A Multi-Agent System for Addressing Cybersecurity Issues in Social Networks*

Antonella C. Garcia[1,2,5,†], Maria Vanina Martinez[4], Cristhian A. D. Deagustini[1,5] and Gerardo I. Simari[2,3,4]

[1]*Fac. de Cs. de la Administración, Universidad Nacional de Entre Ríos (UNER), Argentina*
[2]*Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur (UNS), Argentina*
[3] *Instituto de Ciencias e Ingeniería de la Computación (UNS–CONICET), Argentina*
[4]*Artificial Intelligence Research Institute (IIIA-CSIC), Bellaterra, Spain*
[5]*Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Buenos Aires, Argentina*

Abstract
Constant interaction on social networks shapes individual decisions and behaviors, influencing both mental and social well-being. To address the complex and adversarial nature of these environments—marked by phenomena such as cyberbullying, grooming, and hate speech—this work examines the design of belief revision operators for processing streaming information within a multi-agent framework. Using the HEIST framework for hybrid socio-technical systems, we explore two knowledge dynamics operators: a cautious operator, which promotes stability in belief updates, and a credulous operator, which favors adaptability and rapid assimilation of new information. These operators define distinct reasoning behaviors for interacting agents engaged in continuous, dynamic, uncertain, and potentially adversarial information flows.

Keywords
Belief Revision, Stream Reasoning, Cybersecurity, Social Platforms

1. Introduction

Much of our daily activities are based on continuous direct or indirect interaction with information and news from the internet. This interaction generates large volumes of data, which are used for various purposes. Within these purposes, those with malicious intent deserve special attention, since they can cause harm in various areas and/or affect the decision-making processes of many people worldwide. This motivates research and development closely related to efforts in cybersecurity, understanding this area according to the general conception[1] including not only low-level information security issues, but also human-centered factors such as cyberbullying, hate speech, and attacks against mechanisms that make online trust possible. These represent key challenges and motivate this line of work [2, 3].

These problems, though different in nature, share the fact that solving or addressing them involve an effective use of incomplete, uncertain, and even biased knowledge. Therefore, cybersecurity is a broad area of research and practice that requires leveraging tools to address common issues for different types of attacks. Artificial Intelligence is useful in this context since it provides many basic tools that can be applied on their own or in combination towards the development of an effective and efficient solution. The model we propose in this work is intended to be used in the context of social platforms in general, allowing for systematic processing of a larger amount of data than what humans are capable of. Our model is an instantiation of the HEIST (Hybrid Explainable and Interpretable Socio-Technical Systems) [4] application framework, which provides the foundations developing systems that are

Workshop on the Foundations and Future of Change in Artificial Intelligence (FCAI 2025) at the 28th European Conference on Artificial Intelligence (ECAI 2025, October 25 - 30), Bologna, Italy
*Published in [1].
†Corresponding author.
antonella.garcia@uner.edu.ar (A. C. Garcia); vmartinez@iiia.csic.es (M. V. Martinez); ariel.deagustini@uner.edu.ar (C. A. D. Deagustini); gis@cs.uns.edu.ar (G. I. Simari)

[1]Cybersecurity can be informally defined as "the protection of systems (including software, hardware, or humans) connected to the internet".

capable of offering explanations about the decisions made. We propose a multi-agent system (MAS) of Supervisor Agents to supervise social platforms, seeking to detect malicious content and activities and respond so as to avoid or mitigate their effects. The following are two motivating domains.

Medical content. In this context a supervising system should be able to distinguish between a post with sexual content and a post that mentions sexual matters in a medical/health context. For example, it should prevent censorship of content related to breast cancer awareness—this would reduce false positives of sexual content moderation. It could adjust alerts for suspicious profiles against accounts that are whitelisted because they are known to disseminate alerts, educational content, awareness campaigns, etc. Currently, campaigns for breast cancer prevention cannot be freely shared as platforms censor images of women's breasts, hindering the dissemination of proper self-examination and warning signs.

Parental control. Supervising systems can be leveraged as tools applied by users themselves in specific platforms to exert personalized control. Such a system could be conceived as an extension to be used "on top of" social platforms, as is the case with Google's Family Link[2]. A mobile application could, based on what is displayed on the screen, show alerts or—if the user is a minor—send notifications to guardians. Another functionality to consider is the possibility that, with prior authorization, the device's supervising system send alerts to devices owned by children/guardians within the same class/school/group. This would generate what we will refer to as a *news item* for all such devices, and each corresponding agent would have the chance to decide what to do with that knowledge.

In [1] we made two contributions towards this goal: (i) we proposed a multi-agent system designed to address issues related to malicious behavior in social platforms; the system is based on the instantiation of a recently-proposed application framework for XAI in socio-technical systems; (ii) we proposed a novel kind of belief dynamics operator to guide the flow of knowledge within the framework, which we call stream-based belief revision; and (iii) we identified the hurdles that must be overcome for the development of effective and efficient stream-based revision operators. Preliminary in nature, [1] was meant to serve as a roadmap for ongoing and future research in the area of cybersecurity in social platforms. In this short paper, we will focus specifically on contributions (ii) and (iii).

2. Designing a Supervisor Agent Framework

In this section, we briefly recall the proposed multi-agent system. Figure 1 provides an overview of the proposed system. Each agent in the system is responsible for supervising a particular social network, and exchanges information through a centralized knowledge base called *Alerts-KB*, which serves as a repository for the agents to have up-to-date information on security alerts arising from different social networks. This knowledge repository is reviewed and updated based on the information shared by each agent in the system. As a result, the agents exchange information that enables them to proactively address potential cybersecurity issues that have already been identified in other networks.

Each supervisor agent maintains its own knowledge base of the social network it operates in. Agents engage in belief revision processes in order to update their own KB. Continuous belief revision processes enable agents to make informed decisions and implement appropriate actions in a timely manner.

Example 1. *A recurring security issue in social media platforms is the distribution of sexually suggestive images of the human body. Various security measures are in place to filter out images containing certain sensitive content, such as uncovered body parts, including nipples. These measures aim to address security concerns, but they also hinder the positive uses of such images, such as breast cancer prevention campaigns.*

Consider the scenario where a user called "Medical Center", validated as a health authority, posts a video showing a breast as part of a breast cancer prevention campaign. In the proposed model, the intelligent agent can consult the information about this user in the knowledge base and determine that they consistently share legitimate medical content. Based on this knowledge, it would be best not to censor this particular post, since it would allow for more effective prevention campaigns.

[2]https://families.google/familylink/

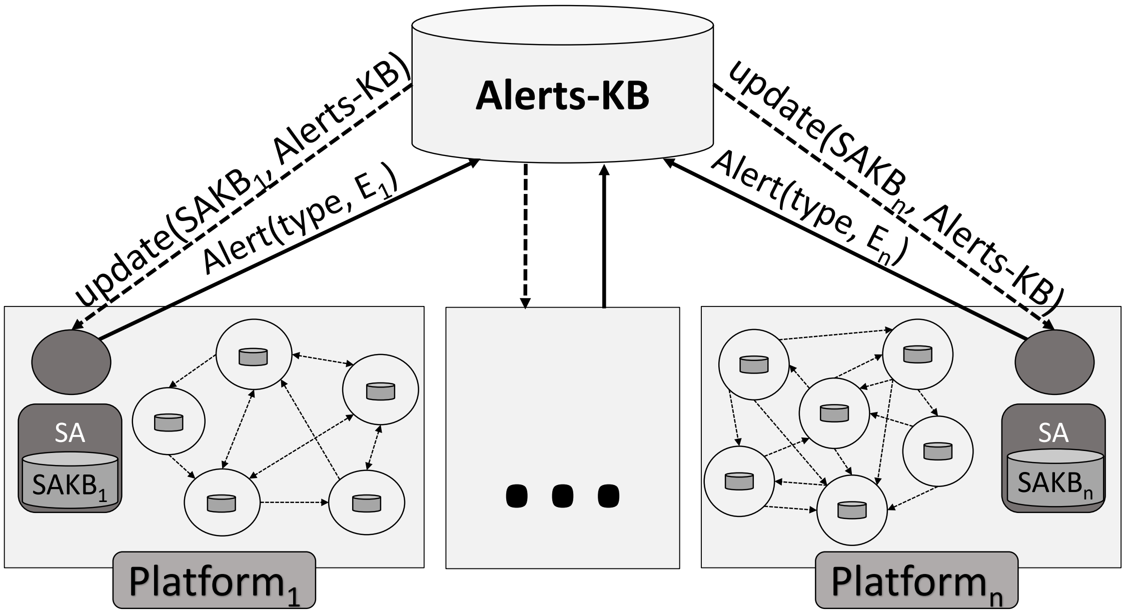

Figure 1: Proposed MAS for addressing cybersecurity issues in social platforms. Each platform has a dedicated *Supervisor Agent* that monitors it and interacts with central repository Alerts-KB.

2.1. Modeling Social Networks

In order to address cybersecurity issues in social platforms, we must be able to model users and their relationships, for which we first adopt the way social networks are modeled in [5]. According to this definition, a social network is a tuple consisting of four elements: a finite set of vertices, a finite set of edges, a vertex labeling function, and an edge labeling function:

A **Social Network** is a 4-tuple $(V, E, l_{vert}, l_{edge})$ where:

1. V is a finite set whose elements are called *vertices*.
2. $E \subseteq V \times V$ is a finite set whose elements are called *edges*.
3. $l_{vert} : V \rightarrow 2^{L_V}$ is a *vertex labeling function*, where L_V is a set of vertex labels.
4. $l_{edge} : E \rightarrow 2^{T}$ is an *edge labeling function*, where $T = \{\langle b, w\rangle \mid b \in L_E, w \in [0, 1]\}$ and L_E is a set of edge labels.

Considering the specific characteristics of social networks, it is crucial to model the different users and their relationships, i.e., the edges and vertices of the network, as well as the various interactions among them. The latter are events within the social network, and thus we must model a continuous sequence of events that contain relevant information for the system, which we refer to as a *data stream*.

Events within the social network can take different forms. We now mention the most common events in these environments and their specific characteristics. However, a particular social network may have additional types of events beyond those mentioned here.

Post. Each time a user creates new content, a Post event is created. This event reaches the vertices that are connected to the posting vertex. It consists of: `event ID`, `source (publisher)`, `text`, `multimedia element`, `set of tags`, and `timestamp`.

Share. Based on a Post event, a user can generate a Share event, which means sharing the original post with or without adding new data. Sharing increases the reach of the original post to the vertices connected to the Share-generating vertex. It contains the same elements as a Post event, plus a pointer to the ID of the original post: `event ID`, `source (Share generator)`, `text`, `multimedia element`, `set of tags`, `pointer to original post ID`, and `timestamp`.

Reaction. Refers to the reactions to a post, such as "like", "love", or other types of reactions depending on the platform. This event does not increase the reach of the original post but can influence its visibility to a greater number of users in their feeds. It consists of: `event ID`, `source (Reaction generator)`, `reaction type`, `pointer to original event ID`, and `timestamp`.

Comment. A comment is the addition of text to a previously generated post, either by the same user or another user. The event data includes: `event ID`, `source (Comment generator)`, `comment text`, `pointer to original event ID`, and `timestamp`.

Connection. Connections between users can be created or removed—each such occurrence is encoded as a Connection event (if two nodes are already connected, a Connection event encodes the removal of the edge). The event data includes: `event ID`, `source`, `target`, and `timestamp`.

As mentioned above, we adapt here the NKB model presented in [6, 5]:

NKBs. A Network Knowledge Bases (NKB) is a 5-tuple $(V, E, l_{vert}, l_{edge}, K)$ where the first four elements comprise a social network, and K is a mapping assigning a knowledge base to each vertex. Given v, $K(v)$ is called the knowledge base associated with vertex v.

In our model, we make a slight modification: whereas in NKBs as proposed by Gallo et al. the KB associated with each vertex is meant to encode the corresponding user's private beliefs, here it will group the posts made by that vertex and the interactions with posts from other vertices; $K(v)$ thus contains the events generated by that specific vertex; as shown in Figure 2, and as we discuss below, these KBs will be referred to as "SAKBs".

With the necessary tools in place for understanding the structure of the social network and characterizing specific aspects of the environment, we can proceed with the definition of our framework.

2.2. Supervisor Agents

Each Supervisor Agent (SA, for short) objective is to provide recommendations and/or make cybersecurity decisions based on the observation of the social network's structure that it is monitoring, and the data stream generated by its events. SAs operate in an alert state within the social network they supervise, and have access both to the NKB model of the corresponding social network and its data stream, so it can access all the events generated by vertices in the network.

Each agent SA_i maintains its own KB, which we will call $SAKB_i$, containing information about the social network and the security alerts occurring in that particular platform. $SAKB_i$ receives information from the NKB model of the social platform i and its data stream. As we will discuss in Section 3, the SA must dynamically keep its KB up to date based on this knowledge. Furthermore, the SA also updates its KB with security alerts from other social platforms sent to *Alerts-KB* by other agents. We propose the application of belief revision operators for these purposes.

Example 2. *Consider the network in Figure 2. The agent analyzes the data stream and observes that Beatriz made a post tagging Elsa that contains offensive language.*

Initially, the SA may decide to remain alert without taking action. At a later time, the SA observes that Daniel makes offensive comments against Elsa in the original post by Beatriz. Subsequently, Amelia and Clara share Beatriz's post, adding offensive comments. Based on this behavior, the SA raises an alert and issues warnings to the involved users.

Making the decisions described in Example 2 is the central problem faced by each SA—there is a wide range of possibilities, and exploring this in detail is outside the scope of this paper. Different alternatives can be formalized as policies that the SA can carry out within its network. A simple approaches based on thresholds (for instance, a three-strike rule that issues alerts after allowing two violations of a posting policy), or more complex schemes such as implementing a user classification mechanism, for instance based on user types as described in [6], that can be used to predict behaviors of interest.

Among the tasks agents must carry out, we have: (i) maintaining an updated SAKB specific to the social network it operates in; (ii) detecting potential threats or suspicious behaviors within the platform;

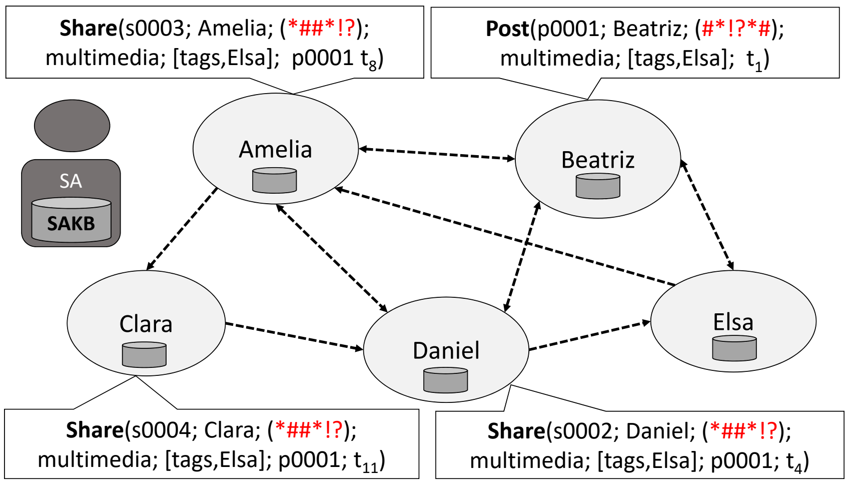
Share(s0003; Amelia; (*##*!?); multimedia; [tags,Elsa]; p0001 t8)
Post(p0001; Beatriz; (#*!?*#); multimedia; [tags,Elsa]; t1)
SA
SAKB
Amelia
Beatriz
Clara
Daniel
Elsa
Share(s0004; Clara; (*##*!?); multimedia; [tags,Elsa]; p0001; t11)
Share(s0002; Daniel; (*##*!?); multimedia; [tags,Elsa]; p0001; t4)

Figure 2: Events in social network example

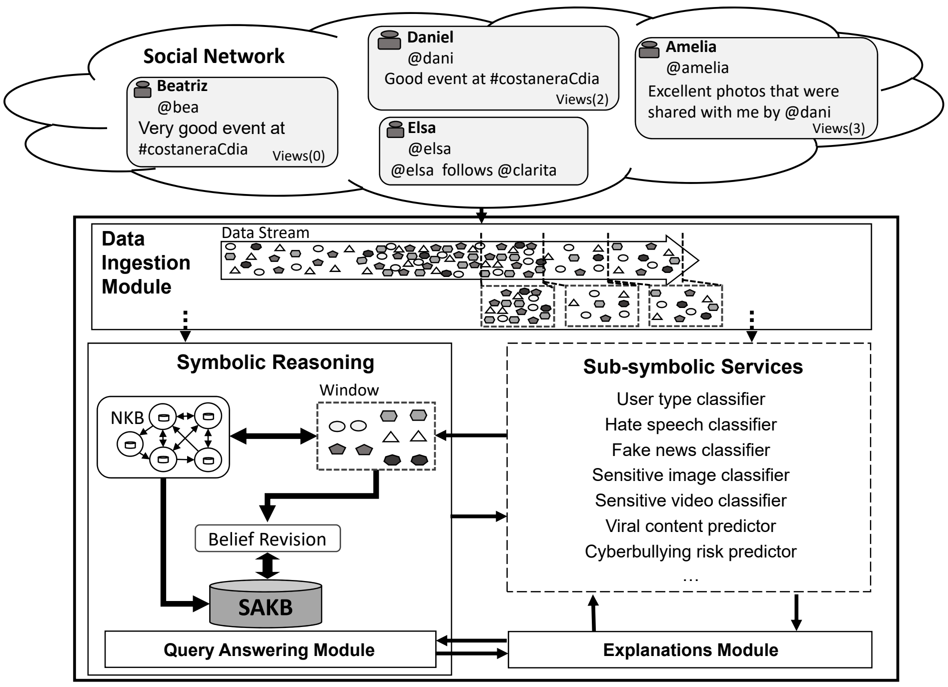

Figure 3: Overview of HEIST-SA

(iii) sending notifications for security-based decision-making, (iv) notifying the individuals involved and the responsible party about security measures taken, (v) sending updates of new security alerts to the *Alerts-KB*, and (vi) revising their knowledge based on updates to the *Alerts-KB* made by other SAs.

Based on the available knowledge, the SA could predict the viral effect of a post and recommend or implement security actions such as detecting negative viral effects, suspending users, managing the relevance level of posts, nullifying posts, or removing fake accounts. We instantiated and extended the framework in HEIST [3, 4] ans the basis for the architectural design of the Supervisor Agents. We choose this model for its flexibility in combining both symbolic and sub-symbolic tools, and because it explicitly considers explanations for query answers, which is central to cybersecurity applications.

Among all the modules that the architecture consists of, we will center our attention on the Symbolic Reasoning module and its specific challenges, as this module is the responsible of carrying out belief revision tasks based on inputs from the data stream.

Symbolic Reasoning. This module takes input from the Data Ingestion module and is thus responsible for implementing the stream reasoning [7] aspects, as well as maintaining the agent's SAKB. Concretely, the SA must perform stream reasoning-based belief revision in its SAKB as events occur in the social platform, seeking to detect malicious behavior. Specifically, the module receives a window from the stream, with which the NKB model is updated at the same time that the sub-symbolic services are applied to the events of that window. Rule-based approaches such as [8], or other formalisms based on computational logic, are good candidates for implementing such functionalities.

3. Stream-based Belief Revision

Belief revision is the problem of deciding how to react to epistemic inputs to a knowledge base [9, 10]. We now discuss how our agents perform belief revision tasks based on epistemic inputs coming from the data stream. First, we provide a more concrete definition of data stream.

Data Stream: The data stream produced by a social network S is a continuous, a priori unbounded, sequence of social network events, where each event is generated by a vertex belonging to S.

These streams contain information that needs to be processed promptly to extract knowledge as soon as relevant information becomes available [11]. The distinctive feature of data streams is that we cannot assume that their elements can be stored for later use. As a first step towards solving this problem, we need to address the processing of the data stream that the Data Ingestion module must perform.

We first briefly discuss basic issues that need to be addressed when considering implementations in this setting, then formalize the statement of the problem we wish to solve, discuss challenges that arise, and give two preliminary proposals for solving our problem.

3.1. Information Stream Processing

Given the nature of social networks and the large volume of data generated in short periods of time, conditions must be established to ensure that the processing of the data can be carried out in the best possible way. To this end, it is necessary to evaluate how stream reasoning [7] can be carried out in such settings so that the agent is capable of making the best possible use of the data stream produced by the associated social network, aiming to gain as much information as possible and respond to events of interest effectively. Next, we will consider different aspects related to handling the data stream from the social network that will enter the Data Ingestion module. As mentioned, data streams may contain inconsistencies, so we need to ensure that these are also processed in a way that provides the best handling of the data for the system's objective. Since we cannot store all the incoming data, we must "discretize" the stream. We now define various aspects of the data stream processing effort.

Data model: the data stream is comprised of events represented as tuples whose structure depends on the specific type of event, as discussed in Section 2.

Window: A subset of events from a data stream selected according to a given criterion. This is typically used to discretize streams [11]. In our case, they will allow us to limit the scope of the revision operators.

Windows can be either **logical**, which implement a selection criterion based on bounds over timestamps, or **physical**, which work with prefixed bounds on the number of tuples to be considered. We also need to address how the bounds of the windows are updated, which corresponds to how the window "moves" with time. Here, we will adopt the more general case of the *sliding* window, where both lower and upper bounds advance with the arrival of new items or the passage of time. A special case of this is the *tumbling* window, in which all items change each time the bounds are updated. The following are two different types of options for discretizing the social platform data streams.

Sliding pane logical window: windows are specified by a fixed time interval, allowing us to know the processing schedule. Note that windows may become overloaded with data during peak activity and can result in processing time that is greater than the validity of the window itself.

Sliding pane physical window: In this case, we cannot predict the speed at which the window is updated since it depends on the number of tuples that arrive in the stream. The advantage, of course, is that by knowing the number of items in each window, we can estimate how long it will take to process each window. On the other hand, we do not know now the frequency at which the window will be updated, which can again lead to problems during peak activity, as a large number of elements need to be processed in a short period of time and this may not be possible.

These two cases show that the processing model needs to define a load shedding policy [11], which essentially decides how to deal with data bursts or spikes in the stream by ignoring some of the data. In the general case, with several social platforms are involved, it seems unavoidable that such a policy be used since the volume of data streams varies depending on the number of active users on the platform at a given time, resulting in increased data volume during peak activity. We plan to study more precisely under which conditions each of these discretizations is most suitable for our system, what the impact of each one is in terms of effectiveness, and whether hybrid solutions might be possible.

3.2. Belief Revision: Problem Statement

Let K be an SAKB and w a window belonging to data stream *DS* of the social network *SN*. We define a *stream-based belief revision operator* ϵ as a function that takes K and w and produces a new SAKB K':

$$K' = \epsilon(K, w)$$

K' is obtained by applying operator ϵ to K with epistemic inputs from w arising from data stream *DS*.

Assuming a scenario with no computational resource limitations, applying ϵ would result in a consistent and updated K that could be handled with existing belief revision operators. However, this ideal scenario is not possible since in general we may not have enough time to process each window. Our system must therefore have principled mechanisms for deciding which elements in the window will not processed, and for this to be effective we must study how that impacts the result.

In the following section, we discuss several challenges that arise in practice: (i) real-time processing, (ii) out-of-order events, and (iii) event overload during peak activity. Given that classical belief revision operators—such as [9, 12, 13, 14, 15]—are not designed to work in this setting, their direct application would lead to one or more of such requirements to not be met.

3.3. Challenges in Stream-based Belief Revision

A straightforward way of implementing belief revision operators on streams would be to simply apply a traditional operator to the current window and continue doing so in an iterated manner. However, things fall apart when we consider the peculiarities of data streams. For instance, events in the stream may arrive out of order; applying an operator with incomplete information can generate different results than if we have all the information in a timely manner, and by the time the remaining data arrives it may be too late to correct the mismatch. Policies for handling out of order events will play a crucial role in deriving effective solutions to this problem, and their properties need to be thoroughly studied.

Since revision operators tend to be computationally costly [16], this leads to the problem of overload in windows where the volume of events is large or where windows are updated within short periods of time. While the operator processes the current window, an update may occur and the Supervisor Agent in this case would fall behind, causing a bottleneck in the operator and an outdated knowledge base. There are essentially three ways in which we may deal with this situation. First, we may implement a load shedding policy that simply chooses which elements are ignored so that the operator finishes in time. A second option is to develop a suite of operators, ranging from a lightweight option suitable for heavy loads to an ideal one that may be applied when time is available. Finally, as a compromise solution, we may consider developing something akin to “second order windows” where unprocessed elements are saved for later processing—though additional cost is incurred in terms of space, and results will not be available in a timely manner, correctness is not sacrificed.

3.4. First Steps towards a Solution

To tackle the challenges identified above, we consider two possibilities: (i) ignoring the unprocessed events (i.e., not addressing them at all), and (ii) allowing the KB to accept inconsistency by incorporating

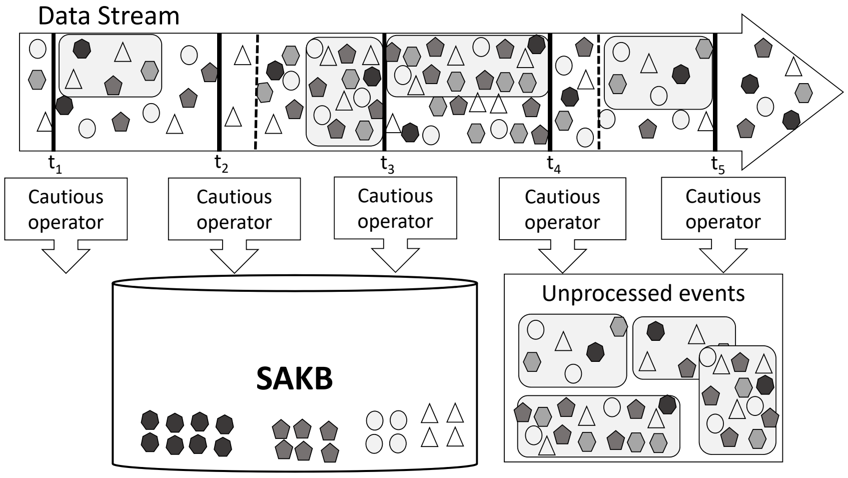

Figure 4: Discarding unprocessed knowledge from window.

the unprocessed events. Note that depending on the specific system load there may be windows in which all events can be processed resulting in the updated SAKB. We focus on the interesting case that corresponds to situations where the available time for window processing may be insufficient to process all events contained in the window.

In the following, in order to be able to use classical revision operators, we simplify the knowledge representation model and assume SAKBs and w are formulas in a propositional language $\mathcal{L}$. Furthermore, let $P(w)$ represent the set of elements in window w that have been processed so far, and $U(w) = w \setminus P(w)$ represent the set of elements in window w that have *not* been processed.

Option 1: *Ignore unprocessed events.* Let "$*$" be a classical multiple revision operator; a cautious operator Υ can be defined as follows:

$$\Upsilon(K, w) = K * P(w)$$

If the unprocessed knowledge ($U(w)$) from the current window is discarded, the SAKB will be consistent, and consequently, queries can be resolved using classical reasoning. This simplifies the processing of the SAKB, but it results in partial knowledge, since a significant number of events may be left out. This could include a multitude of attacks or events that could indicate potential threats, which would not be processed by the SA. This is illustrated in Figure 4.

Consider a scenario where a user on the social network is being targeted by other users, such as a case of cyberbullying. Since the SA does not process all of the events, it may only see a fraction of the comments and overlook the attack. Let's say there were 50 comments in the window, but the agent only processed 10 of them, along with other unrelated events. If we consider an agent that takes action when it detects 30 offensive comments, by processing only 10 comments, it would miss identifying the attack. This simple example highlights the drawbacks of this decision.

Option 2: *Allow inconsistency by incorporating unprocessed events.* Let "$*$" be a classical multiple revision operator, and "$+$" be a classical expansion operator; a credulous operator Φ is defined as follows:

$$\Phi(K, w) = K * P(w) + U(w)$$

Here, we incorporate the unevaluated knowledge into the SAKB, which may become inconsistent (cf. Figure 5). It is at inference or query time where inconsistency-tolerant methods need to be applied.

By incorporating unevaluated data into the SAKB, we may include information that appears to be a threat but is actually not. For example, we may have comments in a post that contain inappropriate words, but this may be consistent with their way of communicating. Since the SA could not evaluate this event and it was incorporated directly into the SAKB, this incident may play a role it wouldn't have if the window had been processed fully. In this case, the problem is pushed to the QA module since

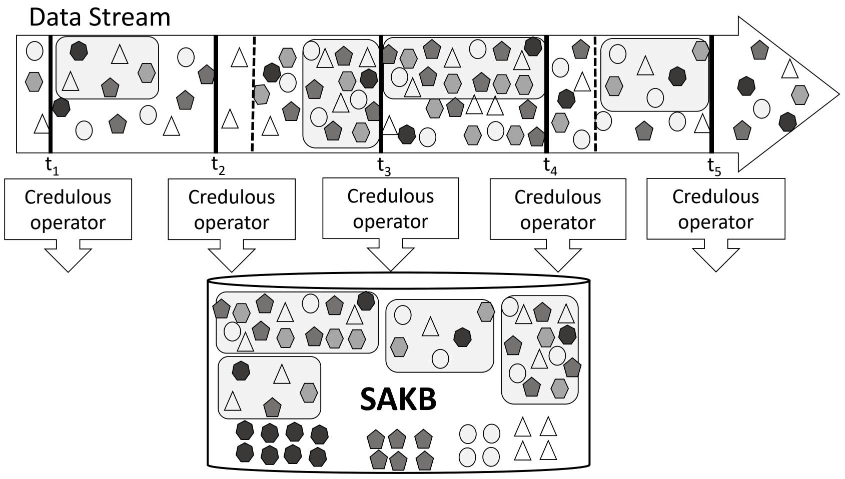

Figure 5: Incorporating unevaluated knowledge into K.

decisions made by the agent will be “contaminated” by unevaluated data. For instance, it would be necessary to define the level of confidence in the information provided by the AS. We could establish a semantics based on trust for conflict resolution. To achieve this, a measure indicating the level of confidence should be assigned to each piece of information and updated in each application of the revision operator. One possibility would be to consider a form of stratified SAKB [17, 15], where all the processed data forms the “hard” layer with a high level of confidence, and then having layers containing the unevaluated data from each window, potentially with different levels of confidence associated.

These operators could address the issue of event overload during peak activity, allowing (near) real-time processing. However, a policy for handling out-of-order events needs to be defined. As for concretization of the operators, as future work, we need to define postulates and constructs that allow us to apply belief revision in these environments. For some of the behavior, one could consider classical postulates (such as consistency) that do not necessarily have to be fully satisfied, but rather think about degrees of satisfaction that provide flexibility to better characterize real-world environments.

4. Conclusions and Future Work

This work outlines a preliminary approach to addressing cybersecurity challenges in social networks through a multi-agent system grounded in a recent application framework. We emphasize the central role of stream-based belief revision operators and the complexities introduced by different windowing strategies. The behavior of two classical operators highlights key limitations and motivates the need for new postulates and operator designs. Future efforts will focus on empirical validation and advancing this line of research toward trustworthy-by-design socio-technical systems.

Acknowledgments

This work was funded in part by Universidad Nacional del Sur (UNS) under grant PGI 24/ZN057, Secretaría de Investigación Científica y Tecnológica FCEN–UBA (RESCS-2020-345-E-UBA-REC), Universidad Nacional de Entre Ríos under grant PDTS-UNER 7066, CONICET under the grant PIP (11220200101408CO), and Agencia Nacional de Promoción Científica y Tecnológica, Argentina under grants PICT-2018-0475 (PRH-2014-0007) and PICT-2020 SERIE A-01481.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] A. C. Garcia, M. V. Martinez, C. A. Deagustini, G. I. Simari, A multi-agent system for addressing cybersecurity issues in social networks., in: ENIGMA@ KR, 2023, pp. 43–54.

[2] G. I. Simari, From data to knowledge engineering for cybersecurity, in: S. Kraus (Ed.), Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, ijcai.org, 2019, pp. 6403–6407.

[3] J. Paredes, J. C. Teze, M. V. Martinez, G. I. Simari, The HEIC application framework for implementing xai-based socio-technical systems, Online Soc. Networks Media 32 (2022) 100239. URL: https://doi.org/10.1016/j.osnem.2022.100239. doi:10.1016/j.osnem.2022.100239.

[4] J. C. L. Teze, J. Paredes, M. V. Martinez, G. I. Simari, Engineering user-centered explanations to query answers in ontology-driven socio-technical systems, Semantic Web (2023) 1–30 (*In Press*). URL: https://content.iospress.com/articles/semantic-web/sw233297. doi:DOI:10.3233/SW-233297.

[5] F. R. Gallo, G. I. Simari, M. V. Martinez, N. A. Santos, M. A. Falappa, Local belief dynamics in network knowledge bases, ACM Transactions on Computational Logic (TOCL) 23 (2021) 1–36.

[6] F. R. Gallo, G. I. Simari, M. V. Martinez, M. A. Falappa, N. A. Santos, Reasoning about sentiment and knowledge diffusion in social networks, IEEE Internet Comput. 21 (2017) 8–17.

[7] E. Della Valle, S. Ceri, F. Van Harmelen, D. Fensel, It's a streaming world! reasoning upon rapidly changing information, IEEE Intelligent Systems 24 (2009) 83–89.

[8] A. Ronca, M. Kaminski, B. C. Grau, I. Horrocks, The delay and window size problems in rule-based stream reasoning, Artificial Intelligence 306 (2022) 103668.

[9] C. E. Alchourrón, P. Gärdenfors, D. Makinson, On the logic of theory change: Partial meet contraction and revision functions, The journal of symbolic logic 50 (1985) 510–530.

[10] P. Gärdenfors, Knowledge in flux: Modeling the dynamics of epistemic states., The MIT press, 1988.

[11] G. Cugola, A. Margara, Processing flows of information: From data stream to complex event processing, ACM Computing Surveys (2012).

[12] S. O. Hansson, Kernel contraction, Journal of Symbolic Logic 59 (1994) 845–859. doi:10.2307/2275912.

[13] L. Amgoud, S. Kaci, An argumentation framework for merging conflicting knowledge bases: The prioritized case, in: Symbolic and Quantitative Approaches to Reasoning with Uncertainty: 8th European Conference, ECSQARU 2005, Barcelona, Spain, July 6-8, 2005. Proceedings 8, Springer, 2005, pp. 527–538.

[14] J. P. Delgrande, D. Dubois, J. Lang, Iterated revision as prioritized merging., KR 6 (2006) 210–220.

[15] M. A. Falappa, G. Kern-Isberner, M. D. Reis, G. R. Simari, Prioritized and non-prioritized multiple change on belief bases, Journal of Philosophical Logic 41 (2012) 77–113.

[16] P. Liberatore, M. Schaerf, Belief revision and update: Complexity of model checking, Journal of Computer and System Sciences 62 (2001) 43–72. URL: https://www.sciencedirect.com/science/article/pii/S0022000000916982. doi:https://doi.org/10.1006/jcss.2000.1698.

[17] G. Brewka, Preferred subtheories: An extended logical framework for default reasoning., 1989, pp. 1043–1048.