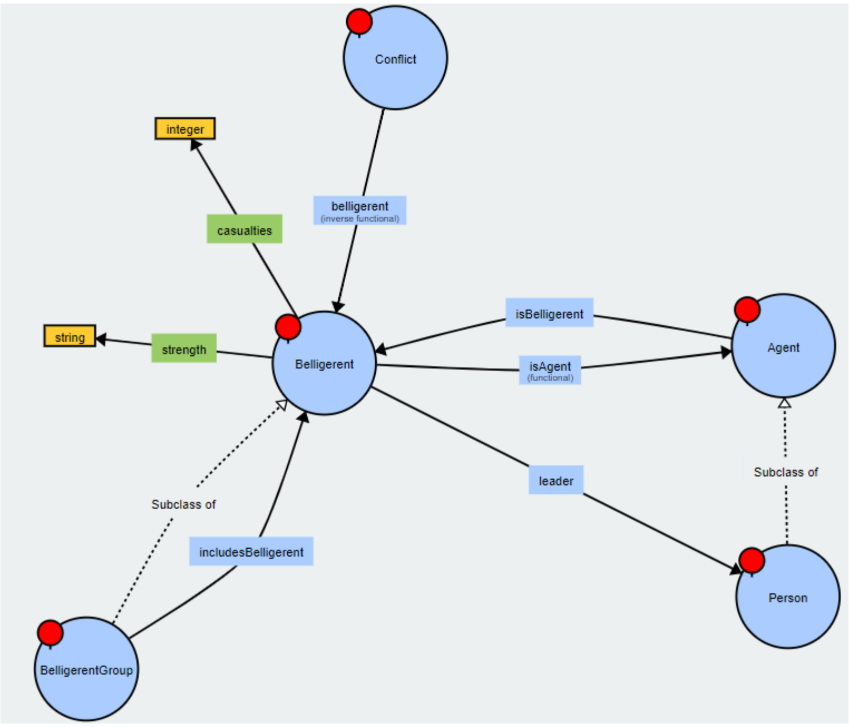


Figure 5: Belligerent class

- constitutionalMonarchy,
- dictatorship,
- parliamentaryDemocracy,
- participatoryDemocracy,
- presidentialDemocracy, and
- theocracy.

The Agent class itself contains a name via the name data property. The dates of its creation and termination, that is, the data properties startDate and endDate. And it also has a religion object property, which assigns a religion to the entity. Its third and final subclass is the AgentGroup class, which is used to form groups of people or states. It is linked to the Agent class by the partOf object property.

The HistoricalPeriod class (Figure 7) helps us to give a chronology to conflicts and their participants. It is also associated with the Agent class through the includesAgent and inHistoricalPeriod properties. Like

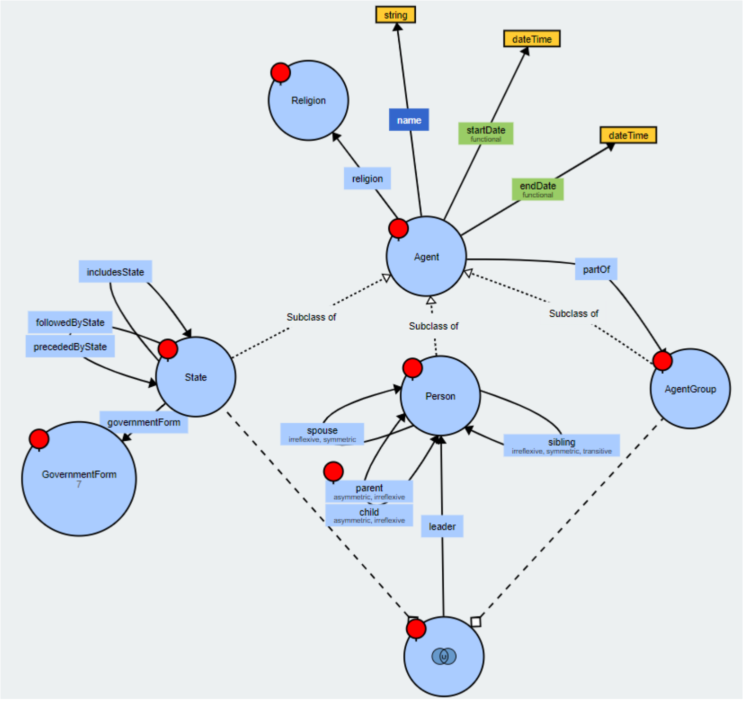


Figure 6: Agent class

other classes, the `HistoricalPeriod` class has data properties name, `startDate`, and `endDate`. Time sequence is provided by the inverse object properties `followedByHistoricalPeriod` and `precededByHistoricalPeriod`. If a historical period includes another period, the `includesHistoricalPeriod` and `partOfHistoricalPeriod` properties are used.

To represent the result of a conflict, we have the `Result` class (Figure 8). It is associated with the `Conflict` class through the `result` and `conflict` object properties. A conflict can end with a victory of one of the parties to the conflict or with a truce. Therefore, the `Result` class has two subclasses `Victory` and `Truce`. The `Victory` class further tells who won through the object property `belligerent`, and about what type this victory is, via the enumeration class `VictoryType`. The latter is linked to the `Victory` class via the `victoryType` property and has the options

- `strategic`, and
- `tactical`.

The last major class that is associated with conflict is the `Aftermath` class (Figure 9). It deals with all the possible aftermaths of a war or battle. It con-

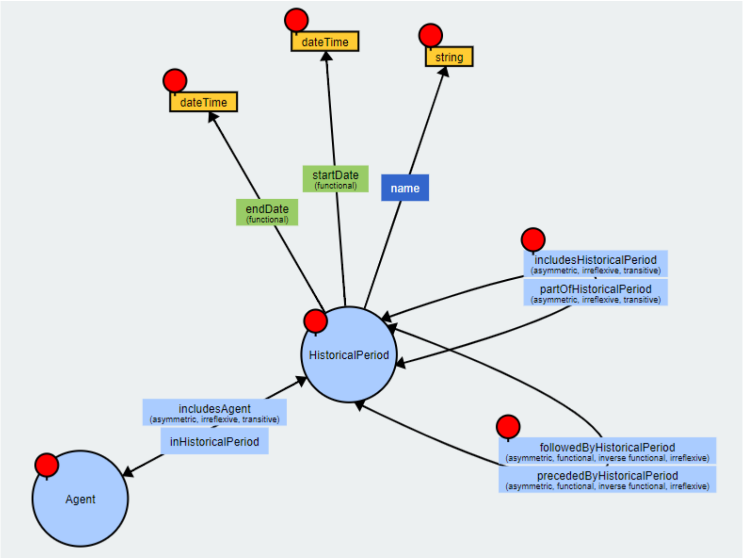

dateTime
dateTime
string
startDate
(functional)
name
endDate
(functional)
includesHistoricalPeriod
(asymmetric, irreflexive, transitive)
partOfHistoricalPeriod
(asymmetric, irreflexive, transitive)
HistoricalPeriod
includesAgent
(asymmetric, irreflexive, transitive)
inHistoricalPeriod
followedByHistoricalPeriod
(asymmetric, functional, inverse functional, irreflexive)
precededByHistoricalPeriod
(asymmetric, functional, inverse functional, irreflexive)
Agent

Figure 7: HistoricalPeriod class

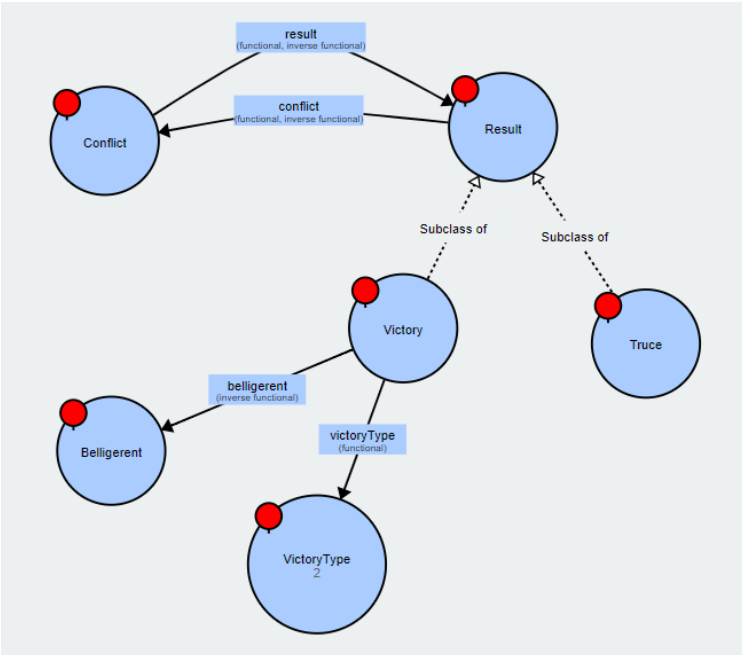


Figure 8: Result class

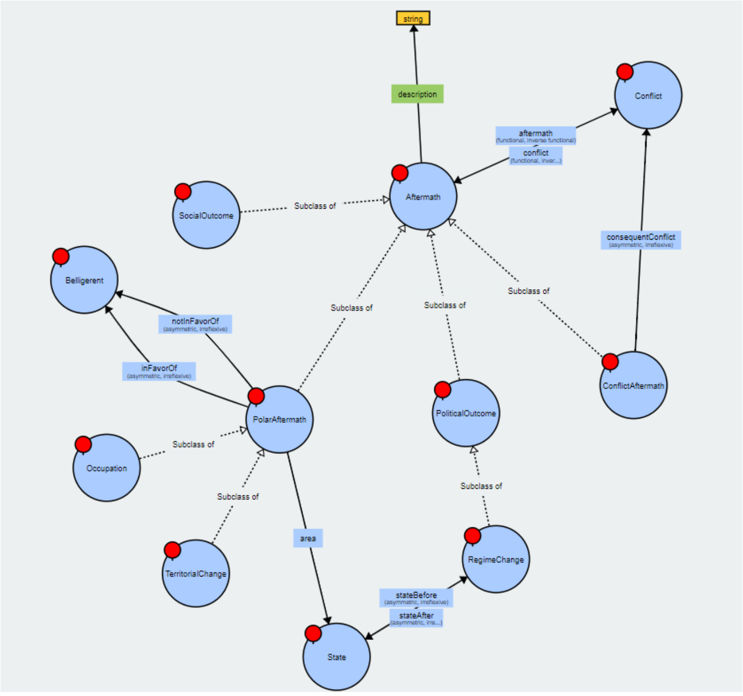


Figure 9: Aftermath class

nects to a particular conflict through the inverse object properties `aftermath` and `conflict`, or through the aforementioned subclass `ConflictAftermath` and the object property `consequentConflict`. The aftermath can be described in any way using the data property `description`. In addition to the `ConflictAftermath` subclass, it is also divided into the `PoliticalOutcome`, `PolarAftermath` and `SocialOutcome` subclasses. The `PoliticalOutcome` class has another subclass `RegimeChange`, which refers to two states via the object properties `stateBefore` and `stateAfter`.The `PolarAftermath` class also references the `State` class, but for the purpose of representing the affected region. The `area` property is used for this purpose. The `PolarAftermath` class further refers to the `Belligerent` class via the `notInFavorOf` and `inFavorOf` properties. And it is also subdivided into `Occupation` and `TerritorialChange` subclasses depending on the significance of the conflict consequence.

5 Instance examples

To make our ontology easier to comprehend, we chose a real war and put it into our ontology. The Second World War was the war we ended up using to showcase our system. The entire war is way too complex to cover in our example, so we picked just the parts needed for our purposes. First, we put the necessary data into a table, and afterward we created a few diagrams from said data.

In Figure 10, we can see that the WWII object is an instance of a class `War`. Furthermore, the arrows pointing from the WWII object are its properties. These properties connect WWII to different objects (for example, `War`'s property `includesBattle` connects WWII to the Battle of Stalingrad). The property `includesBattle` is an inverse property of `partOfWar`. `War` also has a property `conference` which references a `Conference` object that can be instantiated like the Figure 11.

Figure 12 describes the side of the Allies in WWII in a cursory manner. The Allies group is defined as a `BelligerentGroup` which consists of more different belligerents. All of them contain information like their strength in the war, casualties, or names.

Agent class and its subclasses (Figure 13) describe any entity that partakes in a conflict. For the diagram, we chose the USSR, which was a dictatorship and whose leader during the Second World War was Josif Stalin. The empty `religion` object indicates that these properties are optional and can be left blank if no fully accurate option is available.

The end of WWII and its consequences are captured by Figures 14 and 15. The result we chose was Allied Victory, which can be used to describe the outcome of the war and the Battle of Normandy. The `VictoryType` for this result is both strategic and tactical, requiring superior planning skills in addition to prowess on the battlefield. After any war or battle, its consequences begin to emerge, represented by the `Aftermath` class and its subclasses. One of those consequences was the stop of Germany's expansion after the Battle of Stalingrad.

Lastly, Figure 16 shows consecutive time periods in history. USA existed during Modern Era and Postmodern Era `HistoricalPeriod`'s.

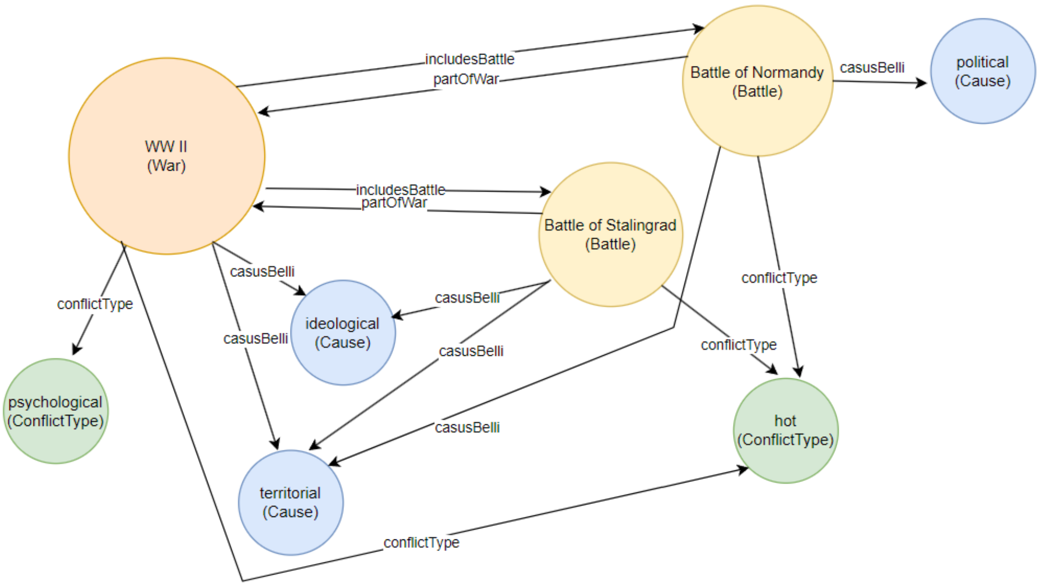
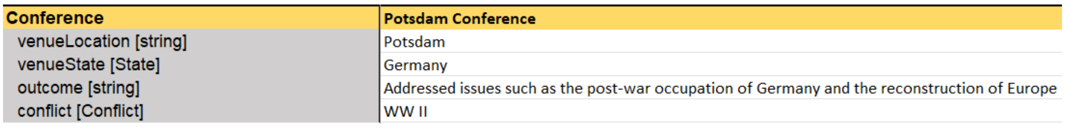


Figure 10: Diagram of classes `War`, `Battle`, `Cause` and `ConflictType` for a small part of WWII



Conference	Potsdam Conference
venueLocation [string]	Potsdam
venueState [State]	Germany
outcome [string]	Addressed issues such as the post-war occupation of Germany and the reconstruction of Europe
conflict [Conflict]	WW II

Figure 11: Example of an instance of the `Conference` class in the table form.

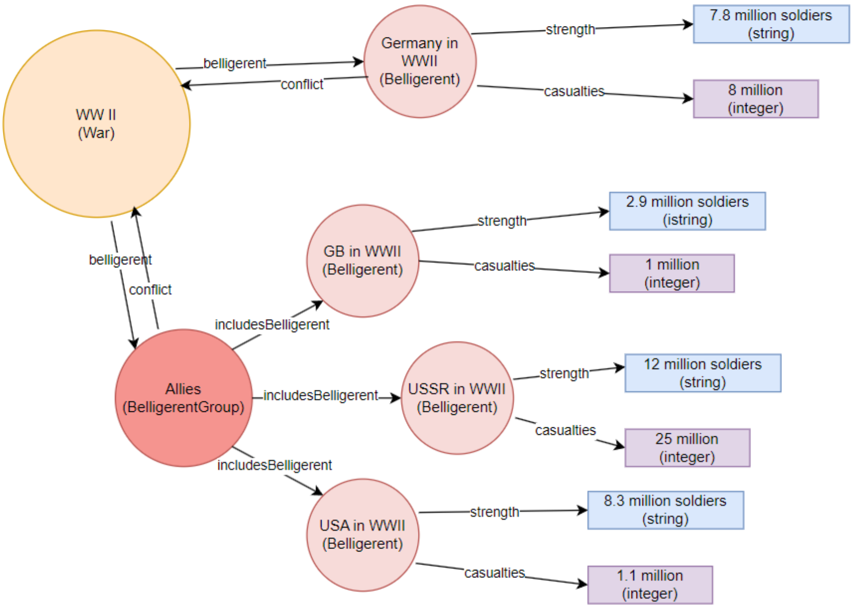


Figure 12: Diagram of classes `Belligerent` and `BelligerentGroup` for a small part of WWII

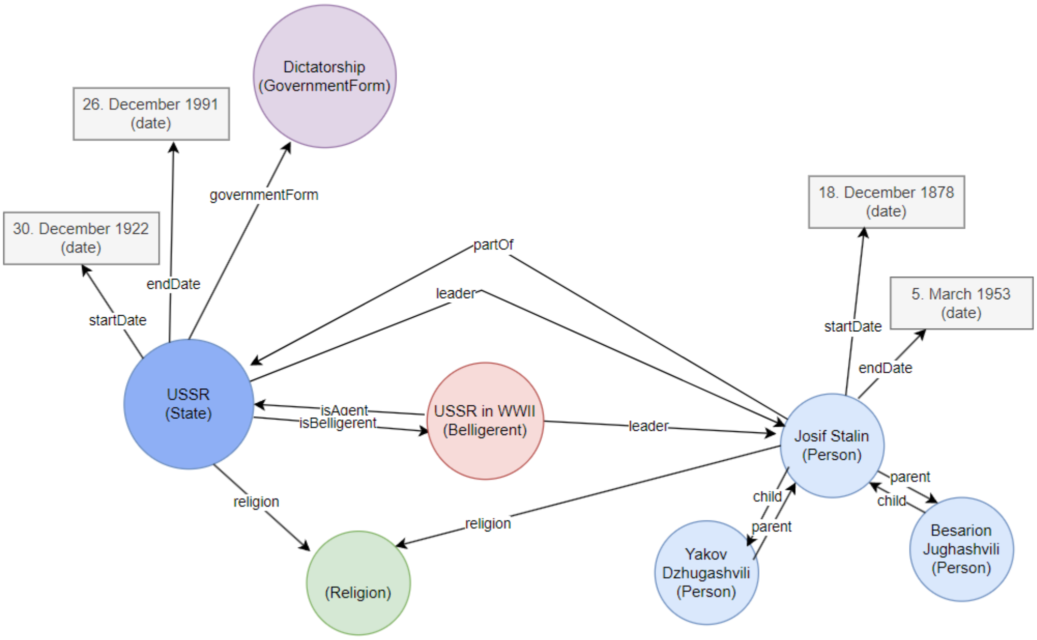


Figure 13: Diagram of classes `Person`, `State`, `GovernmentForm` and `Religion` for a small part of WWII

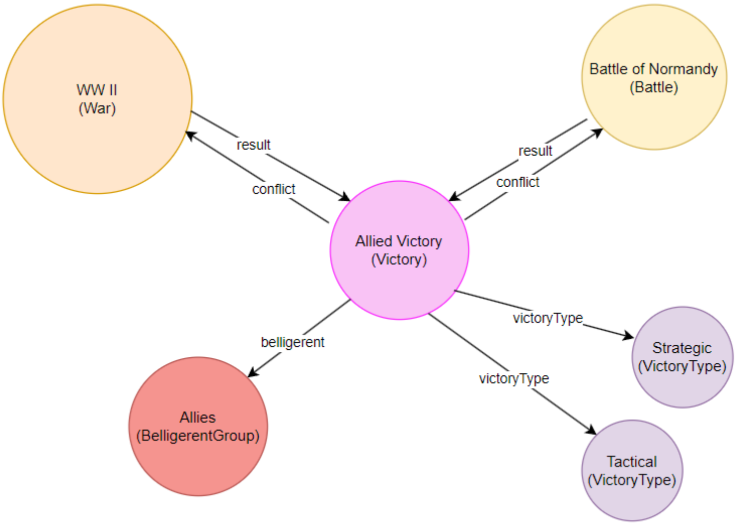


Figure 14: Diagram of classes `Victory` and `VictoryType` for the Battle of Normandy

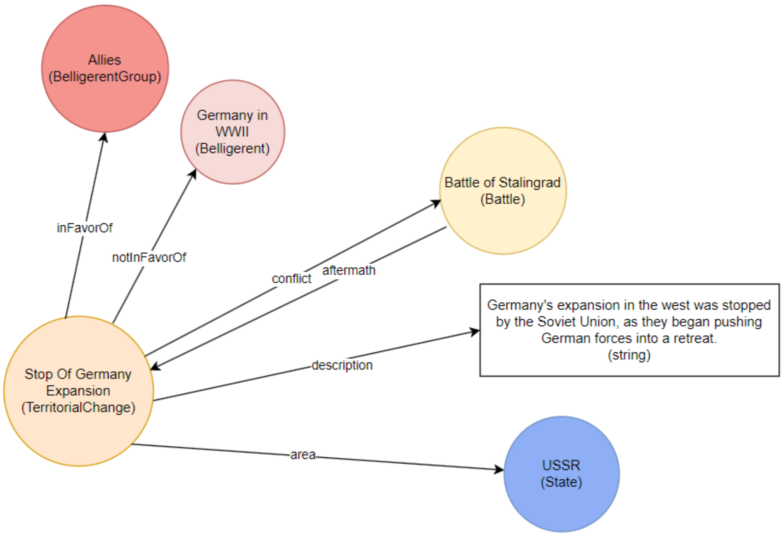


Figure 15: Diagram of class `TerritorialChange` as an aftermath of the Battle of Stalingrad

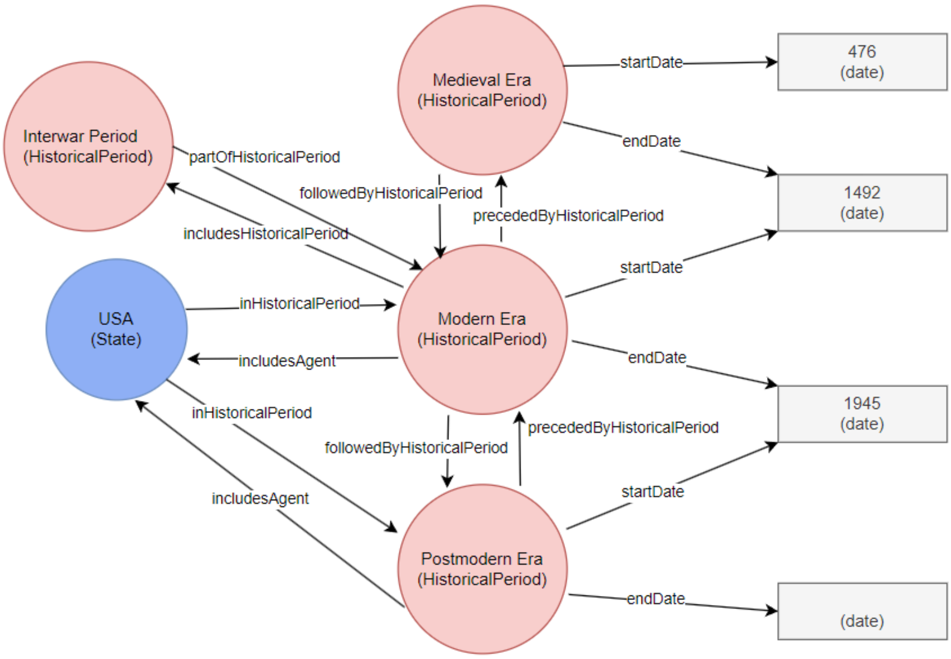


Figure 16: Diagram that shows different instances of `HistoricalPeriod` class and their relations

6 Conclusion and future work

In this project, we designed and implemented an ontology representing historical conflicts and related concepts. While general-purpose ontologies and graph databases such as Wikidata or DBpedia contain such data, they are not primarily focused on historical data or conflicts and thus miss specialized features or structures.

We designed the ontology as compactly as possible while supporting various conflict-related entities and their geopolitical context. Additionally, we wanted to organize concepts in the ontology in chronological and hierarchical structures.

While the ontology structure is complete, we encountered some challenges that require refactoring and optimization. For instance, during instance creation, we discovered that dense connections between certain classes can make it impractical in some cases, leading to the storage of large amounts of redundant data in different entities. This is particularly evident in the chronology system and its main `HistoricalPeriod` class, which necessitates multiple redundant relation definitions due to its bidirectional connections to multiple other classes.

Another problem to be solved in the future is that entities such as agents are created individually for each historical period they existed in. While this approach linearizes the chronological structure, it also introduces many duplicate non-interlinked information.

In conclusion, the task of formalizing and representing historical data has proven immensely complex, given the modelled domain's variability, complexity, and irregularity. While our ontology theoretically captures complex entities and a variety of relationships between them, its current form is not suitable for practical applications due to its high space complexity. To make it viable for practical applications, we would need to redesign the chronology system at the very least.

References

Koho, M., Heino, E., Leskinen, P., Tamper, M., Ikkala, E., Mäkelä, E., Tuominen, J., and Hyvönen, E. (2017). Linked open data and ontology infrastructure for second world war history.

Koho, M., Ikkala, E., Leskinen, P., Tamper, M., Tuominen, J., and Hyvönen, E. (2021). WarSampo knowledge graph: Finland in the second world war as linked open data. *Semantic Web*, 12(2):265–278.

Mcibula, silviabielikova, jgavura, and FPato (2024). Github repository conflict-ontology. https://github.com/Mcibula/conflict-ontology.