Ontological Foundations of Modelling Security Policies for Analysis[⋆]

Karolina Bataityte[1], Vassil Vassilev[2], and Olivia Jo Gill[1]

[1] School of Computing, London Metropolitan University, London, UK
[2] Cyber Security Research Centre, London Metropolitan University, London, UK
{k.bataityte,v.vassilev,o.gill}@londonmet.ac.uk

Abstract. Modelling of knowledge and actions in AI has advanced over the years but it is still a challenging topic due to the infamous frame problem, the inadequate formalization and the lack of automation. Some problems in cyber security such as logical vulnerability, risk assessment, policy validation etc. still require formal approach. In this paper we present the foundations of a new formal framework to address these challenges. Our approach is based on three-level formalisation: ontological, logical and analytical levels. Here we are presenting the first two levels which allow to model the security policies and provide a practical solution to the frame problem by efficient utilization of parameters as side effects. Key concepts are the situations, actions, events and rules. Our framework has potential use for analysis of a wide range of transactional systems within the financial, commercial and business domains and further work will include analytical level where we can perform vulnerability analysis of the model.

Keywords: Security Policies, Modelling, Ontologies, Knowledge Representation, Situations and Actions, Frame Problem

1 Introduction

In recent years there has been an increase in interest of analysing the logical vulnerability and the security policies of cyber systems. The security policies cover a wide range of situations: how to identify and authenticate the user, how to authorize the operations, how to maintain sessions and control the transactions, how to neutralize malicious activities and prevent unauthorized access to the information, etc. Any gaps or inconsistencies can open the door for logical vulnerabilities and leave the system exposed [3]. Our approach for analysis is to represent the domain knowledge in the ontological model and to formulate the security policies as a system of rules, so that we can analyse them *formally.* For this purpose, we developed a theory of Situations and Actions in Description

[⋆] This research is partially funded by Lloyds Banking Group in London, UK. However, no actual data from the bank has been used, the results and the opinions formulated in the paper are the author's and the examples are for illustration purpose only, without any resemblance to the actual banking policies and practices.

Logic (DL) and modelled the Security Policies in Clausal Logic (CL), which can be implemented using the standard languages of Semantic Web - Ontology Web Language (OWL) [4] and Semantic Web Rule Language (SWRL)[5]. We can model dynamic changes and synchronous actions with different security events asynchronously.

The paper is organized as follows. In Section 2 we will present the overall methodology which we follow. In Section 3 we will present logical foundations. In Section 4 we will introduce the ontological level. Section 5 will consider the security policies as rules on logical level. Section 6 we will conclude the paper and comment on the security policy analysis on Analytical Level.

2 Methodology

We separate the model of the world (*ontological level*) from the model of the policies which govern the changes in the world (*logical level*) and the model of the dynamic changes as a result of decisions (*analytical level*) (see Fig. 1). For each of the three levels we will use different formal systems, suitable for modelling of an aspect of the problem in a manner, similar to the infamous "layered cake" of the Semantic Web [7].

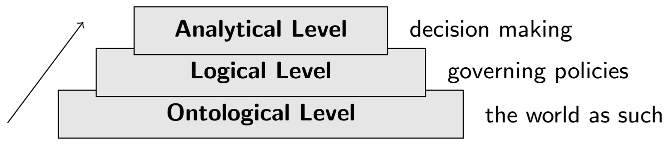

Fig. 1. Multi- level Model for Analysis

On Ontological Level we model the world using the vocabulary presented in Section 4.1. The conceptualization is similar to the famous *situation calculus (SitCalc)* [6], but formulating it using the language of DL makes it more "object-oriented" and allows for a new solution of the *frame problem* [8]. Using DL on this level allows us to implement the model entirely using OWL.

The Logical Level models the policies, captures constraints and completeness. It reflects the expert knowledge in the domain, which can be formulated as logical rules in CL and can be represented in computer format using SWRL.

The Analytical Level will deal with the analysis of the policies on a directed graph, considering the situations as nodes and the actions as edges however it is beyond the scope of this paper and is left for the next publication.

3 Logical Foundations

For developing of the theory of situations and actions we consider DL called $\mathcal{ALC}$ [11] which is not the most expressive but is expressive enough to support our needs without being too complicated beyond the necessity. More constructors can be added to extend $\mathcal{ALC}$ if the modelling requires it. As we choose DL for comfortable implementation in OWL, similarly we choose CL as we can implement rules in SWRL. The following two logics can be glued together for modelling the domain ontology and the policies within that domain.

3.1 Description Logic $\mathcal{ALC}$ as a Modelling Language

The syntax and the semantic interpretation is shown in Table 1. The interpretation I is a pair $I = (\triangle^I, \cdot^I)$, where $\triangle^I$ is a non-empty set (domain) and $\cdot^I$ is a mapping function [10].

Table 1. Syntax and Semantics

Concepts		**Roles**	
Syntax	Semantics	Syntax	Semantics
$\top$	$\triangle^I$	R	$R^I \subseteq \triangle^I \times \triangle^I$
$\bot$	$\emptyset$	$Domain(R,C)$	$< a,b >\in R^I \rightarrow a \in C^I$
A	$A^I \subseteq \triangle^I$	$Range(R,C)$	$< a,b >\in R^I \rightarrow b \in C^I$
$\neg C$	$\triangle^I \backslash C^I$		
$C \sqcap D$	$C^I \cap D^I$		
$C \sqcup D$	$C^I \cup D^I$		
$\forall R.C$	$\{a \in \triangle^I \vert \forall b.(< a,b >\in R^I \rightarrow b \in C^I)\}$		
$\exists R.C$	$\{a \in \triangle^I \vert \exists b.(< a,b >\in R^I \wedge b \in C^I)\}$		
where C, D are concepts, A is an atomic concept, R is a role.			

Given interpretation I in model M with axiom α, we say that M is a model of α under I if M satisfies α, written $I \models \alpha$. We will be expressing the domain restrictions as $\exists R.\top \sqsubseteq C$ and the range restrictions as $\top \sqsubseteq \forall R.C$ [11]. By adding domain and range axioms we are able to have a fixed structure of the real world we are modelling without the necessity to use more expressive language or non-standard semantics.

3.2 Clausal Logic and SWRL

In most logical languages it is possible to formulate rules, which are necessary for modelling the static constraints and dynamic changes. We have chosen a version of the first order clausal logic similar to the horn-clause logic because its serialized version SWRL refers directly to the terms of OWL.

SWRL Knowledge Base (K) is defined as follows: $K = (\Sigma, R)$ where Σ is KB of $\mathcal{ALC}$ and R is set of rules. The rules is compostged of *body* and *head* which is represented as following: *body* $\rightarrow$ *head*. It consists of a conjunctions of atoms which are classes $C(i)$ (concepts in $\mathcal{ALC}$) and object properties $R(i,j)$ (roles in $\mathcal{ALC}$) [5].

4 Ontological Level: The Domain Model

The term *ontology* in a narrow logical sense provides the *terminology*, which can be used for building the domain model, together with its *interpretation* in the *semantic domain* [9].

4.1 Ontology of the Domain

In our ontology the semantic domain, $\triangle$, is a non-empty set, split into three disjoint subdomains: **Entities, Events** and **Situations** (plural) as $\triangle_{Entities}$, $\triangle_{Events}$ and $\triangle_{Situations}$ respectively. In our theory we will use three terms with predefined meaning: *Entity*, *Event* and *Situation* (singular), which will be three separate taxonomies representing the static model of the world. The interpretation of $\mathcal{ALC}$ concepts in the domain are as follows: $Entity^I \subseteq \triangle^I_{Entities}$, $Event^I \subseteq \triangle^I_{Events}$ and $Situation^I \subseteq \triangle^I_{Situations}$. Our terminology (Table 2) will also include some predefined roles, one of them is *Action* ($Action^I \subseteq \triangle^I_{Situations} \times \triangle^I_{Situations}$), which can be used as a top of the hierarchy of actions. The ontology can have as many specific *named concepts* and *named roles* as needed, (noted as $\mathrm{Entity_x}$, $\mathrm{Situation_y}$, $\mathrm{Event_e}$, $\mathrm{Action_z}$), with the intended meaning and interpretations in the semantic subdomains introduced above in accordance with the syntax and semantics of $\mathcal{ALC}$ as presented in Section 3.1. Concepts from three subdomains must be disjoint as follows:

$$Situation \sqcap Event \sqsubseteq \bot, Situation \sqcap Entity \sqsubseteq \bot, Entity \sqcap Event \sqsubseteq \bot. \qquad (1)$$

Table 2. Vocabulary of the Domain Ontology

Term	DL Category	Use in modelling	Condition
Situation	concept	partial static description of the world	axiom 1
Event	concept	asynchronous activity	axiom 1
Entity	concept	qualitative descriptor	axiom 1
Action	role	synchronous activity	axiom 2
occur–in	role	event occurrence	axiom 4
present–at	role	situation description	axiom 6
part–of	role	event description	axiom 5
describe	role	describing entities and specifying dependencies	axiom 7
chain	role	connecting events causally	axiom 3

At the moment we are using only $\mathcal{ALC}$ TBox for terminological axioms and RBox for relational axioms.

4.2 Static Model of the World

Here we are defining a fix structure of the modelling world using terms above. A *Situation* is a concept, which represents a partial description of the world in a specific moment of time. Two *Situation* concepts can be connected via *Action* roles to model the potential change:

$$\exists Action.\top \sqsubseteq Situation, \top \sqsubseteq \forall Action.Situation \qquad (2)$$

The events are asynchronous activities which are modelled using *Event* concepts, linked through the predefined role *chain* in a causal chain (axiom 3). The intended meaning of *Event* is to represent a real-world events which can occur in the situations through the predefined role *occur–in* with domain *Event* and range *Situation* (axiom 4). This way we can formulate security policies with regard to unexpected or expected activities (events) which may or may not happen in the situations.

$$\exists chain.\top \sqsubseteq Event, \top \sqsubseteq \forall chain.Event \tag{3}$$

$$\exists occur\text{–}in.\top \sqsubseteq Event, \top \sqsubseteq \forall occur\text{–}in.Situation \tag{4}$$

The *Entity* concepts are used to describe both situations and events using the predefined roles: *part–of* with domain *Entity* and range *Event* (axiom 5); *present–at* with domain *Entity* and range *Situation* (axiom 6); *describe* with domain *Entity* and range *Entity* (axiom 7).

$$\exists part\text{–}of.\top \sqsubseteq Entity, \top \sqsubseteq \forall part\text{–}of.Event \tag{5}$$

$$\exists present\text{–}at.\top \sqsubseteq Entity, \top \sqsubseteq \forall present\text{–}at.Situation \tag{6}$$

$$\exists describe.\top \sqsubseteq Entity, \top \sqsubseteq \forall describe.Entity \tag{7}$$

It is important to note that the events do not change the situations in our theory, they can only occur in them; the changes can be caused only by actions.

4.3 World Dynamics

In state-based dynamic theories which uses DL, the actions are represented as ⟨*pre-condition, occlusion, post-condition*⟩ triplets [1], [2]. Unfortunately, there is no easy implementation of such a formalism since it has additional syntactic structure.

We have adopted the view that the dynamic changes are possible only through actions, similar to the original SitCalc from the early days of AI [8]. This logic formalism encounters the infamous frame problem, caused by the propositional treatment of the situations which require them to incorporate their parameters as arguments.

However, in our approach the definition of the actions (as relations between the situations) looks almost identical to SitCalc approach. The partitioning of our ontology has interesting and unexpected characteristics with practical importance for applications. We define the parameters of the actions contextually. In our approach the actions can change the situations only through their parameters, which are entities, but the action parameters are no longer attributed to the actions – they are attributed to the situations which the actions relate instead. This completely eliminates the need for heavy "frame axioms" because the complete absence of any "side effect" of the actions.

If we have TBox T with situations and entities as follows:

$$T := \{\mathrm{Entity}_{\mathrm{x}} \sqsubseteq Entity, \mathrm{Situation}_{\mathrm{y}} \sqsubseteq Situation\} \tag{8}$$

and $\mathrm{Entity}_{\mathrm{x}}$ describe $\mathrm{Situation}_{\mathrm{y}}$, T is extended as follows:

$$T' := T \cup \{\mathrm{Entity}_{\mathrm{x}} \sqsubseteq \exists present\text{–}at.\mathrm{Situation}_{\mathrm{y}}\}. \tag{9}$$

Example 1. Let's consider the situation *LoggedIn* and the entity *User*. For this scenario the TBox T is as follows:

$$T := \{User \sqsubseteq Entity, LoggedIn \sqsubseteq Situation, User \sqsubseteq \exists present\text{–}at.LoggedIn\}$$

Each situation can be described by a number of entities. Since the actions change the situations, they will affect these entities but not directly. So, we can consider the entities which describe all situations in which a given action applies as its *input parameters* and similarly, entities which describe the situations to which the action leads as its *output parameters*. NB: not all entities are input and/or output parameters, some of them just describe the situation without being needed for an action. To specify the parameters of all actions, we can create a GBox G as follows:

Definition 1. *A GBox* $G = \{\langle \mathrm{Entity_x}, \mathrm{Action_z}\rangle, \langle \mathrm{Action_z}, \mathrm{Entity_y}\rangle\}$ *is a set of pairs of actions and entities, representing the action parameters where pair* $\langle \mathrm{Entity_x}, \mathrm{Action_z}\rangle$ *is for input parameters and pair* $\langle \mathrm{Action_z}, \mathrm{Entity_y}\rangle$ *is for output parameters.*

The action parameters will be important on the Analytical Level since the input parameters are binding the actions, making them executable, while the output parameters are producing the effect, determining the changes in the situations.

In order for an entity to be an input parameter, it must meet the following conditions:

1. $\exists \mathrm{Action_z}.\top \sqsubseteq \mathrm{Situation_x}$,
2. $\mathrm{Entity_e} \sqsubseteq \exists \mathit{present–at}.\mathrm{Situation_x}$.

If both conditions hold, we can say GBox $G = \{\langle \mathrm{Entity_e}, \mathrm{Action_z}\rangle\}$. It can be formalized as the following axiom:

$$\mathrm{Entity_e} \sqsubseteq \exists \mathit{present–at}.(\mathrm{Situation_x} \sqcap \exists \mathrm{Action_z}.\top) \tag{10}$$

which says that $\mathrm{Entity_e}$ is connected to a $\mathrm{Situation_x}$ via *present–at* and there is an $\mathrm{Action_z}$ starting at $\mathrm{Situation_x}$ and leading to another unknown *Situation*. This gives us the first criteria for analysing the descriptive completeness of the security policies with respect to the possibility of binding the input parameters of the applicable actions to the descriptions of the situations in which they apply.

In order for an entity to be an output parameter, it must meet the following conditions:

1. $\top \sqsubseteq \forall \mathrm{Action_z}.\mathrm{Situation_y}$,
2. $\mathrm{Entity_e} \sqsubseteq \exists \mathit{present–at}.\mathrm{Situation_y}$.

If both conditions hold, we can say GBox $G = \{\langle \mathrm{Action_z}, \mathrm{Entity_e}\rangle\}$. It can be formalized as follows:

$$\mathrm{Entity_e} \sqsubseteq \exists \mathit{present–at}.\exists \mathrm{Action_z}.\mathrm{Situation_y} \tag{11}$$

which says that $\mathrm{Entity_e}$ describes $\mathrm{Situation_y}$ via *present–at* and $\mathrm{Action_z}$ leads to $\mathrm{Situation_y}$ after it executes.

In our dynamical model the situations change as a result of an action. The only way this can affect the description of new situations is through the output parameters of the actions causing the transition. The specific changes caused by the actions must be specified by the corresponding rules of the security policy.

Principle of preservaltion: Any description of the situations within the domain of the action in terms fo input parameters remains unchanged.

Principle of propagations: Any description of the situations within the range of the action in terms fo output parameters defines the change.

Example 2. In Fig. 2 we have a scenario which starts in situation Situation_1 and finishes in Situation_3 after executing Action_1 and Action_2 . The two actions have parameters amongst the entities which are present in the corresponding situations. In this case $G = \{\langle \text{Entity}_2, \text{Action}_1\rangle, \langle \text{Action}_1, \text{Entity}_3\rangle, \langle \text{Action}_1, \text{Entity}_4\rangle, \langle \text{Entity}_4, \text{Action}_2\rangle\}$. Amongst the parameters Entity_4 is both input and output parameter of Action_2. Entity_1 and Entity_5 simply describe the situations without being needed for actions.

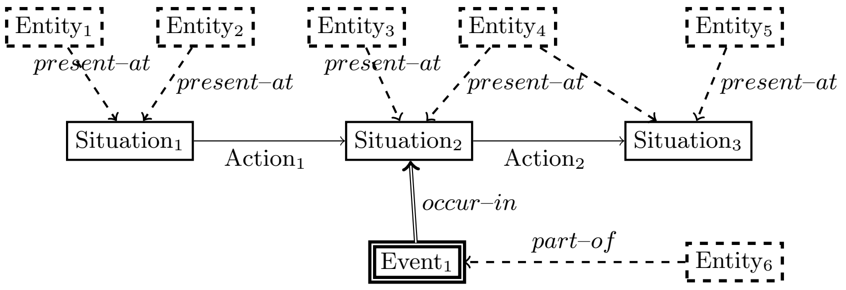

Fig. 2. A graphical representation of two-step journey

The ontological considerations we have presented so far can be constructed in any variation of DL. Since such a theory can be serialized directly in OWL, the process of developing the ontology can be done entirely interactively using any standard ontology editor, such as **Protégé**.

5 Logical Level: Constraints, Dependencies and Domain Policies

In order to describe the logical characteristics of the model, as well as to represent adequately the domain policies controlling the execution of the actions, we can use axioms, rules of inference and heuristic rules. Although DL and CL, as theoretical base of our framework, have well-defined inference mechanisms for practical purposes, it is more convenient to work with derived inference rules rather than the rules of inference within the underlying logic. In this section we will discuss some derived rules of our framework which allow us to automate this process.

5.1 Parameter Binding and Entity Completion

To make sure that our KB is descriptively complete, we need to guarantee that it contains all needed information in the TBox (the ontology model) to match the SWRL rules (the policies) so that the policy rules which prescribe actions actually lead to executable actions. In practice this means that all parameters of the actions in the head of the rules must be bound to the situations in which the rules apply. This can be implemented using an algorithm which uses the ontology in the TBox to check if the parameters of the actions prescribed by the rules are defined.

The following derived rule captures the parameters of various events in the situations to prevent the loss of bindings. It is used to implement a “reasoner” which performs a secondary logical inference according to the following schema:

$$\frac{\begin{array}{c} \mathit{Entity} \sqsubseteq \exists \mathit{part\text{–}of}.\mathit{Event} \\ \mathit{Event} \sqsubseteq \exists \mathit{occur\text{–}in}.\mathit{Situation} \end{array}}{\therefore\ \mathit{Entity} \sqsubseteq \exists \mathit{present\text{–}at}.\mathit{Situation}}$$

Derived Inference Rule 1 (Entity Triangulation). Let the following TBox T be given:

$$T := \{\mathit{Entity} \sqsubseteq \exists \mathit{part\text{–}of}.\mathit{Event}, \tag{12a}$$

$$\mathit{Event} \sqsubseteq \exists \mathit{occur\text{–}in}.\mathit{Situation}\} \tag{12b}$$

Then the following holds:

$$T' := T \cup \{\mathit{Entity} \sqsubseteq \exists \mathit{present\text{–}at}.\mathit{Situation}\}. \tag{13}$$

Proof. The TBox T holds since it states the domain and range of *part–of* (12a) and *occur–in* (12b) roles which satisfy the axioms 5 and 4 respectively. The same concept *Event* is used as range of *part–of* (12a) and as a domain of *occur–in* (12b). Therefore, we can substitute *Event* in 12a by the right-hand side of 12b to derive $\mathit{Entity} \sqsubseteq \exists \mathit{part\text{–}of}.\exists \mathit{occur\text{–}in}.\mathit{Situation}$. As we can see, *Entity* is connected to *Situation* via two roles. We know from Section 4.2, this can be done via *present–at* (axiom 6), therefore, it can be expressed as $\mathit{Entity} \sqsubseteq \exists \mathit{present\text{–}at}.\mathit{Situation}$ (13).

□

5.2 Transitivity of the Roles and Entity Propagation

The next derived rule reflects the abstract “transitivity” of the logical descriptions within one and the same situation. It can be accounted by another “reasoner” which performs secondary inference according to the following schemas against concept *Situation* or *Event*:

$$\frac{\begin{array}{c} \mathrm{Entity_y} \sqsubseteq \exists \mathit{describe}.\mathrm{Entity_x} \\ \mathrm{Entity_x} \sqsubseteq \exists \mathit{present\text{–}at}.\mathrm{Situation_x} \end{array}}{\therefore\ \mathrm{Entity_y} \sqsubseteq \exists \mathit{present\text{–}at}.\mathrm{Situation_x}} \quad \frac{\begin{array}{c} \mathrm{Entity_y} \sqsubseteq \exists \mathit{describe}.\mathrm{Entity_x} \\ \mathrm{Entity_x} \sqsubseteq \exists \mathit{present\text{–}at}.\mathrm{Event_z} \end{array}}{\therefore\ \mathrm{Entity_y} \sqsubseteq \exists \mathit{present\text{–}at}.\mathrm{Event_z}}$$

Derived Inference Rule 2 (Entity Transitivity). Let the following TBox T be given:

$$T := \{\mathrm{Entity_y} \sqsubseteq \exists \mathit{describe}.\mathrm{Entity_x}, \tag{14a}$$

$$\mathrm{Entity_x} \sqsubseteq \exists \mathit{present\text{–}at}.\mathrm{Situation_x}\} \tag{14b}$$

Then the following holds:

$$T' := T \cup \{\mathrm{Entity_y} \sqsubseteq \exists \mathit{present\text{–}at}.\mathrm{Situation_x}\}. \tag{15}$$

Proof. The TBox T holds since it states the domain and range of *describe* (14a) and *present–at* (14b) roles which satisfy the axioms 7 and 6 respectively. The same concept $\mathrm{Entity_x}$ is used as range of *describe* (14a) and domain of *present–at* (14b). Therefore, we can substitute $\mathrm{Entity_x}$ in 14a by the right-hand side of 14b to derive $\mathrm{Entity_y} \sqsubseteq \exists \mathit{describe}.\exists \mathit{present\text{–}at}.\mathit{Situation_x}$. As we can see, $\mathrm{Entity_y}$ is connected to $\mathrm{Situation_x}$ via two roles. Therefore $\mathrm{Entity_y}$ is connected to $\mathrm{Situation_x}$ and we can simply rewrite it as $\mathrm{Entity_y} \sqsubseteq \exists \mathit{present\text{–}at}.\mathrm{Situation_x}$ (15).

□

5.3 Conceptual Taxonomies and Entity Inheritance

Although the DL allows to automate the subsumption of concepts, we can extend our framework with additional inheritance mechanisms to allow full "parameter inheritance" in the style of object-oriented programming. This is possible because the entities, which are connected to situations or to events, are like the class attributes in object-oriented parlance. It is relatively straightforward to construct algorithmic reasoners which tackle more complex inheritance of entities, along the taxonomic hierarchies of situations and events.

$$\frac{\mathrm{Situation_y} \sqsubseteq \exists \mathrm{Situation_x} \qquad \mathrm{Entity_x} \sqsubseteq \exists \mathit{present\text{–}at}.\mathrm{Situation_x}}{\therefore\ \mathrm{Entity_x} \sqsubseteq \exists \mathit{present\text{–}at}.\mathrm{Situation_y}} \qquad \frac{\mathrm{Event_y} \sqsubseteq \exists \mathrm{Event_x} \qquad \mathrm{Entity_x} \sqsubseteq \exists \mathit{present\text{–}at}.\mathrm{Event_x}}{\therefore\ \mathrm{Entity_x} \sqsubseteq \exists \mathit{present\text{–}at}.\mathrm{Situation_y}}$$

Derived Inference Rule 3 (Entity Inheritance). Let the following TBox T be given:

$$T := \{\mathrm{Situation_y} \sqsubseteq \mathrm{Situation_x}, \tag{16a}$$

$$\mathrm{Entity_x} \sqsubseteq \exists \mathit{present\text{–}at}.\mathrm{Situation_x}\} \tag{16b}$$

Then the following holds:

$$T' := T \cup \{\mathrm{Entity_x} \sqsubseteq \exists \mathit{present\text{–}at}.\mathrm{Situation_y}\}. \tag{17}$$

Proof. The TBox T holds since it states that $\mathrm{Situation_y}$ is a sub-concept of $\mathrm{Situation_x}$ (16a) and $\mathrm{Entity_x}$ is related to $\mathrm{Situation_x}$ via *present–at* (16b). Therefore, $\mathrm{Entity_x}$ is also related to sub-concept of $\mathrm{Situation_x}$, which is $\mathrm{Situation_y}$ (17).

□

5.4 Policy Rules

The policies on the Logical Level are rules which link the concepts and roles from the Ontological Level. Such rules have clausal form and can be represented as SWRL expressions (Section 3.2). This makes possible the use of the ontological editors like **Protégé** for modelling of the policies on the Logical Level.

The policy rules can be modelled in SWRL using different templates which combine *Situation*, *Event*, *Entity* and *Action* atoms in the body and the head of the rule to serve different purposes - for analysis of the situations, making decisions for continuation of the journey or responding to events. Two such templates are shown below, which can be finely tuned to the particular need.

1. $\langle \mathit{situation}\rangle(?sa) \wedge \langle \mathit{entity}\rangle(?ia) \wedge \mathit{present\text{–}at}(?ia, ?sa) \wedge ... \wedge \langle \mathit{situation}\rangle(?sb) \wedge \langle \mathit{action}\rangle(?sa, ?sb) \rightarrow \langle \mathit{entity}\rangle(?ib) \wedge \mathit{present\text{–}at}(?ib, ?sb) \wedge ...$
2. $\langle \mathit{situation}\rangle(?sa) \wedge \langle \mathit{entity}\rangle(?ia) \wedge \mathit{present\text{–}at}(?ia, ?sa) \wedge \langle \mathit{event}\rangle(?ea) \wedge \mathit{occur\text{–}in}(?ea, ?sa) \wedge ... \wedge \langle \mathit{entity}\rangle(?ib) \wedge \mathit{part\text{–}of}(?ib, ?ea) \wedge ... \wedge \langle \mathit{situation}\rangle(?sb) \wedge \langle \mathit{action}\rangle(?sa, ?sb) \rightarrow \langle \mathit{entity}\rangle(?ic) \wedge \mathit{present\text{–}at}(?ic, ?sb)...$

In the templates above $\langle \mathit{situation}\rangle(?s)$, $\langle \mathit{entity}\rangle(?en)$, $\langle \mathit{event}\rangle(?ev)$ are SWRL classes (which correspond to $\mathcal{ALC}$ concepts), $\langle \mathit{action}\rangle(?sa, ?sb)$ are SWRL object properties (which correspond to $\mathcal{ALC}$ roles) and they have to be adopted to the specific scenario. Other classes/concepts and object properties/roles do not have to be adopted to the scenario and can be used as it is ($\mathit{present\text{–}at}(?ib, ?sb)$, $\mathit{occur\text{–}in}(?ea, ?sa)$, etc.)

5.5 Detailed Example

The following fragment was built in Protégé 5.1.0 with FaCT++ 1.6.5 reasoner. WebVOWL 1.1.7 was used for visualization of Fig. 3 and Fig. 4 was created using a drawing tool since softaware to generate these graphs is still in the development stage. Some specifications such as TBox, RBox and some of the named concepts are omitted due to space limitation. The purpose of this example is to illustrate our framework as well show the interpretation and understanding of it.

Lets consider the case when transaction is requested (*S_TransactionRequested*) and there are three possible events which may or may not happen (*E_AccountInOverdraft*, *E_MaxOverdraftReached*, *E_AccountOverloaded*). The end situation as well as entities depend on policy rules expressed in SWRL.

Fig. 3 shows the interpretation of ontological vocabulary from Ontological Level where yellow arrows represents some of the derived inference rules from Logical Level. Fig. 4 visualise the rules on Logical Level and will be used for Analytical Level for analysis which is left for another publication. The rules are as follows:

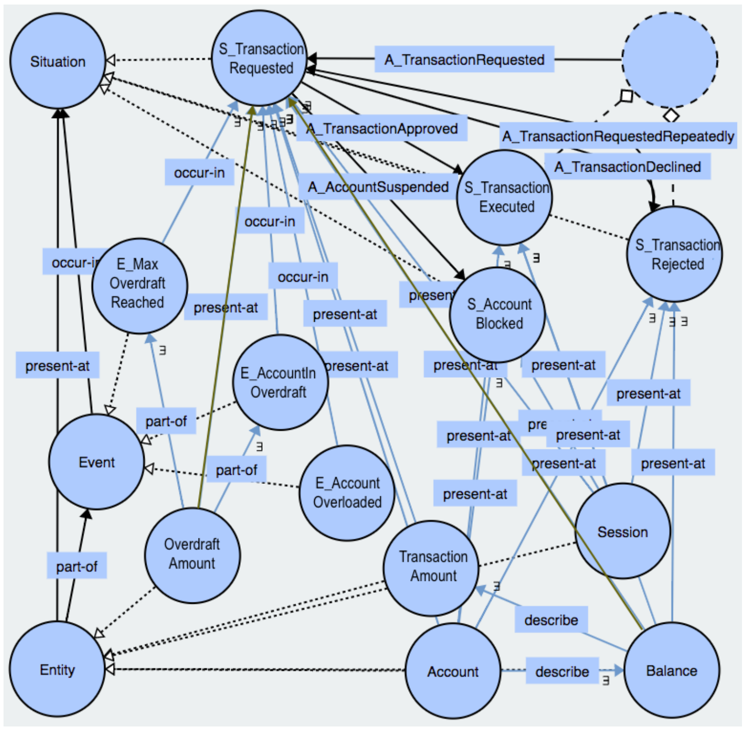

Fig. 3. Example Visualization of Ontological and Logical Levels

– $S\_TransactionRequested(?sa) \land Balance(?ix) \land present\text{–}at(?ix, ?sa) \land TransactionAmount(?it) \land present\text{–}at(?it, ?sa) \land E\_AccountInOverdraft(?ea) \land occur-in(?ea, ?sa) \land OverdraftAmount(?io) \land part\text{–}of(?io, ?ea) \land present\text{–}at(?io, ?sa) \land S\_TransactionExecuted(?sb) \land A\_TransactionApproved(?sa, ?sb) \rightarrow Balance(?iy)$

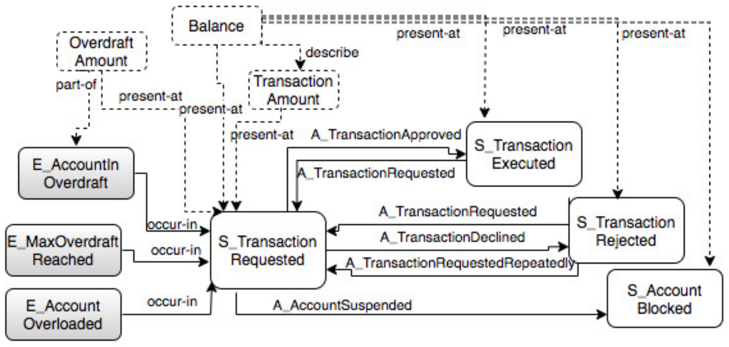

Fig. 4. Example Visualization of Logical and Analytical Levels

- $S\_TransactionRequested(?sa) \wedge Balance(?ib) \wedge present\text{–}at(?ib, ?sa) \wedge TransactionAmount(?it) \wedge present\text{–}at(?it, ?sa) \wedge E\_MaxOverdraftReached(?eb) \wedge occur - in(?eb, ?sa) \wedge S\_TransactionRejected(?sb) \wedge A\_TransactionDeclined(?sa, ?sb) \rightarrow Balance(?ib)$

- $S\_TransactionRequested(?sa) \wedge Balance(?ib) \wedge present\text{–}at(?ib, ?sa) \wedge TransactionAmount(?it) \wedge present\text{–}at(?it, ?sa) \wedge S\_TransactionRejected(?sb) \wedge A\_TransactionRequestedRepeatedly(?sa, ?sb) \rightarrow Balance(?ib)$

- $S\_TransactionRequested(?sa) \wedge Balance(?ib) \wedge present\text{–}at(?ib, ?sa) \wedge TransactionAmount(?it) \wedge present\text{–}at(?it, ?sa) \wedge E\_AccountOverloaded(?eb) \wedge occur - in(?eb, ?sa) \wedge S\_AccountBlocked(?sb) \wedge A\_AccountSuspended(?sa, ?sb) \rightarrow Balance(?ib)$

Although some bigger rules can look too complex, most of the literals in it are type checking conditions which can be eliminated from the formulation by assuming a separate type checking algorithm.

6 Conclusion and Further Work

In this paper we presented ontological and logical considerations of knowledge representation. Also, the processing of transactions in dynamic systems, which involve synchronous and asynchronous activities such as events and actions have been described. We outlined a multi-level framework for modelling security policies for analysis. It is entirely based on the use of standard modelling languages of the Semantic Web, which greatly simplifies the implementation, makes it transparent and efficient. Our framework provides a theoretical basis for solving some of the hard problems in modelling dynamic behaviour. We utilize the concept of state, to provide a proper distinction between the static characteristics of the situations and the possible side effect of the actions on them. We have a pilot implementation of the framework as a Java program, which makes use of the APIs for OWL and SWRL in Jena for processing the ontological representation and the security policies in symbolic form [12], which allows us to perform various logical analytics related to risk assessment and policy validation.

We have successfully applied this framework to cross–channel transaction processing, in digital banking, for preventing social engineering fraud.

Currently, we are working on an extension of the framework with risk analysis capabilities, based on Bayesian theory. We are also exploring the potential use of the same framework in other domains, related to workflow control, production line fault recovery and safety management, like evacuation in the event of fire or other disastrous situations.

The semantic and logical considerations discussed above provide the formal ground for formalizing the concepts of accessibility, logical vulnerability and risks. Within our framework this can be done by simulating different scenarios for execution of the actions, under the conditions imposed on the situations and with possibility for events happening in them. Although such an analysis is beyond the scope of this paper, the experiments we conducted using our prototype implementation, have demonstrated that this approach is both transparent and convenient to be used for practical purposes [12].

References

1. Baader, F., Lutz, C., Miličic, M., Sattler, U., Wolter, F.: Integrating description logics and action formalisms: First results. In: Proc. of the 20th National Conference on Artificial Intelligence - Volume 2. pp. 572–577. AAAI'05, AAAI Press (2005)
2. Chang, L., Lin, F., Shi, Z.: A dynamic description logic for representation and reasoning about actions. In: KSEM (2007)
3. Eric Jizba, Yan Chen, F.S.G.P.: Web logic vulnerability, https://users.cs.northwestern.edu/~ychen/classes/cs450-s14/lectures/Web%20Logic%20Vulnerability.pdf, [Online; accessed January-2020]
4. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
5. Lawan, A., Rakib, A.: The semantic web rule language expressiveness extensions-a survey (03 2019)
6. McCarthy, J., Hayes, P.: Some philisophical problems from the standpoint of artificial intelligence. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence, vol. 4, pp. 463–502. Edinburgh University Press, Edinburgh, UK (1969)
7. Passin, T.B.: The Explorer's Guide to the Semantic Web. Manning Publications (2004)
8. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press (2001)
9. Sãnchez, D., Cavero, J.M., Marcos Martnez, E.: The Road Toward Ontologies, vol. 14, pp. 3–20. Springer US (2007)
10. Szeredi, P., Lukácsy, G., Benkő, T.: The Semantic Web Explained: The Technology and Mathematics Behind Web 3.0. Cambridge University Press, New York, NY, USA (2014)
11. Tsarkov, D., Horrocks, I.: Efficient reasoning with range and domain constraints. In: Proc. of the 2004 International Workshop on Description Logics (DL2004) (2004)
12. Vassilev, V., Sowinski-Mydlarz, V., Gasiorowski, P., Ouazzane, K., Phipps, A.: Intelligence graphs for threat intelligence and security policy validation of cyber systems. In: Proc. Int. Conf. on Artificial Intelligence and Applications (ICAIA2020). Advances in Intelligent Systems and Computing, Springer (2020)