ARGUMENTATION FOR PROTECTING USERS' PRIVACY IN ONLINE SOCIAL NETWORKS

by

Nefise Gizem Yağlıkçı

B.S., Computer Engineering, TOBB University of Economics and Technology, 2014

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University

2016

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my thesis supervisor Prof. Pınar Yolum. I have learned so much during the two years that I have worked with her. She was always patient with me and encouraged me. I can sincerely say that she is one of the people who inspires me in life.

I would like to thank my thesis committee members Assist. Prof. Albert Ali Salah and Assist. Prof. Reyhan Aydoğan for accepting to be in my thesis committee. I would also like to thank Nadin Kökciyan for her support and valuable contributions to this work.

I have met with great people in the Computer Engineering Department. I would like to thank Berkant Kepez, Dilara Keküllüoğlu, Binnur Görer, Çağıl Uluşahin, Okan Aşık, Arda Çelebi, İlke Gültekin, Metehan Doyran, Cihan Camgöz, Çağatay Yıldız, Taha Ceritli, Alper Kamil Bozkurt, Hazal Koptagel and Yiğit Yıldırım for their valuable friendship and support.

I would also like to thank my dear friends Harun Serçe, Betül Kekik, Şeyda Çetintaş, Ayşegül Şahin, Merve Kaştan, Alena Beyer, Ahmet Gündüz, Ramazan Metkin and Çağlar Bayraktar for being a part of my life. I am grateful for their friendship and support.

Finally, I am most grateful to my dear parents and my brother for their love and support. I would not have all of the great things that I have today without them.

This work has been supported by the Scientific and Technological Research Council of Turkey (TUBİTAK) under grant 113E543.

ABSTRACT

ARGUMENTATION FOR PROTECTING USERS' PRIVACY IN ONLINE SOCIAL NETWORKS

Preserving privacy in online social networks (OSNs) is becoming increasingly important as more people expose their personal information in those networks. A social network user can easily share a post concerning other users. However, privacy constraints of these relevant users may be different from each other so that privacy violations occur. Therefore, all relevant users must be able to engage in a discussion and express themselves, so that they can protect their privacy.

We propose PriArg to enable relevant users discuss on a post using Assumption-Based Argumentation (ABA). We present each user in an OSN with an agent, which is equipped with an ontology. Ontologies involve the social network knowledge of users as well as their privacy constraints. Hence, agents are fully aware of the privacy constraints of their user and can act on behalf of them. When an agent wants to share a post, an argumentation session starts between the agent and other relevant agents. They provide each other arguments to express themselves and to convince other agents that their claim is true. At the end of the argumentation, we use an ABA engine to find whether sharing the post is justified according to the provided arguments. Agents may not have necessary information in their ontology to support their arguments. In this case, they can ask other agents in the social network for information. We evaluate our approach by showing its applicability with the real world scenarios. Moreover, we conduct a survey and an interview to compare the PriArg results with user expectations. At last, we compare the related work in the literature with PriArg.

ÖZET

ÇEVRİMİÇİ SOSYAL AĞLARDA KULLANICILARIN MAHREMİYETİNİN MÜNAKAŞA İLE KORUNMASI

Çevrim içi sosyal ağlarda mahremiyetin korunması, bu ağlarda kişisel bilgilerini paylaşan insanların çoğalmasıyla daha da önemli hale gelmektedir. Herhangi bir sosyal ağ kullanıcısı kolaylıkla başka bir kullanıcıyı ilgilendiren bir paylaşımda blunabilir. Fakat, paylaşım ile bağlantılı olan kullanıcıların mahremiyet kısıtları birbirinden farklılık gösterebilir ve bu durum mahremiyet ihlallerine neden olabilir. Bu nedenle tüm bağlantılı kullanıcılar, mahremiyetlerini korumak için kendi aralarında bir müzakere gerçekleştirmeli ve kendilerini ifade etmelidir. Bu çalışma, Varsayım-Tabanlı Münakaşa kullanarak kullanıcıların bir paylaşım üzerine müzakere etmelerini sağlayan PriArg'ı sunmaktadır. Bir çevrimiçi sosyal ağdaki her kullanıcı bir etmen tarafından temsil edilmektedir. Her bir etmen kullanıcıların sahip olduğu bilgileri ve mahremiyet kısıtlarını içeren bir ontoloji ile donatılmıştır. Bu sayede, her etmen kendi kullanıcısının mahremiyet kısıtlarından haberdardır ve onun yerine hareket edebilir. Bir etmen bir paylaşım yapmak istediğinde, bu etmen ile diğer tüm bağlantılı etmenler arasında bir münakaşa gerçekleşir. Etmenler, kendilerini ifade edebilmek ve karşılarındaki etmeni ikna edebilmek için birbirlerine argümanlar sağlarlar. Münakaşa sonunda paylaşımın yapılıp yapılmamasına karar vermek için, münakaşa sırasında sağlanan tüm bilgiler alınır ve bir Varsayım-Tabanlı Münakaşa motoruna verilir. Etmenler gerek duydukları her bilgiyi kendi ontolojilerinde bulamayabilirler. Bu durumda, sosyal ağ içerisindeki diğer etmenlere danışabilirler. Çalışmamız, PriArg'ın gerçek dünya senaryolarına uygulanabilirliği gösterilerek değerlendirilmiştir. Bunun yanısıra bir kişisel mülakat ve bir çevrim içi anket yürütülerek PriArg sonuçları kullanıcı beklentileri ile karşılaştırılmıştır. Son olarak, literatürde bulunan benzer çalışmalar ve PriArg karşılaştırılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a	Assumption
A	Assumption set
$aList$	List of predefined assumption predicates
as_X	Xth assumption
C	Contrary set
$C(a)$	Contrary of an assumption a
$contraryList$	List of the contraries
c_X	Xth contrary mapping
$fList$	List of predefined fact predicates
i	contrary of predicate p'
I	Inference Rules
i	Instance
$iList$	List of instances
I_{A_X}	Alice's Xth inference rule
I_{B_X}	Bob's Xth inference rule
I_{C_X}	Carol's Xth inference rule
L	Language that is used for Assumption-Based Argumentation
o	Ontology
p	Predicate
P	Privacy Rules
P_{A_X}	Alice's Xth privacy rule
P_{B_X}	Bob's Xth privacy rule
P_{C_X}	Carol's Xth privacy rule
R	Rule set
$rList$	List of rules
r_X	Xth fact
s	Received case
s'	Response case

status	Status of a case

LIST OF ACRONYMS/ABBREVIATIONS

AA	Argumentation Framework
ABA	Assumption-Based Argumentation
ADF	Argumentation Decision Framework
API	Application Programming Interface
CoPE	Collaborative Privacy Management
JSON	JavaScript Object Notation
MCDM	Multi Criteria Decision Making
OSNs	Online Social Networks
OWL	Web Ontology Language
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PriArg	Privacy Argumentation Framework
PriNego	Privacy Negotiation Framework
SWRL	Semantic Web Rule Language

1. INTRODUCTION

Online Social Networks (OSNs) are web-based services, which enable individuals to express themselves with an identity referred as *profile*, build connections with other individuals and view the information of others in the system [1]. OSNs gained a rapid popularity with the last decade as millions of people easily communicate from all over the world. The success of these social networks depends heavily on their popularity and on the amount of user data they have. Therefore, most of them encourage users to share more information about themselves and to have relations with more people [2] [3] [4]. Social networks generally enable users to have a profile to express themselves. A user can share her personal information such as her age, gender, nationality etc. Additionally, she can share posts such as photographs, videos, personal preferences, texts and locations. Even though there exists exceptions, users generally reveal their true identities within these profiles. Therefore, the information that they provide can be used by online crooks, cyberbullies or stalkers to abuse them [4]. Published information of users may cause harm even though there are no malicious users. In 2007, Sean Lane bought a diamond ring for his future wife. Even though he revealed no information about it, Facebook's Beacon feature broadcast this information in the social network and ruined the surprise. As a result, Facebook faced a lawsuit regarding privacy in the social media. Similar scenarios have caused people to lose their jobs [5] or court cases because of the information that is revealed unintended users. Moreover, information in the social networks has been used for job screening [6] and student monitoring [7] as well as behaviour tracking and even for government funded monitoring [8] [9]. These examples show how online social networks brought privacy concerns with it. Privacy is considered as one's right to control when, where and how her information is revealed to an audience [10]. A mechanism, which prevents unwanted information exposure is needed in OSNs to protect the privacy of users.

Privacy violations can occur in different ways. Kokciyan categorizes the privacy violations in a social network as in Table 1.1 [11]. In a social network, a post related to a user can be shared either by the user herself or by another user in the social network.

Table 1.1. Violation categories of current privacy policies.

	No Inference	**Inference**
User	(*i*) User causes a violation herself, since she cannot control the privacy settings or the OSNs reveal the user information without her consent	(*ii*) User shares a post that indirectly violates her privacy
Others	(*iii*) Other users in the social network share a post of the user that directly violates her privacy	(*iv*) Other users share a post that indirectly violates her privacy

Therefore, the user or another user can cause the violation. Another parameter to categorize the violation is directly or indirectly revealing the private information. Users can directly reach the information from a post or can make inferences based on the direct information. Remember the Beacon feature of Facebook, which revealed the information without user knowledge. The user himself gave the information into the system and the OSN revealed it without user confirmation. Audience did not use inference to reach private information. Hence, violation belongs to the category (*i*).

Social network sites generally provide users with *privacy policies*. These policies describe why the information of a user is collected, how it will be protected and how it will be shared. However, current privacy policies make it harder for a user to protect her privacy [12]. There are several reasons for that. Most of the time, it is cumbersome for a user to completely follow and understand these policies. Therefore, users usually are not aware of the privacy policies of social networks. In addition to their readability problem, policies change time to time. Hence, a social network user has to constantly follow the privacy policies to prevent possible violations. Being not aware of the privacy policies may cause a user to reveal her information to unintended users as in category (*i*).

Imagine a user who is fully aware of the privacy policy of a social network. It is still not certain that this user's privacy will be protected. Users need customized privacy constraints for different contents and audience. For example, a user may not

want to share her party pictures with her family, since she thinks they are embarrassing or she may not want to show her family pictures to her colleagues as she wants to keep her private life separate. On the contrary, another user may feel okay to share her pictures with all of the users in the social network. Different requirements of users in the social network call for customized privacy policies. However, in the current social networks users are not available to define their own privacy constraints. Instead, they choose from the privacy constraints that the OSN provides. For example, a user can set the audience of a post to friends, public, only me etc. These privacy constraints are static, which means that they cannot be changed automatically. Therefore, each time that a user wants to make a change, she has to update the settings manually. Further, she has to make these changes for all of the posts, which are considered by her as of the same type. For example, if there are two party pictures of a user, she cannot simply tell the OSN to apply the same constraints for both pictures. Instead, she has to make the same settings again.

Handling the privacy policies of social networks manually is a frustrating challenge for users. Considering the number of posts that are shared in the social networks every day, it is implausible to think that a user can follow and resolve all of the privacy violations. Therefore, an intelligent system that is aware of the privacy constraints of the user and can automatically act on behalf of them is needed.

A user's privacy may also be violated with the indirect information that is inferred from a post. Imagine a scenario where a user shared her picture in front of the tower of Pisa. Even though, the user did not put a location tag into the picture, her location will be understood by the audience (category (*ii*)). Therefore, privacy policies must also handle the violations that occur because of the inference mechanisms of users.

Social networks are structured as they enable users to express themselves and communicate with each other freely. However, most of the times users share posts that are not only concerning themselves. There may exist information in a post concerning other users. For example, when a user tags another user in a picture, she reveals the identity of that user in the social network (category (*iii*)). Moreover, when another

user sees the picture, she infers that these users are friends (category (*iv*)). For these reasons, privacy of the other related users also must be taken into account. We refer all of the users who are somehow related to the post as relevant users. Having more than one relevant user for a post brings the challenge of satisfying the constraints of each user. One approach may be applying the constraints of all relevant users to the post. However, most of the times privacy constraints of different users conflict with each other. Hence, applying all of the constraints may also cause a privacy violation. Another approach may be using a simple voting mechanism and applying the constraints with the most votes. Even though this approach solves the conflict problem, the less voted constraints will not be satisfied and there will again occur a privacy violation. For those reasons, users need a mechanism where all of them are able to contribute to the resulting policy and satisfied with the result. Note that a user can be satisfied with the resulting policy either when her privacy constraints are fulfilled or she is somehow convinced that her privacy constraints are unnecessary.

Current social networks do not let users discuss a post before sharing it. Hence, if a user is somehow related to a post, she does not have a right to express her concerns about the post before realizing its existence in the social network. Moreover, there is no mechanism for the user to directly intervene the post after she realizes its existence. The user can either ask the post owner to remove it or can complain to the social network administration about it. Even though she manages to remove the post, meanwhile other users in the social network will be able to see it. Hence, the privacy of the user will be violated. For this reason, a privacy protection mechanism must enable relevant agents to come to an agreement before the picture is revealed in an OSN.

A recent study shows that most users would have not share a post if they knew that it would violate the privacy of their friends [13]. This shows the need of a discussion where users can exchange their ideas and decide whether to share a post or not. Further, the discussion has to be conducted before the post is put up online. Several approaches in the literature follows this idea. Squicciarini *et al.* propose a collaborative approach for users to build privacy policies together [14]. However, in OSNs it is not plausible to expect users to handle the privacy policy of each post manually. Hence, users need

an automatic system, which handles the privacy settings for them. Mester *et al.* [15] and Such and Rovatsos [16] benefit from agents, which are computational mechanisms that can perceive, reason and act. Each agent represents a user in the social network and acts on behalf of the user. They propose *negotiation* to enable agents to discuss on a post and come to an agreement. However, agents are not able to question each other's claims. They accept the provided information as they are.

We propose an approach, which enables relevant users to discuss a post before it is revealed in the social network. Users together decide whether to share the post or not. We represent each user in the social network with an agent to take the burden of users. Each agent in the system is equipped with an ontology to represent the knowledge of the user. Moreover, agents use semantic rules to express the privacy constraints of their user. Agents conduct the discussion by using Assumption-Based Argumentation (ABA). When an agent wants to share a post, it finds the other relevant agents and starts an argumentation session. During the argumentation, agents provide arguments to each other to express their ideas. These arguments are created based on the information in their ontology. If an agent cannot find the necessary information in its ontology to create an argument, it can ask other agents in the social network for that knowledge. When an agent disagrees with another agent, it creates an argument, which rebuts (i.e., attacks) the arguments of the other agent. In that way, an agent can convince another agent that its claim is not true. Argumentation provides different semantics to calculate the justified arguments. Hence, it is able to handle the inconsistencies between different arguments. Moreover, since arguments are created based on the information of the agents, argumentation enables agents to understand why an argument is justified. At the end of the argumentation, we check if the post can be shared by using an ABA engine, *abagraph* [17].

Users need to be able to control their privacy policies to prevent any privacy violations. Current social networks provide users several options as privacy constraints but do not let them to create their own. Moreover, it is frustrating for users to follow and understand these policies. In our approach, semantic rules enable agents to create understandable policies. For example, an agent is able to create a semantic rule, which

indicates that a picture should not be shared, if it is taken in a party. In this example, the agent creates a customized rule for a specific situation. Moreover, the rule is generalized for all party pictures. Thus, the agent does not need to ask its user for a privacy setting for each party picture. Users can benefit from the semantic rules also to prevent the violations that occur because of the inference. For example, a user may create an inference rule stating that a picture reveals the friendship relation of users if they are both tagged in it. Then, her agent can reject all of the pictures, which reveal the friendship relation of the user.

Chapter 2 describes our semantic agent and how we represent the social network. Chapter 3 explains the Assumption-Based Argumentation and its usage in the privacy context. Chapter 4 describes our approach PriArg and explains how argumentation is conducted with our algorithm. Chapter 5 describes the implementation of PriArg. Chapter 6 evaluates PriArg by following three different approaches. First, it shows the applicability of PriArg with real world scenarios. Second, it provides an online survey and a personal interview to compare the PriArg results with user expectations. At last, it makes a qualitative comparison on PriArg and three most related works in the literature. Chapter 7 explains the related works in the literature and makes a comparison.

2. AGENT-BASED REPRESENTATION OF SOCIAL NETWORKS

It is possible for any user in an online social network to share a content relevant to other users. In this case, the content has more than one owner and privacy constraints of all users must be taken into account to prevent possible violations. One solution may be applying privacy constraints of all users to the post. However, each owner may have a different privacy setting for that post. To handle this issue, a mechanism is needed for users to express their privacy concerns and to communicate with each other to come to an agreement.

Consider the following scenarios where Alice wants to share a picture of Bob in a social network. The picture shows Bob with a wristband, which is obtained from the Oktoberfest.

Scenario 1. Form 1: Alice shares the post without consulting Bob.

Scenario 1. Form 2: Alice consults Bob about the post. Bob does not want his attendance to a festival to be known by others. He states that the wristband is given from the Oktoberfest and thinks that it is a unique object. Thus, he claims that the wristband will signal his attendance to the Oktoberfest and refuses to share the post.

Scenario 1. Form 3: Following Form 2, Alice claims that the wristband can also be found in a shop named $Gifty$. Thus, it is not unique and does not necessarily implies Bob's attendance to the Oktoberfest. Therefore, there is no harm to share the post.

Scenario 1. Form 4: Following Form 3, Bob does not have any information to support his claim. Thus, he consults another friend; Carol in the social network. Carol suggest Bob to check if the website of $Gifty$ can be reached. Bob checks the website and sees it cannot be reached. As a result, he comes to conclusion that $Gifty$ is closed. Hence,

the post should not be shared.

Most of the current social networks work as in Form 1. Content owners share a post without consulting relevant users. It is possible for a sensitive user to violate other users' privacy just as much as a careless user. Even though sensitive user cares about other users' privacy, she may not be aware of the privacy constraints of others and may not see a harm to share the post. This shows the need for a communication mechanism between users. Negotiation approaches in the literature [15, 16] can partially handle Form 2, as they enable requester agent to consult other relevant agents and get their response. However, these approaches are not sufficient to provide a detailed reason behind a response. Mester *et al.* enables relevant agents to give partial reasons such as the audience or the date of the post etc. Similarly, in the work of Such and Rovatsos agents only give the audience as their reason. However, agents need to be able to give more detailed reasons to convince the other agent. Moreover, these approaches mostly focus on coming to an agreement and do not enable agents to question other agents' actions. In Form 2, Bob gives a detailed reason for rejecting the post. He explains why the post is in the Festival context. Thus, Alice can check if the reasons of Bob are reliable and can be convinced by these reasons instead of directly accepting Bob's rejection. Form 3 and Form 4 show the situations where the users do not agree with each other. Hence, users provide arguments to express themselves and challenge each other's arguments to defend their claims. Arguments are created with the users' own knowledge or with the knowledge of their friends.

In current online social networks, if a user thinks her privacy is violated, she can either ask post owner to remove the post or can complain to the social network administration by providing reasons. Meanwhile her privacy will be already violated. Thus, it should be possible for users to discuss on a post *before* putting it up online. In addition to that, this process should be done automatically, since it is not reasonable for a user to handle each post in the social network by herself.

We propose an agent-based social network in which we represent each user with an agent. An agent is an autonomous computer program, which can perceive, reason,

act and communicate with other agents [18] [19] [20]. Agents can act on behalf of their users if they have their users' knowledge base. We represent the knowledge base of users with ontologies. Agents are equipped with a common ontology so that they talk the same language. Each agent personalizes the common ontology with its own privacy constraints and social network knowledge. Therefore, our agents are fully aware of the social network knowledge and privacy constraints of their users and can act on behalf of them.

2.1. Social Network Ontology

A social network consists of *users* in specific *relations* and *contents* that are shared among the users. We represent the social network using a Web Ontology Language (OWL). We benefit from the ontology model developed by Mester *et al.* [15]. Ontologies consist of concepts, data properties and object properties. Concepts represent a class of individuals (e.g. wristband `:wband` is an instance of *Object*). Data properties relate data values to individuals (e.g. *isOrdinary* relates `:wband` to either *true* or *false*). Object properties relate different individuals with a specific relation (e.g. *includesObject* relates a *Medium* to an *Object*).

An *Agent* makes a *PostRequest* including a *Medium* or a *PostText*. *hasMedium* and *hasText* object properties relate *Medium* and *PostText* to the *PostRequest*, respectively. A *Medium* such as a *Video* or a *Picture* may include an *Agent*, a *Location* or an *Object*. *includesPerson*, *includesLocation* and *includesObject* are used to relate corresponding concepts, respectively. In addition to that, a person may be mentioned in a *PostText* with *mentionedPerson* property or may be tagged with other people with *withPerson* property. A *PostRequest* is presented to an *Audience*, which is a group of users. *hasAudience* specifies the *Audience* of a *PostRequest* and *hasMember* specifies which users involved in the *Audience*.

In a social network, users connected to each other with *isConnectedTo* property. Since relationship type between users may vary, we need different relationship types in the ontology. We benefit from *isColleagueOf*, *isFriendOf* and *isPartOfFamilyOf*

subproperties to express the relationship type of users.

An essential concept in our ontology is *Context*. Each *PostRequest* is related to some *Context* with *isInContext* property. *Context* information is useful, since it may be used to define privacy rules. For example, a user may define a general privacy rule to not to share a picture in a *Party* context with her family members. In this case, agent uses *rejects* property to reject all post requests in the *Party* context. However, there are no strict rules to infer a context, since the same time and same location may mean different contexts for different people [21]. For example, a picture that is taken in a restaurant is in the *Leisure* context for a customer while it is in the $Work$ context for a waiter. Hence, we enable agents to define their own rules to infer context information. Imagine both the customer and the waiter do not want to show their pictures that are in the *Leisure* context. Since the agents use personalized inference rules to find the context information, the customer will reject to share the picture that is taken in the restaurant while the waiter will not.

2.2. Semantic Rules

We benefit from two types of semantic rules in our ontology: (1) inference rules (I) derive new information from existing knowledge in the ontology, (2) privacy rules (P) define what type of post requests should be rejected by the agent. Head of each privacy rule includes *rejects* property. We use Semantic Web Rule Language (SWRL) to define semantic rules [22]. Each rule is in the form of $Body \Longrightarrow Head$. Both body and head consist of concepts, data properties and object properties that are defined in the ontology. Thus, a rule can easily be constructed by using different concepts of ontology such as location, date, context. Further, it is also possible to use the combination of concepts to construct a rule (e.g. do not share a post taken in a specific time and location). Head of an SWRL rule is true if and only if its body is true.

Semantic rules can be defined by users themselves with a user friendly interface [15]. Moreover, they can be automatically learned by using some machine learning techniques [23] [24]. In our work, we assume that each agent is already aware of its

user's semantic rules.

Table 2.1. Semantic rules in Scenario 1 as SWRL rules.

I_{A_1}:	*foundAt*(?object, ?shop) → *isOrdinary*(?object, true)
I_{B_1}:	*isInContext*(?postRequest, ?context), *hasMedium*(?postRequest, ?medium), *includesObject*(?medium, ?object), `Oktoberfest`(?location), *obtainedFrom*(?object, ?location), *isOrdinary*(?object, false) → `Festival`(?context)
I_{B_2}:	*hasUrl*(?shop, ?url), *isAccessible*(?url, false) → *isClosed*(?shop, true)
P_{B_1}:	`Festival`(?context), *isInContext*(?postRequest, ?context) → *rejects*(`:bob`, ?postRequest)

Table 2.1 shows the semantic rules for Form 4 of Scenario 1. Alice only has one inference rule (I_{A_1}), which states that an object is not ordinary if it can be found at a shop. Bob has a privacy rule (P_{B_1}), which states that a post request in the festival context violates his privacy. Thus, must be rejected. Bob also has two inference rules. I_{B_1} states that if a post request has a medium that includes a unique object taken from `Oktoberfest`, then the post request is in the `Festival` context. I_{B_2} states that a shop is closed if it has an inaccessible url.

2.3. Reasoning

When an agent wants to share a post, it finds all relevant agents to the post and asks for their permission. Then, the relevant agents use their reasoning mechanisms to evaluate the post and come to a decision. To evaluate a post, an agent uses the direct and indirect information it has. Direct information is the information that an agent already has in its ontology, information that the requester agent provides within the post and information provided by the friends of the agent. Information that is provided by the requester agent may express the audience, tagged people, location, date of the post etc. (e.g. `:alice` tags `:bob` and states that there is an object in the post, which is obtained from the Oktoberfest). An agent can have indirect information by using its inference rules. Inference rules make it possible to derive new information from the existing knowledge (e.g. `:bob` infers that the uniqueness of the wristband

makes the post in the Festival context). In OWL ontology, properties may have different characteristics. For example, a property may be defined as transitive, symmetric, functional, equivalent, subproperty etc. These characteristics sometimes yield to find new information from the existing knowledge. An agent may benefit from these different characteristics to find indirect information. For example, imagine a post including a specific landmark. If an agent is aware of the city that this landmark is located in and if it is also aware of which country the city is located in, then it will know which country that the landmark signals even though there is no direct information about it in its ontology. It is also possible for an agent to obtain new information by using machine learning techniques. For example, an agent can detect the users in the post by using image recognition techniques.

An agent uses the information it obtains to decide whether a post violates its privacy or not. Agents use privacy rules to indicate when they reject a post request. If a privacy rule holds in an agent's ontology, then the post request should be rejected. For example, `:bob` defines P_{B_1} in Scenario 1. Since it thinks that a post is in the Festival context, P_{B_1} holds in its ontology and it rejects the post request by creating arguments.

When an agent is able to represent the social network knowledge and the privacy constraints of its user, it can use these information to protect the privacy of its user. Agents can involve in discussions with other agents to come to an agreement about sharing a post. We benefit from argumentation to enable agents to express themselves and to come to an agreement.

3. ARGUMENTATION FOR PRIVACY IN ONLINE SOCIAL NETWORKS

Argumentation is a technique to evaluate claims in an argument set by considering attack and support relations between arguments. In an online social network, agents can protect their privacy by expressing themselves with arguments. However, it is most likely to have inconsistencies between different arguments. Argumentation is able to handle the inconsistencies between arguments [25] [26] [27]. It provides computation mechanisms to calculate justified argument sets. Dung *et al.* propose Abstract Argumentation Framework (AA), which is a pair $\langle \mathcal{AR}, attacks \rangle$ where $\mathcal{AR}$ is a set of arguments and *attacks* is the attack relations between arguments [28]. One can resolve the conflicts between arguments and find the justified argument sets by using the proposed computation mechanisms of AA. However, AA does not give information about how an argument is constructed and why an attack is made. We benefit from structured argumentation to have a finer-grained level of representation of arguments as well as attacks. ABA [29], ASPIC+ [30], DeLP [31], and deductive argumentation [32] are different frameworks in the literature for structured argumentation. ASPIC+, DeLP, and deductive argumentation allow to use preferences to resolve attacks. On the other hand, ABA maps the different forms of reasoning that includes preferences onto ABA framework, which does not include preferences. In this way, argumentation framework remains as simple as possible while it also supports the different reasoning forms. Similarly, many structured argumentation approaches allow different types of attacks. ABA maps these attacks to its standard attack form so that the framework remains as simple as possible while it allows for different type of attacks. ABA framework also provides different implementations such as $CaSAPI$ [33], *proxdd* [34] and *abagraph* [17] so that the result of an argumentation can be calculated automatically.

3.1. Assumption-Based Argumentation for Privacy

We benefit from ABA framework to build and discuss arguments. ABA framework is a tuple $\langle \mathcal{L},\mathcal{R},\mathcal{A},\mathcal{C} \rangle$ where $\mathcal{L}$ is the language, $\mathcal{R}$ is the set of rules, $\mathcal{A} \subseteq \mathcal{L}$ is the set of assumptions and $\mathcal{C}$ is the total mapping of assumptions with their contraries. Each rule in $\mathcal{R}$ consists of a body $\sigma_1,...,\sigma_m$ and a head σ_0 where $\sigma_1,...,\sigma_m \rightarrow \sigma_0$ ($m \geq 0$, $\sigma_i \in \mathcal{L}$). The assumption set $\mathcal{A}$ includes the weak points of arguments that can be attacked by other arguments. In ABA framework, $\mathcal{A}$ should be non-empty. Otherwise, there is no point of argumentation, since there is no weak point to attack. Each assumption has to have one single contrary, which defines how to rebut the assumption. The contrary set $\mathcal{C}$ involves the contrary of each assumption where a contrary of an assumption a is represented as $\mathcal{C}(a)$.

One can either hypothesise an assumption to construct an argument or may derive it by using rules with other assumptions. If ABA framework does not include derived assumptions then it is defined as $flatt$. Since non-flatt ABA frameworks are computationally demanding, current computation mechanisms for calculating justified arguments are defined for flatt ABA frameworks. Thus, we also use $flatt$ ABA frameworks in our work and do not have assumptions in the head of rules.

In current social networks, content owners share a post based on their own privacy constraints. However, privacy constraints of relevant agents must be also considered to prevent privacy violations. ABA framework enables agents to discuss over a content by creating arguments. Hence, we describe each privacy scenario as an ABA specification. Table 3.1 shows the ABA specification for Scenario 1. $\mathcal{L}$ is the language used in the ontology. $\mathcal{R}$ is the set of rules provided by agents in Table 2.1. In ABA framework, assumptions are questionable whereas facts are not. Thus, one can decide whether something is an assumption or a fact by considering if it is arguable. In our example scenario, r_1 - r_7 are not arguable for agents and are defined as facts. Facts are considered as rules with an empty body. Thus, r_1 - r_7 are placed in $\mathcal{R}$ in the specification. $\mathcal{A}$ includes four assumptions (as_1, as_2, as_3, as_4) that agents made during the argumentation. An argument can attack to an assumption if and only if it can support the

Table 3.1. ABA specification of Scenario 1.

$\mathcal{R} = I_{A_1} \cup I_{B_1} \cup I_{B_2} \cup P_{B_1} \cup_{i=1}^{7} r_i$
$r_1 = \{\rightarrow \mathit{isInContext}(\texttt{:pr},\texttt{:context})\}$
$r_2 = \{\rightarrow \mathit{hasMedium}(\texttt{:pr},\texttt{:medium})\}$
$r_3 = \{\rightarrow \mathit{includesObject}(\texttt{:medium},\texttt{:wband})\}$
$r_4 = \{\rightarrow \texttt{Oktoberfest}(\texttt{:location})\}$
$r_5 = \{\rightarrow \mathit{obtainedFrom}(\texttt{:wband},\texttt{:location})\}$
$r_6 = \{\rightarrow \mathit{taggedPerson}(\texttt{:medium},\texttt{:bob})\}$
$r_7 = \{\rightarrow \mathit{hasUrl}(\texttt{:Gifty},\texttt{:url})\}$
$\mathcal{A} = \{as_1,as_2,as_3,as_4\}$
$as_1 = \mathit{foundAt}(\texttt{:wband},\texttt{:Gifty})$
$as_2 = \text{not}(\mathit{rejects}(\texttt{:alice},\texttt{:pr}))$
$as_3 = \mathit{isOrdinary}(\texttt{:wband},\text{false})$
$as_4 = \mathit{isAccessible}(\texttt{:url},\text{false})$
$\mathcal{C} = \{c_1,c_2,c_3,c_4\}$
$c_1 = (\mathit{foundAt}(\texttt{:wband},\texttt{:shopX}) = \mathit{isClosed}(\texttt{:Gifty},\text{true}))$
$c_2 = (\text{not}(\mathit{rejects}(\texttt{:alice},\texttt{:pr})) = \mathit{rejects}(\texttt{:bob},\texttt{:pr}))$
$c_3 = (\mathit{isOrdinary}(\texttt{:wband},\text{false}) = \mathit{isOrdinary}(\texttt{:wband},\text{true}))$
$c_4 = (\mathit{isAccessible}(\texttt{:url},\text{false}) = \mathit{isAccessible}(\texttt{:url},\text{true}))$

contrary of that assumption. $\mathcal{C}$ includes the mapping between assumptions and their contraries. For example, `:alice` assumes that there is no need to reject the post request `:pr` and states that its assumption can be rebutted if `:bob` rejects the post request. This mapping specified in c_2 with assumption as_2 and its contrary. In ABA framework, defeasibility only comes with assumptions. Thus, rules are not questionable. A rule has defeasibility if and only if it has one or more assumptions in its body. Otherwise, it is strict.

An argument has the form $S \vdash^R \sigma$ where $S \subseteq \mathcal{A}$, $R \subseteq \mathcal{R}$ and $\sigma \in \mathcal{L}$. S is a set of assumptions, R is a set of rules and σ (the claim) is a derivation. An argument is supported by both assumptions and rules. A claim is derived using a rule or a chain of rules (e.g., $R_1, R_2, R_3 \in R$, $R_3 = R_1 \cup R_2$). An argument can be supported by an assumption a and an empty set of rules with the form $\{a\} \vdash a$. Moreover, an argument can be supported by an empty set of assumptions and a rule r with the form $\{\} \vdash^r h$. Consider h as the head of the rule r.

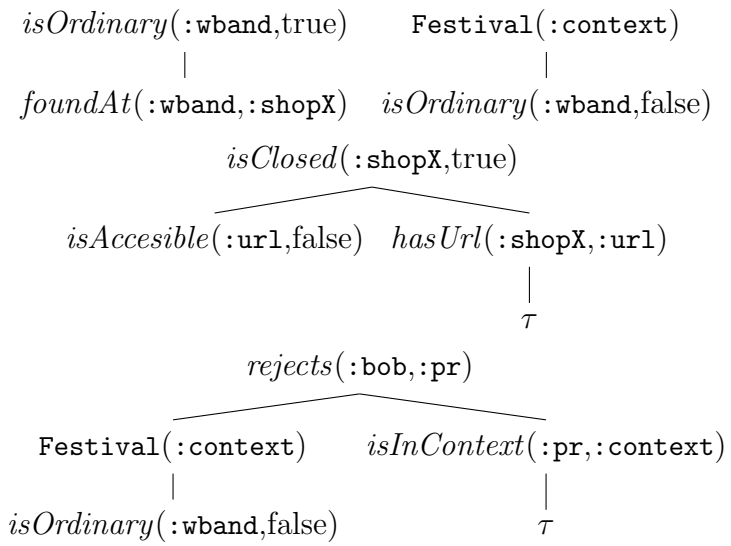


Figure 3.1. Deduction trees for arguments a_3, b_5, b_2 and b_3 in Scenario 1.

The arguments are derived from deduction trees. Figure 3.1 shows the deduction trees for the arguments a_3, b_2, b_3 and b_5. Each deduction tree has the head of a rule as its root. The leaves can either be τ (true) or an assumption that supports the root. In the deduction tree for argument a_3, assumption *foundAt*(`:wband`,`:Gifty`) supports *isOrdinary*(`:wband`,false) with the rule I_{A_1}. The assumption *isOrdinary*(`:wband`,false) and the

facts $\cup_{i=1}^{5} r_i$ support the `Festival(:context)` with the rule I_{B_1}. *isClosed*(`:Gifty`,true) derived with the assumption *isAccessible*(`:url`,false) and the rules $I_{B_2} \cup r_7$. P_{B_1} derives the conclusion *rejects*(`:bob`,`:pr`) with another derivation `Festival(:context)` and with the fact *isInContext*(`:pr`,`:context`). Thus, *rejects*(`:bob`,`:pr`) is the root of the tree for b_3. Facts have τ in their leaves, since they are supported by an empty set of assumptions (e.g. *isInContext*(`:pr`, `:context`) and *hasUrl*(`:Gifty`, `:url`) has τ in their leaves). For clarity, we do not show the supporting facts for argument b_5.

Table 3.2. Arguments derived from Scenario 1.

$f_1 : \{\} \vdash^{r_1}$ *isInContext*(`:pr`,`:context`)
$f_2 : \{\} \vdash^{r_2}$ *hasMedium*(`:pr`,`:medium`)
$f_3 : \{\} \vdash^{r_3}$ *includesObject*(`:medium`,`:wband`)
$f_4 : \{\} \vdash^{r_4}$ `Oktoberfest(:location)`
$f_5 : \{\} \vdash^{r_5}$ *obtainedFrom*(`:wband`,`:location`)
$f_6 : \{\} \vdash^{r_6}$ *taggedPerson*(`:medium`,`:bob`)
$f_7 : \{\} \vdash^{r_7}$ *hasUrl*(`:shopX`,`:url`)
$a_1 : \{$*foundAt*(`:wband`,`:shopX`)$\} \vdash$ *foundAt*(`:wband`,`:shopX`)
$a_2 : \{$not(*rejects*(`:alice`,`:pr`))$\} \vdash$ not(*rejects*(`:alice`,`:pr`))
$a_3 : \{$*foundAt*(`:wband`,`:shopX`)$\} \vdash^{I_{A_1}}$ *isOrdinary*(`:wband`,true)
$b_1 : \{$*isOrdinary*(`:wband`,false)$\} \vdash$ *isOrdinary*(`:wband`, false)
$b_2 : \{$*isOrdinary*(`:wband`,false)$\} \vdash^{I_{B_1} \cup_{i=1}^{5} r_i}$ `Festival(:context)`
$b_3 : \{$*isOrdinary*(`:wband`,false)$\} \vdash^{I_{B_1} \cup P_{B_1} \cup_{i=1}^{5} r_i}$ *rejects*(`:bob`,`:pr`)
$b_4 : \{$*isAccessible*(`:url`,false)$\} \vdash$ *isAccessible*(`:url`,false)
$b_5 : \{$*isAccessible*(`:url`,false)$\} \vdash^{I_{B_2} \cup r_7}$ *isClosed*(`:shopX`,true)

Table 3.2 shows the arguments derived from the specification of Scenario 1. f_i represents the fact arguments that are supported by an empty set of assumptions and a rule r_i. a_1, a_2, b_1 and b_4 are assumption arguments that are supported with the corresponding assumption and an empty set of rules.

Attack relations are used to express the disagreement between arguments. In ABA, an argument $S_1 \vdash \sigma_1$ can attack another argument $S_2 \vdash \sigma_2$ if and only if σ_1 is the contrary of one of the assumptions in S_2 [29], [35]. Figure 3.2 shows the attack

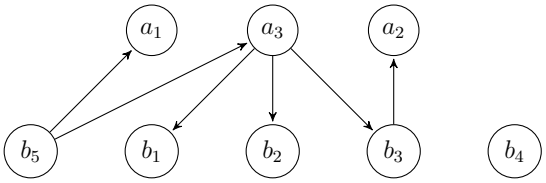


Figure 3.2. Attacks between arguments for Scenario 1.

relations between arguments from Table 3.2. We do not show the fact arguments (f_i) in the figure, since they do not have weak points and cannot be attacked by other arguments. b_3 attacks a_2, since its claim *rejects*(`:bob`, `:pr`) is the contrary of not(*rejects*(`:alice`, `:pr`)), which supports a_2. As a response, a_3 attacks b_1, b_2, and b_3, since *isOrdinary*(`:wband`, true) rebuts the assumption of *isOrdinary*(`:wband`, false), which supports b_1, b_2 and b_3. At last, b_5 attacks a_1 and a_3, since *isClosed*(`:Gifty`,true) is the contrary of *foundAt*(`:wband`, `:Gifty`), which supports a_1 and a_3.

3.2. Semantics for ABA

After having arguments and attack relations between them, one can calculate justified argument sets by using ABA semantics. A justified argument set can be *credulously justified* or *sceptically justified*. Credulous semantics allow for alternative winning argument sets while sceptical semantics only allow for a unique winning argument set. Thus, it depends on the application which one to use. If resulting argument set is too critical for the application, it is better to use sceptically justified semantics as it will provide an uncontroversial argument set. However, it may be also critical to find at least one justified argument set under all circumstances. If this is the case, credulously justified semantics may be better for the application. For example, when there is a tie between mutually attacking arguments, sceptical semantics will return an empty set of winning arguments. On the other hand, credulously justified arguments will return two result sets for each argument. In our work, we need to return a result to the agents under all circumstances. Thus, we use credulous semantics to find all alternatives of justified argument sets. There exists other semantics to find the justified arguments out of arguments with attack relations. A set of arguments $\mathcal{A}$ is

- *conflictfree* iff none of its arguments attack it.
- *admissible* iff it is conflict free and it can defend itself against all attacks.
- *preferred* iff it is the maximum admissible set.
- *complete* iff it is admissible and has all arguments it defends.
- *grounded* iff it is the minimal complete set.
- *ideal* iff it is preferred and all preferred sets contain it.
- *stable* iff it is conflict free and it attacks all arguments outside of it [35].

Table 3.3. Justified arguments in Scenario 1 according to different semantics.

	Credulously Justified	**Sceptically Justified**
Admissible	$\{\}$, $\{b_5\}$, $\{b_4\}$, $\{b_4, b_5\}$, $\{b_3, b_5\}$, $\{b_3, b_4, b_5\}$, $\{b_2, b_5\}$, $\{b_2, b_3, b_5\}$, $\{b_2, b_4, b_5\}$, $\{b_2, b_3, b_4, b_5\}$, $\{b_1, b_5\}$, $\{b_1, b_4, b_5\}$, $\{b_1, b_3, b_5\}$, $\{b_1, b_3, b_4, b_5\}$, $\{b_1, b_2, b_5\}$, $\{b_1, b_2, b_4, b_5\}$, $\{b_1, b_2, b_3, b_5\}$, $\{b_1, b_2, b_3, b_4, b_5\}$	$\emptyset$
Complete	$\{b_1, b_2, b_3, b_4, b_5\}$	$\{b_1, b_2, b_3, b_4, b_5\}$
Stable	$\{b_1, b_2, b_3, b_4, b_5\}$	$\{b_1, b_2, b_3, b_4, b_5\}$
Grounded	$\{b_1, b_2, b_3, b_4, b_5\}$	$\{b_1, b_2, b_3, b_4, b_5\}$
Preferred	$\{b_1, b_2, b_3, b_4, b_5\}$	$\{b_1, b_2, b_3, b_4, b_5\}$
Ideal	$\{b_1, b_2, b_3, b_4, b_5\}$	$\{b_1, b_2, b_3, b_4, b_5\}$

Any of these semantics can be used to find the justified argument sets. Table 3.3 shows the justified argument sets for Scenario 1 according to different semantics. Notice that there is no argument in a justified argument set that attacks another argument in the same set. It is because all of the semantics are conflict free. A set is considered as defending itself, if no argument attacks it or arguments attack it but at least one of its arguments can attack these arguments back. a_1, a_2 and a_3 cannot defend themselves against the attacks. Therefore, they cannot be in an admissible argument set. However, b_1, b_2 and b_3 can be in an admissible set if the set also contains b_5, which defends b_1, b_2 and b_3. Since b_4 and b_5 is not attacked,

they can be in an admissible set. The credulously admissible set with the highest number of elements is $\{b_1, b_2, b_3, b_4, b_5\}$. Thus, it is considered as a credulously preferred set. If an argument is not attacked by any other argument, then it is considered to be defended by an empty set of arguments and by all of the other argument sets. Therefore, b_5 and b_4 are defended by other argument sets. Moreover, they defend each other and both of them have to be included in all complete argument sets. b_5 also defends b_1, b_2 and b_3. Therefore, $\{b_1, b_2, b_3, b_4, b_5\}$ is considered as a credulously complete argument set. There is no other credulously complete argument set with less number of arguments. Thus, $\{b_1, b_2, b_3, b_4, b_5\}$ is also the credulously grounded argument set. Since $\{b_1, b_2, b_3, b_4, b_5\}$ is the only credulously preferred argument set, it contains all of the preferred argument sets. Therefore, it is also the ideal argument set. $\{b_1, b_2, b_3, b_4, b_5\}$ is a conflict free argument set and the arguments inside of it cannot attack each other. Thus, subsets of $\{b_1, b_2, b_3, b_4, b_5\}$ cannot be credulously stable argument sets. However, $\{b_1, b_2, b_3, b_4, b_5\}$ attacks all of the arguments that are not inside of. Therefore, it is a credulously stable argument set. Notice that only admissible semantics provide more than one justified set for this example. In other words, it provides alternative result sets. Sceptically justified semantics do not allow alternative result sets. Therefore, the sceptically justified admissible result set is $\emptyset$. On the other hand, complete, stable, grounded, preferred and ideal semantics provide only one result set. Therefore, these result sets are also sceptically justified.

There are two reasons for us to use admissible semantics. First, *abagraph* only provides AB-dispute derivations and GB-dispute derivations, which find if a claim is admissible or grounded respectively. Second, admissible semantics are good enough for us to give a decision. We do not need further restrictions for our result set.

Definition 1. *The requesting agent* `:agent` *shares a post* `:pRequest` *iff an argument* $A \vdash$ *not(rejects(*`:agent,:pRequest`*)) is in the justified admissible argument sets, where* $A \subseteq \mathcal{A}$.

In our work, after having arguments and attack relations between them, we do not find all of the justified argument sets. Instead, requesting agent queries *abagraph*

with its first assumption (e.g. not(*rejects*(`:alice`,`:pr`)) and *abagraph* returns if it is justified or not according to the required semantics. In Figure 3.2, b_3 attacks a_2. Then, a_3 defends a_2 by attacking b_3. However, since b_5 rebuts a_3, a_2 cannot defend itself against attacks and cannot be justified according to credulously admissible semantics. Definition 1 shows how agents decide to share a post or not. Considering Definition 1, Alice does not share the post.

abagraph uses dispute derivations to calculate if an input sentence is justified or not. Dispute derivations can be considered as a zero-sum game between a proponent and an opponent. Proponent tries to prove that the input sentence is justified while opponent tries to rebut it. If a dispute derivation is successful, then it returns: (1) a *defence set*, which includes the assumptions of the proponent, (2) *culprits*, which includes the assumptions of opponent that are attacked by the proponent and (3) *dialectical trees* of arguments (e.g., derivation tree at Figure 3.1).

4. DISTRIBUTED ARGUMENTATION

In Chapter 3, we explained the ABA framework and showed how it can be used for privacy. When an ABA specification such as the one shown in Table 3.1 is provided to an ABA engine, it is possible to find if an input sentence is justified or not. One may try to create such specification centrally. However, in our domain we try enable relevant agents to come to an agreement. Therefore, we need the information of all relevant agents while creating the specification. In Scenario 1, `:alice` would have put only its information into a specification, give the specification into the *abagraph* and decide whether she should share the post. In this case, `:alice` would have not consider the privacy of `:bob`. In our approach, we adapt a distributed argumentation structure, where information comes from more than one agent to make a decision. Therefore, `:alice` starts an argumentation with `:bob` to prevent any privacy violations and the specification is created with the information of both `:bob` and `:alice`.

Different agents can provide different rules, facts, assumptions and contraries, which lead different arguments and attack relations. Moreover, agents do not have to follow the same strategy. A well-meant agent may only provide relevant information into the argumentation while another agent provides irrelevant or fictious information to sabotage the argumentation. Further, an agent may try to contribute in every argumentation that it is involved in while another agent only contributes in argumentations that are conducted with the agents it trusts. For that reason, two agents may lead different ABA specifications even though they have the exact same knowledge.

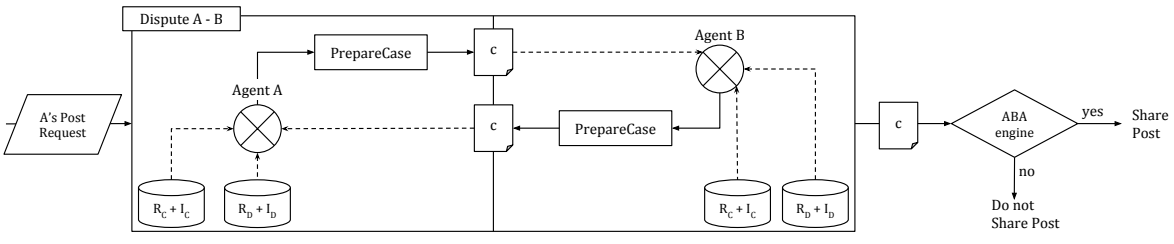


Figure 4.1. Distributed model of PriArg for Agent A and Agent B.

We propose *PriArg* (Privacy Argumentation Framework) to protect the privacy of users with argumentation. Figure 4.1 shows the distributed model of our approach. When an agent (*AgentA*) wants to share a post, it makes a post request to the relevant agents and starts an argumentation session. Agents communicate by sending a case (s) to each other (see Definition 2).

Definition 2. *A case is a tuple* $\langle \mathcal{R}, \mathcal{A}, \mathcal{F}, \mathcal{C}, status \rangle$ *where* R *is a set of rules,* A *is a set of assumptions,* F *is a set of facts,* C *is the assumption contrary mapping and status is either ongoing or stop. A case status specifies if the argumentation should stop or continue.*

When *AgentA* starts the argumentation, it prepares an initial case with *ongoing* status and sends it to the relevant agents. A relevant agent (*AgentB*) receives the initial case and evaluates it in its ontology. If the post does not violate the agent's privacy, it sends the case back with *stop* status and ends the argumentation. Otherwise, it tries to create arguments to oppose the sharing request. Therefore, it extends the initial case with new facts, assumptions, contraries and rules. Then, it sends the case back to the requester agent. Agents send the case each other in a turn taking fashion, until there is no more contribution to the dispute. After argumentation stops, requester agent gives the case into the ABA engine and queries if its initial assumption (e.g. not(*rejects*(`:alice`,`:pr`))) to share the post is valid. If the ABA engine returns true, the post is shared. Otherwise, it is not shared, since the relevant agent convinces the requester agent to not share the post.

4.1. Attacking Postrequests

An agent creates arguments to attack other agents' claims and to support its own. As explained in Chapter 3, an argument consists of assumptions with rules (with an empty or not empty body). An agent can create arguments by using centralized rules (R_C) and centralized instances (I_C) as well as decentralized rules (R_D) and decentralized instances (I_D).

Centralized Rules (R_C) and Centralized Instances (I_C): An agent can create arguments based on only the knowledge in its ontology. In that case, the agent has all the necessary information in its own ontology so that it does need to ask its friends for information. Consider Scenario 1 Form 2. Bob's agent can provide the necessary assumption (*isOrdinary*(: `wband`, $false$)), rules (I_{B_1}, P_{B_1}) and facts (`Oktoberfest`(: `location`), *obtainedFrom*(: `wband`, : `location`)) from its own ontology to support its claim.

Centralized Rules (R_C) and Decentralized Instances (I_C): An agent may not be able to create an argument even though it has the necessary rules. In this case, the agent should find the instances to fulfill the rule. A possible solution is to ask other agents in the network for the missing information. It should be up to the agent whom to ask for that information. We enable each agent to develop its own strategy to ask for decentralized instances. For example, Alice would have rule I_{A_1} but not assumption *foundAt*(`:wband`,`:Gifty`) in its ontology. Then, she could have choose a shop owner friend (e.g. David) to ask for that information instead of choosing her colleague.

Decentralized Rules (R_C) and Centralized Instances (I_C): An agent attacks an assumption by providing rules that support the contrary of the assumption. It is possible for an agent to not have a rule to attack its opponent's assumption. In this case, the agent can ask other agents in the social network for decentralized rules. For example in Scenario 1 Form 4, Bob does not know how to support *isClosed*(`:Gifty`,true)). Then, Carol suggests rule I_{B_2}. Since Bob already has the assumption *isAccessible*(: `url`, $false$) and the fact *hasUrl*(: `Gifty`, : `url`) in its ontology, he can attack Alice's argument. Again, it is up to Bob to whom to ask for decentralized rules.

Decentralized Rules (R_C) and Decentralized Instances (I_C): An agent may not have any rules and instances in its ontology to attack its opponent's arguments. In this case, the agent first needs to obtain decentralized rules from the other agents in the social network. If it manages to obtain decentralized rules, it tries to fulfil them by providing instances. The agent may not also have assumptions or facts to support the decentralized rules. In this case, it asks other agents for decentralized instances. In Scenario 1 Form 4, `:bob` asks `:carol` for a decentralized rule and `:carol` provides I_{B_2}. If

:bob could not support I_{B_2}, it would ask other agents for instances *hasUrl*(:Gifty,:url) and *isAccessible*(:url,false).

4.2. An Algorithm for Argumentation

We propose an algorithm, which enables agents to conduct argumentation sessions. Argumentation starts with an initial case created by the requester agent. Requester agent assumes that sharing the post is okay for relevant agents. It also states that if a relevant agent finds the post harmful, its assumption is rebutted. When creating the initial case, the requester agent puts its initial assumption in A, assumption contrary mapping in C and facts about the post in F. It does not add any rules in R, since it does not need to derive new information in that point. A case status is *ongoing* as long as argumentation continues. Thus, the agent sets the case status to *ongoing* and sends the case to the relevant agents. We assume that an agent is related to a post if it is tagged or mentioned in the post. However, one may want her agent to be more sensitive about finding the relevant agents. For example, an agent may think that an object in a picture reveals its owner's presence. Thus, if there is an object in a post, then the object owner is related to that post. We enable agents to define their own mechanism to find relevant agents. Below, we describe some auxiliary functions that we use in PrepareAttack algorithm.

- initCase() creates a case with an empty set of rules R, facts F, assumptions A and contrary mappings C. *status* is null.
- updateOntology(F, o) takes the agent's ontology and the facts as input. Then, updates the agent's ontology with the facts. It is important to update the ontology with the new information, since it helps to capture inferences.
- getContrariesToAttack(A, o) takes opponent's assumption set and the agent's own ontology as input. Then, finds all of the contraries that the agent must support to attack these assumptions.
- prepareCase(R, A, F, C, *status*) takes a set of rules R, assumptions A, facts F, contrary mappings C and status as input. Then, creates a case. The *status* is set either to *ongoing* or *stop*.

- getRelatedRules(*contraryList*, *o*) takes a contrary list *contraryList* and an ontology *o* as input. Then, finds the rules that infer each contrary in the contrary list. An agent first tries to find the rules in its own ontology. If there are no such rules, it asks other agents for rules. A rule is related to a contrary if it is used to infer that contrary. Thus, a semantic rule may still be relevant to a contrary even though it does not has the contrary in its head. For example, I_{B_1} does not directly support *rejects*(`:bob`, `:pr`). However, it infers `Festival`(`:context`), which supports *rejects*(`:bob`, `:pr`) with the rule P_{B_1}. Therefore, rule I_{B_1} is related to the contrary *rejects*(`:bob`, `:pr`).
- getInstantiations(*rList*, *o*) takes rules *rList* and an ontology *o* as input. Then, tries to instantiate each rules in the *rList* with the ontology *o*. Head of a rule is true if and only if body of it is true. Therefore, agents must instantiate the body of a rule to infer the head of it. An agent first try to instantiate *rList* with its own knowledge. If it does not have enough information, it asks other agents.
- canRebutA(*iList*, *contraryList*) checks whether the instantiations in iList can support all of the contraries in *contraryList* (i.e. all of the assumptions of the opponent is rebutted by the agent's rules). An agent should be able to rebut all of the assumptions of its opponent to be able continue the argumentation. Consider Scenario 1 Form 4. If Bob would be able to ignore Alice's assumption *foundAt*(`:wband`,`:Gifty`) and provide an irrelevant argument, Alice's assumption would never be rebutted. Thus, the argumentation would not make sense any more.
- getBody(*i*) takes a rule instantiation *i* as input. Then, returns the predicates of that instantiation's body.
- getContrary(*a*) takes an assumption *a* as input and returns its contrary.

PrepareAttack algorithm takes a case *s* as input, extends it to another case s' and returns s'. At the beginning of the PrepareAttack algorithm, the agent first initializes the case s' (line 1). Then, it checks whether the previous agent set the argumentation status to *stop* (line 2). If so, the argumentation stops. Otherwise, the agent takes the rule set, the assumption set, the fact set and the contrary mapping of *s* and sets them to R, A, F and C, respectively (line 3). Then, it adds the new facts in

Require s, case received from other agent
$s' \leftarrow$ initCase()
if $s.status \neq stop$ **then**
  $R \leftarrow s.R$, $A \leftarrow s.A$, $F \leftarrow s.F$, $C \leftarrow s.C$
  $o \leftarrow$ updateOntology(F, o)
  $contraryList \leftarrow$ getContrariesToAttack(A, C)
  if $contraryList.size() = 0$ **then**
    $s' \leftarrow$ prepareCase$(R, A, F, C, stop)$, **return** s'
  else
    $rList \leftarrow$ getRelatedRules$(contraryList, o)$
    $iList \leftarrow$ getInstantiations$(rList, o)$
    if $canRebutA(iList, contraryList) = false$ **then**
      $s' \leftarrow$ prepareCase$(R, A, F, C, stop)$; **return** s'
    else
      for each i in $iList$ **do**
        $R \leftarrow R \cup \{i\}$
        for each p in getBody(i) **do**
          if $p.name \in aList$ **then**
            $A \leftarrow A \cup \{p\}$
            $p' \leftarrow$ getContrary(p)
            $C \leftarrow C \cup \{p : p'\}$
          else if $p.name \in fList$: **then**
            $F \leftarrow F \cup \{p\}$
          end if
        end for
      end for
    end if
  end if
  $s' \leftarrow$ prepareCase$(R, A, F, C, ongoing)$
else
  $s' \leftarrow s$
end ifreturn s'

Figure 4.2. PrepareAttack (s) Algorithm.

its ontology to capture further inferences (line 4). The agent finds all of the contraries that it should support and puts them into the *contraryList* (line 5). If there are no contraries that it should support (line 6), then there are no weak points that the agent can attack. Therefore, the agent prepares a case with the *stop* status and returns it back to its opponent (line 7). If there are contraries that the agent should support (line 8), then it finds all of the rules that infer the contraries in the *contraryList* (line 9). The agent has to be able to instantiate a rule to infer its head. Thus, the agent tries to instantiate all of the rules in $rList$ (line 10). Then, it checks whether it could instantiate the necessary rules that support the contraries (line 11). If it cannot support all of the contraries, the assumptions of the opponent will not be rebutted. Thus, the agent prepares a case with the *stop* status to end the argumentation (line 12). If it manages to attack all of the assumptions (line 13), then it takes each instantiation i and adds it into the rule set R (line 14-15). The agent checks each predicate p in the body of i and classifies it as an assumption, a fact or a deduction. We benefit from a predefined fact list ($fList$) and an assumption list ($aList$) to specify which predicates should be considered as a fact or an assumption by the agent. If p is an assumption (in $aList$), then it is added into the set A (line 18). Since each assumption in ABA must also have a contrary, the assumption-contrary pair (p:p') is added into the C (line 20). If p is a fact (in $fList$), then it is added into the set F (line 21-22). At line 23, the agent has a set of rules, a set of assumptions, a set of facts and the contrary mappings to attack the opponent's arguments. The agent prepares the new case s' with the modified sets (A, F, C and R), sets the status to *ongoing* and sends it to the other agent (line 26).

Notice that during the argumentation the agents exchange information by extending a case. Thus, at the end of the argumentation the case involves all of the information provided by the agents. In that point, the requester agent can create an ABA specification from the case and give it to an ABA engine. Then, it queries the ABA engine to decide if it can share the post or not (see Definition 1).

Definition 3. *Consider two cases* $s = \langle R, A, F, C, status\rangle$ *and* $s' = \langle R', A', F', C', status'\rangle$ *as* s *is the case produced by an agent, and* s' *is the any case that can be produced by that agent.* s *is a complete case iff* $R' \subseteq R$, $A' \subseteq A$, $F' \subseteq F$ *and* $C' \subseteq C$.

Our agents follow *thorough strategy-move* by default [36]. In other words, they provide all of the relevant information they know during the argumentation and could not provide an improved case. Therefore our agents provide each other *complete cases* as described in Definition 3. Following proof proves that PrepareAttack always produces complete cases.

Proof. Let s be a complete case produced by an agent and s' be a case that can be produced by that agent. Assume that s' is not complete. In other words, s includes assumptions, contraries, facts or rules that s' does not have. However, lines 6-22 of PrepareAttack algorithm makes sure that s' includes all of the facts, rules, assumptions and contraries the agent provides. The agent can provide these information from its own ontology as well as the ontologies of its friends. Therefore, our initial assumption cannot be correct. □

5. IMPLEMENTATION

We implemented PriArg, which is a system that enables semantic agents to come to an agreement with assumption-based argumentation. Our implementation is based on the implementation of Mester *et al.* [15]. We benefit from ontologies to represent an agent's knowledge base, OWL to define the concepts in the ontologies and SWRL rules to formally represent the rules of agents.

We use the ontology created by Mester *et al.* by adding new concepts. The ontology is created with Protégé [37] (see Figure 5.1). Since agents have to speak the same language to communicate, we define a base ontology with concepts, object properties and data properties. Then, we personalize it for each agent by adding new individuals, relations between these individuals and new rules. We create agent rules using SWRL (see Figure 5.2).

We implemented our work using JAVA and OWL API [38]. Each concept in the ontology is represented with a Java class. There is a hierarchical relation between the classes as post request is at the top. You can see the current object model of a post request in Figure 5.3. A $PostRequest$ includes either a $PostText$ or a $Medium$. A $Medium$ is in the type of $Picture$ or $Video$. Each $PostRequest$ is in some $Context$ such as $Work$, $Leisure$ and $Meeting$. A $Medium$ can include an $Object$ and a $Location$. $Location$ may be in the type of $City$, $Country$, $Oktoberfest$ etc. A $PostRequest$ is made by an $Agent$ to present the post to an $Audience$. This hierarchy enables agents to be more expressive when creating privacy constraints. For example, in Scenario 1 `:bob` can easily define a rule to reject a post request, if it is in the $Festival$ context. Moreover, he can state that a post request is in the $Festival$ context, when its medium includes a unique $Object$ taken from the $Okoberfest$. Any other ontology compatible with ours can be used in the system. It is also possible to use a more comprehensive ontology by adding corresponding Java classes into the system.

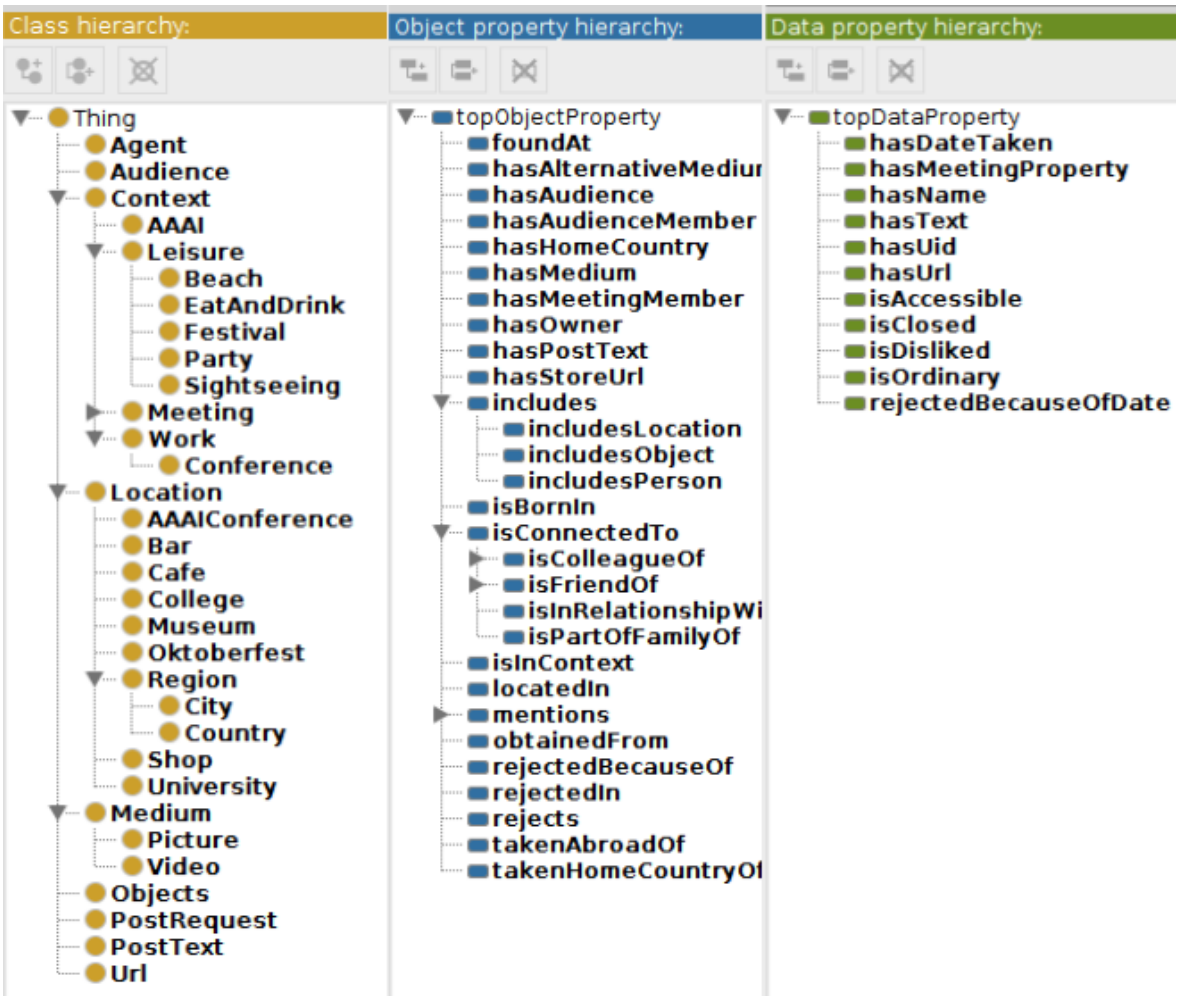


Figure 5.1. Ontology model created with Protégé. Concepts, object properties and data properties respectively.

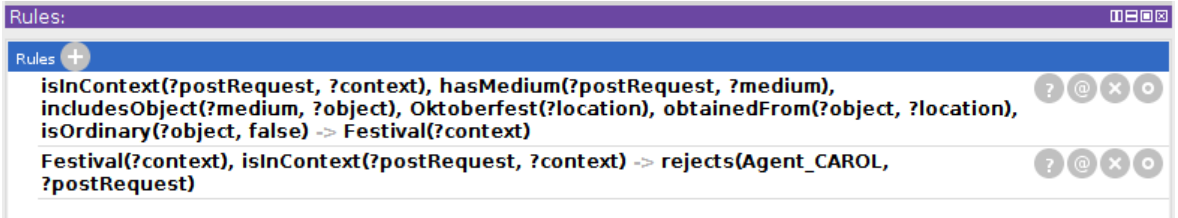


Figure 5.2. SWRL rules created with Protégé.

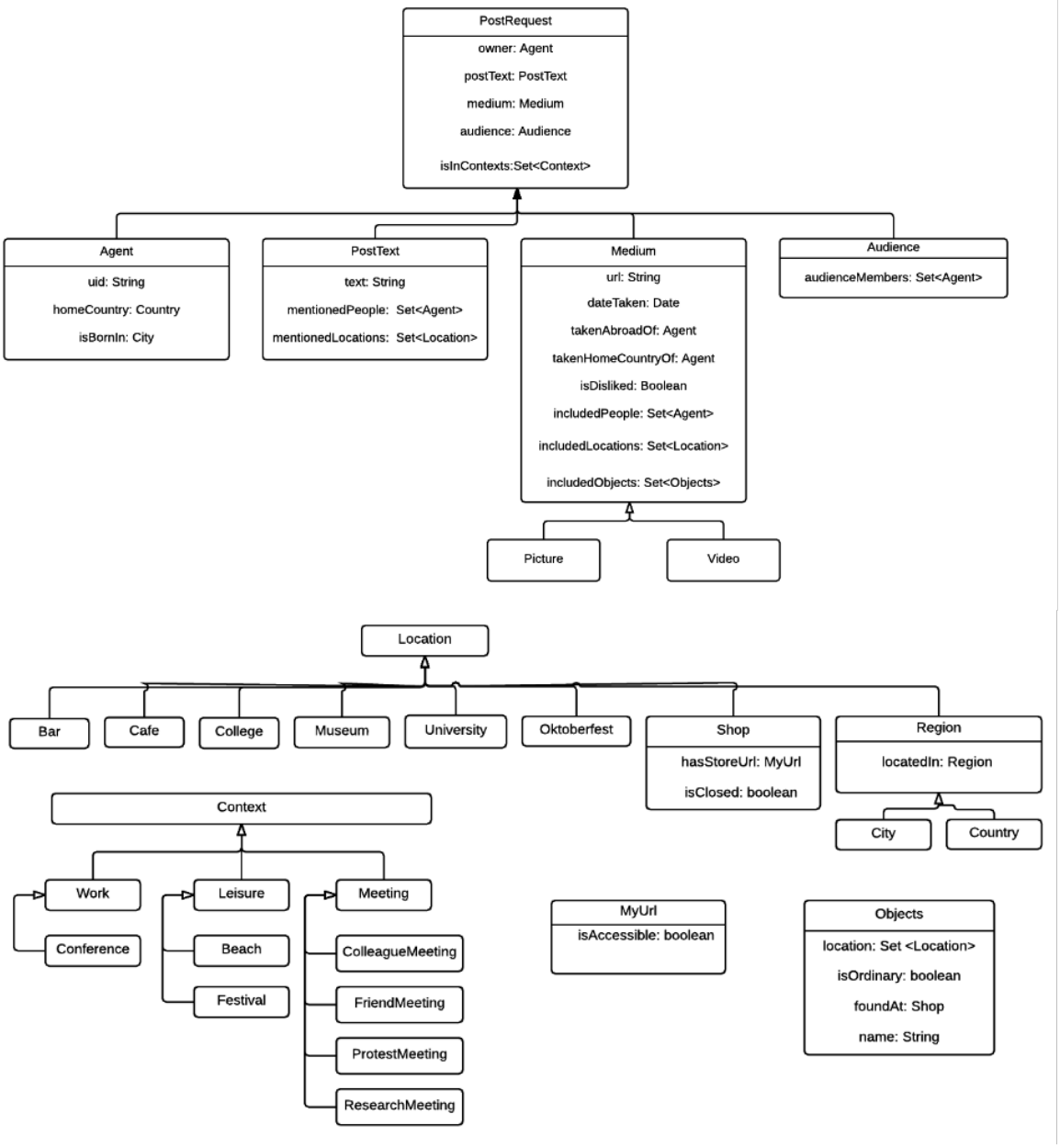


Figure 5.3. Object model of a post request.

WELCOME TO PRIARG
File Name scenario4 Submit

SCENARIO DESCRIPTION

Alice wants to share a post, which is related to Bob and Carol. However, Carol does not want any post to be shared in the festival context. Since the post includes an unique object (wristband) taken from a festival (Okteberfest), Carol considers the picture in the festival context. Upon this, Alice opposes Carol since she thinks the wristband can be found in a shop (Gifty). Finally Carol comes to conclusion that the shop is closed and the post should not be shared. Bob also has a privacy constraint to not to share any photo that is taken at abroad. Since he thinks that the picture is taken at abroad, he rejects Alice's request. On the other hand, Alice has information about the home country of Bob and knows the place that the photo is taken. As a result she comes to conclusion that the photo is taken at Bob's home country and the photo should be shared. After both argumentation sessions, Bob is convinced to share the post while Carol is not. Thus, the post is not shared.

INTERNAL REPRESENTATION OF SOLUTION FOR AGENT_ALICE & AGENT_CAROL

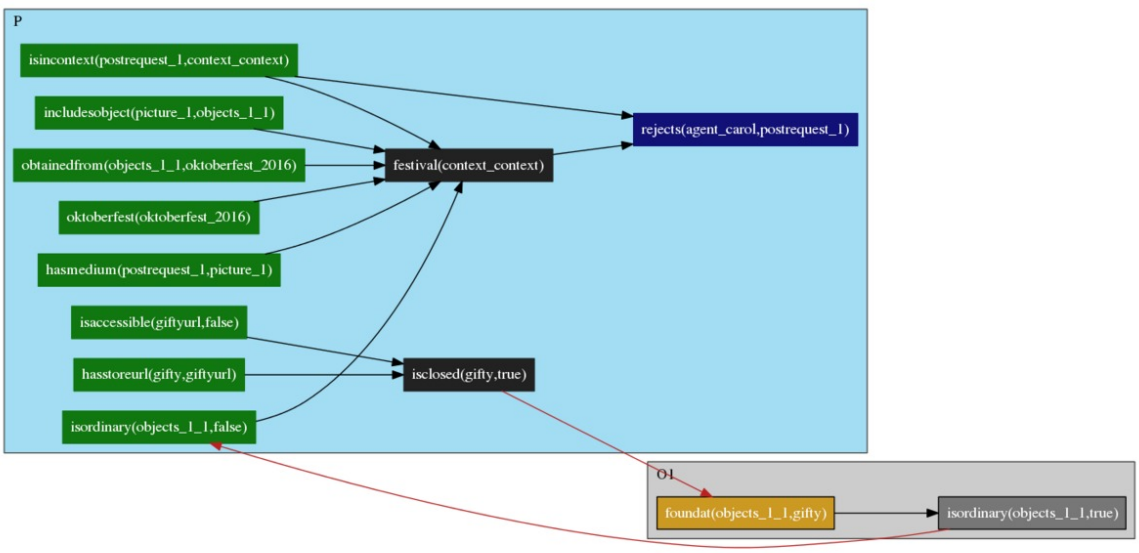
PGRAPH: [festival(context_context)-[oktoberfest(oktoberfest_2016), hasmedium(postrequest_1, picture_1), includesobject(picture_1, objects_1_1), isincontext(postrequest_1, context_context), isordinary(objects_1_1, false), obtainedfrom(objects_1_1, oktoberfest_2016)], oktoberfest(oktoberfest_2016)-[], hasmedium(postrequest_1, picture_1)-[], hasstoreurl(gifty, giftyurl)-[], includesobject(picture_1, objects_1_1)-[], isaccessible(giftyurl, false)-[], isclosed(gifty, true)-[hasstoreurl(gifty, giftyurl), isaccessible(giftyurl, false)], isincontext(postrequest_1, context_context)-[], isordinary(objects_1_1, false)-[], obtainedfrom(objects_1_1, oktoberfest_2016)-[], rejects(agent_carol, postrequest_1)-[festival(context_context), isincontext(postrequest_1, context_context)]]

DEFENCE: [isaccessible(giftyurl, false), isordinary(objects_1_1, false)]

OGRAPHS: [isordinary(objects_1_1, true)-[]-[foundat(objects_1_1, gifty), isordinary(objects_1_1, true)]-[foundat(objects_1_1, gifty)-[], isordinary(objects_1_1, true)-[foundat(objects_1_1, gifty)]]]

CULPRITS: [foundat(objects_1_1, gifty)]

POST CANNOT BE SHARED! rejects(agent_carol, postrequest_1) is justified.



INTERNAL REPRESENTATION OF SOLUTION FOR AGENT_ALICE & AGENT_BOB

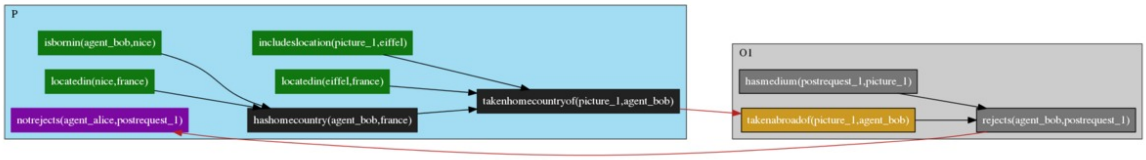
PGRAPH: [hashomecountry(agent_bob, france)-[isbornin(agent_bob, nice), locatedin(nice, france)], includeslocation(picture_1, eiffel)-[], isbornin(agent_bob, nice)-[], locatedin(eiffel, france)-[], locatedin(nice, france)-[], notrejects(agent_alice, postrequest_1)-[], takenhomecountryof(picture_1, agent_bob)-[hashomecountry(agent_bob, france), includeslocation(picture_1, eiffel), locatedin(eiffel, france)]]

DEFENCE: [notrejects(agent_alice, postrequest_1)]

OGRAPHS: [rejects(agent_bob, postrequest_1)-[]-[hasmedium(postrequest_1, picture_1), rejects(agent_bob, postrequest_1), takenabroadof(picture_1, agent_bob)]-[hasmedium(postrequest_1, picture_1)-[], rejects(agent_bob, postrequest_1)-[hasmedium(postrequest_1, picture_1), takenabroadof(picture_1, agent_bob)], takenabroadof(picture_1, agent_bob)-[]]]

CULPRITS: [takenabroadof(picture_1, agent_bob)]

POST CAN BE SHARED! notrejects(agent_alice, postrequest_1) is justified.



Not all related agents accept the request. POST CANNOT BE SHARED!

Figure 5.4. Web screenshot of PriArg.

When application first starts a Web screen with a text field and a submit button is shown to the user (see Figure 5.4). Any scenario in the static folder of the project can be run by typing the name of it in the text field. Then, PriArg returns the result of the scenario. Each scenario consists of a post request in the json format. Thus, any other scenario can be easily created by following the same format. When we run a scenario, the json format is converted to a post request object including the objects of other classes (e.g. Post request object includes a post text object) and the post request object is given into the system. We use Tomcat as our application server.

When a post request is made, the relevant agents need to check if the post violates their privacy. Thus, they put the request in their ontology and reason with their SWRL rules. OWL API enables us to perform ontological reasoning from our application. First, we convert a post request into an ontological individual. For that purpose, we use class assertion, data property assertion and object property assertion features of OWL API. Then, we use Pellet [39], a Description Logic reasoner, to reason in the ontology with the SWRL rules. OWL API cannot show the rules that are initialized with the reasoning. Thus, we implemented our own method that finds the fired rules for different instantiations. A rule is fired if its body holds. Thus, we check if each predicate in the body holds in the ontology. It is also possible for a predicate, which is in the body of a rule to be the head of another rule and that rule may not be instantiated by our system yet. For example, consider the two rules I_{B_1} and P_{B_1} of `:bob` in Scenario 1. Body of P_{B_1} includes `Festival`(?context), which is the head of I_{B_1}. Therefore, `:bob` must resolve I_{B_1} to resolve P_{B_1}. Similarly, `:bob` has to find which object is in the picture to find the place it is obtained from. Therefore, the order of rules and predicates are important. To prevent the mistakes due to the order of the rules and predicates, we define the Big-O for a rule to be instantiated in the system and try to resolve it in that times.

Agents talk by sending a case object to each other in a turn taking fashion. A case object includes assumption-contrary pairs, rules and facts to create the ABA instance of the discussion. When a requester agent first starts the argumentation, it puts an assumption in the initial case claiming that there is no need to reject the

current post request. It also puts the contrary of its assumption, which claims that the relevant agent rejects the post request. After the relevant agent evaluates the post in its ontology, it provides new rules, new assumptions and their contraries within the case object. We also enable an agent to ask another agent for decentralized information. If an agent receives decentralized information from another agent, it updates its post request object. Then, it uploads the post request in its ontology and reason again. It is necessary for agents to understand each other and express the knowledge in their ontology in ABA format. Thus, each assumption and fact is in the ontology triple format. Further, it is important to decide whether a triple that an agent provides is an assumption or a fact. Agents decide if a predicate is an assumption or a fact by looking at the predefined lists in the system. Thus, when a new triple is used in the system, it should be defined in the assumption or fact list. Similarly, contrary of each assumption with a specific predicate is predefined, since agents need to know the contrary of an assumption to rebut it (see Section 4.2).

Since in each turn agents add new information into the case, at the end of argumentation all information provided into the discussion is collected into the case object. The case object can be evaluated to find a justified conclusion about sharing the post. We use an open source Prolog program, *abagraph* [17], to evaluate the output of our assumption based argumentation. Abagraph runs in SICStus Prolog (4.2+) [40]. When argumentation ends, we create a document with the information that we get from the case object (see Figure 5.5). Then, from our Java application we call a shell script that runs *abagraph* on SICStus. We load the created document into abagraph and query whether the post can be shared or not. Result of the argumentation is returned to the Java project and presented in the web screen of the application. The scenario description, internal representation of the solution, a visualized explanation and the ultimate result of the argumentation are showed in the web screen. Abagraph needs graphviz [41] to create the visualized explanation of the solution. Internal representation and the visualized explanation is produced by abagraph when the queried argument is justified. Thus, we query both the initial assumption and its contrary in PriArg. The agent whose assumption is justified is called as proponent while the other agent is called as opponent. Internal representation of the solution consists of PGRAPH, DEFENCE,

OGRAPHS and CULPRITS. PGRAPH is the argument graph of the proponent, OPGRAPHS is the argument graph of the opponent, DEFENCE is the assumption set that supports arguments of the proponent and CULPRITS is the assumption set that supports arguments of the opponent and counter attacked by the proponent. In the case where more than one agent are related to the post, argumentation result for each proponent-opponent pair is presented in the web page. A post cannot be shared if at least one agent rejects the post request. Figure 5.4 shows an example where `:alice` wants to share a photo of both `:bob` and `:carol`. `:bob` is convinced to share the picture while `:alice` is not. As a result, the post is not shared.

```
% Initialized rules of agents

        myRule(festival(context_context), [oktoberfest(oktoberfest_2016),
        hasmedium(postrequest_1 ,picture_1), includesobject(picture_1 ,objects_1_1),
        isincontext(postrequest_1 ,context_context), obtainedfrom(objects_1_1 ,oktoberfest_2016),
        isordinary(objects_1_1 ,false)]).
        myRule(rejects(agent_bob ,postrequest_1), [festival(context_context),
         isincontext(postrequest_1 ,context_context)]).
        myRule(isordinary(objects_1_1 ,true), [foundat(objects_1_1 ,gifty)]).
        myRule(isclosed(gifty ,true), [hasstoreurl(gifty ,giftyurl), isaccessible(giftyurl ,false)]).

% Given facts by agents.

         myRule(oktoberfest(oktoberfest_2016),[]).
         myRule(hasmedium(postrequest_1 ,picture_1),[]).
         myRule(includesobject(picture_1 ,objects_1_1),[]).
         myRule(isincontext(postrequest_1 ,context_context),[]).
         myRule(obtainedfrom(objects_1_1 ,oktoberfest_2016),[]).
         myRule(isincontext(postrequest_1 ,context_context),[]).
         myRule(hasstoreurl(gifty ,giftyurl),[]).

% Assumptions

         myAsm(isaccessible(giftyurl ,false)).
         myAsm(foundat(objects_1_1 ,gifty)).
         myAsm(isordinary(objects_1_1 ,false)).
         myAsm(notrejects( agent_alice ,postrequest_1)).

% Contraries

         contrary(isaccessible(giftyurl ,false),isaccessible(giftyurl ,true)).
         contrary(foundat(objects_1_1 ,gifty),isclosed(gifty ,true)).
         contrary(isordinary(objects_1_1 ,false),isordinary(objects_1_1 ,true)).
         contrary(notrejects( agent_alice ,postrequest_1),rejects( agent_bob ,postrequest_1)).

         toBeProved([notrejects( agent_alice ,postrequest_1)]).
```

Figure 5.5. ABA file created by our system for Scenario 1.

We enabled agents to discuss about sharing a post through argumentation. We benefit from the agent communication structure developed by Mester *et al.* by updating it for argumentation. RESTful Web services and JSON format are used in this structure. JSON is used for exchanging messages between agents. Each agent must

provide the same Web service endpoint to run a specific method during the argumentation. However, implementation of those methods can vary from agent to agent. Thus, agents may follow different strategies.

6. EVALUATION

We evaluate our work by following three different approaches. First, we test PriArg against the real world scenarios. We provide a walk through for the four Forms of Scenario 1. Moreover, we provide three new real world scenarios to show the capability of PriArg. Second, we conduct two different surveys to understand whether the results of the PriArg overlap with the expectations of users. At last, we compare PriArg with other works in the literature through constraints of a privacy protection model.

6.1. Testing Real World Scenarios

6.1.1. A Walk-through of PriArg

Consider the 4 scenario forms that we provide in Chapter 2. Agents execute the scenarios as in Table 6.1.

Table 6.1. Cumulative iteration steps for Scenario 1.

		Case					**Rules**		**Instances**		
Form	**Turn**	R	A	F	C	*status*	R_C	R_D	I_C	I_D	**Shared?**
1	:alice	{}	{}	-	{}	-	-	-	-	-	✓
2	:alice	{}	$\{as_2\}$	$\cup_{i=1}^{6} r_i$	$\{c_2\}$	*ongoing*	- P_{B_1}	-	as_2, $\cup_{i=1}^{6} r_i$	-	-
	:bob	$\{I_{B_1}, P_{B_1}\}$	$A \cup \{as_3\}$	F	$C \cup \{c_3\}$	*ongoing*	I_{B_1}, P_{B_1}	-	as_3	-	-
	:alice	R	A	F	C	*stop*	-	-	-	-	✗
3	:alice	$R \cup \{I_{A_1}\}$	$A \cup \{as_1\}$	F	$C \cup \{c_1\}$	*ongoing*	I_{A_1}	-	as_1	-	-
	:bob	R	A	F	C	*stop*	-	-	-	-	✓
4	:bob	$R \cup \{I_{B_2}\}$	$A \cup \{as_4\}$	$F \cup \{r_7\}$	$C \cup \{c_4\}$	*ongoing*	- -	I_{B_2}	r_7, as_4	-	-
	:alice	R	A	F	C	*stop*	-	-	-	-	✗

In each iteration an agent receives a case, evaluates it in its ontology, extends it with the new information and sends it back. The new information is obtained either from the agent itself (centralized rules R_C, centralized instances I_C) or from the other

agents in the social network (decentralized rules R_C, decentralized instances I_C). If the agent cannot provide further information, it sets the case status to *stop* and ends the argumentation. At this point, the requester agent creates a specification from the case, which involves all of the facts F, assumptions A, contrary mappings C and rules R provided by the agents. Then, it puts the specification into *abagraph* to decide whether to share the post or not.

Scenario 1. Form 1: Alice's agent (`:alice`) wants to share the post without consulting Bob's agent (`:bob`). Thus, it does not start an argumentation. Instead, it directly shares the post.

Scenario 1. Form 2: `:alice` wants to consult `:bob` before sharing a post `:pr`. It starts an argumentation by creating a case with the assumption as_2 (not(*rejects*(`:alice`,`:pr`))). It also specifies the contrary of its assumption to show how `:bob` can dissuade it. Then, it adds the facts about the post into the case. `:alice` uses its own ontology to provide assumptions and facts. Since it adds new information and wants an answer, it sets the case status to *ongoing* and sends the case to `:bob`. `:bob` receives the case and executes it as in the PrepareAttack algorithm. First, it checks the status of the case and decides to continue the argumentation. Then, it updates its ontology with the facts that `:alice` provided in the previous iteration. Contrary of `:alice`'s assumption (*rejects*(`:bob`,`:pr`)) holds in `:bob`'s ontology. Thus, it provides the rules (I_{B_1}, P_{B_1}) and the instance as_3 (*isOrdinary*(`:wband`,false)), which leads to that contrary. `:bob` also uses the information from its own ontology. It sets the case status to *ongoing* and sends the case back to `:alice`. `:alice` evaluates the post in its ontology and cannot find any information to attack `:bob`'s argument. Hence, it sets the case status to *stop* and terminates the argumentation. As a result, the post cannot be shared.

Scenario 1. Form 3: Following Form 2, `:alice` has a rule I_{A_1} to support the contrary of as_3 (*isOrdinary*(`:wband`,true)) and the necessary instance as_1 (*foundAt*(`:wband`,`:Gifty`)) to instantiate its rule. It checks if as_1 is an assumption or a fact. It finds the predicate in *aList*. Thus, it adds as_1 into A and its contrary (*isClosed*(`:Gifty`,true)) into C. `:alice` sets the case status to *ongoing* and sends the case to `:bob`. Since `:bob` cannot

find any information to attack :alice's argument, the argumentation stops and :alice shares the post.

Scenario 1. Form 4: Following Form 3, :alice tries to support the contrary of as_1 (*isClosed*(:Gifty,true)). Since it cannot manage this with the information in its own ontology, it asks another agent for decentralized rules. :carol provides rule I_{B_2}. :bob instantiates the rule with the assumption as_4 (*isAccessible*(:url,false)) and the fact r_7 (*hasUrl*(:Gifty,:url)). :bob adds the new information into the case, sets the case status to *ongoing* and sends it to :alice. :alice evaluates the case and cannot find any information to attack :bob's assumption. As a result, :bob convinces :alice to not share the post. Table 3.1 shows the specification for Form 4.

6.1.2. Additional Scenarios

We provide three additional scenarios to show the capability of PriArg. Agents follow PrepareAttack Algorithm to conduct an argumentation. PriArg creates a specification from the rules and the instances that the agents provide (i.e., Table 6.3, Table 6.6, Table 6.9). Then the requester agent puts the specification into *abagraph*. *abagraph* finds the arguments from the specifications (i.e., Table 6.4, Table 6.7 and Table 6.10). Then, it calculates if the initial assumption of the requester agent is justified.

Scenario 2: Alice wants to share a picture of Bob in a social network. She consults Bob before sharing the picture. Bob has a privacy constraint to not share any picture that is taken abroad. Since he thinks that the picture is taken abroad, he rejects Alice's request. On the other hand, Alice knows which city Bob was born in. Therefore she has information about the home country of Bob. Moreover, she knows the place that the photo is taken. As a result, she comes to the conclusion that the photo is taken at Bob's home country. Bob cannot rebut Alice's argument.

Table 6.2 shows the SWRL rules of Alice and Bob. Alice knows that Bob was born in Nice, which is a city located in France. Alice uses I_{A_2} to infer that France is

Table 6.2. Semantic rules in Scenario 2 as SWRL rules.

I_{A_1}:	*hasHomeCountry*(?agent, ?country), *hasGeoTag*(?medium, ?location), *locatedIn*(?location, ?country) → *takenHomeCountryOf*(?medium, ?agent) A medium indicates a user's home country if it has a location, which is in her home country.
I_{A_2}:	*locatedIn*(?city, ?country), *isBornIn*(?agent, ?city) → *hasHomeCountry*(?agent, ?country) A country is a user's home country if the city she was born is located in that country.
P_{B_1}:	*hasMedium*(?postRequest, ?medium), *takenAbroadOf*(?medium, `:bob`) → *rejects*(`:bob`, ?postRequest) A medium within a post request violates Alice's privacy if the medium signals abroad as its location.

Bob's home country. The picture has a geotag indicating Eiffel Tower. Alice knows that Eiffel is located in Paris and Paris is located in France, which is the home country of Bob. Alice comes to conclusion that the picture is taken at Bob's home country by following I_{A_1}.

Table 6.3 shows the specification of Scenario 2. Notice the fact r_5. Normally there is no direct information in Alice's knowledge base about Eiffel being located in France. However, since she knows that Eiffel is located in Paris and Paris is located in France she comes to the conclusion that Eiffel is located in France. Alice does not have an additional rule to make this inference. Instead, she benefits from the power of OWL ontology. *locatedIn* property is defined as a transitive property. Therefore, if *locatedIn*(`:eiffel`,`:paris`) and *locatedIn*(`:paris`,`:france`), then *locatedIn*(`:eiffel`,`:france`).

Figure 6.1 shows the derivation trees for the arguments b_2, a_3 and a_2 respectively. The only assumption argument is b_2, since it contains the assumption as_2 (*takenAbroadOf*(`:medium`,`:bob`)) in its body. Remaining ones are the fact arguments, since they only have facts in their body.

Table 6.3. ABA specification of Scenario 2.

$\mathcal{R} = I_{A_1} \cup I_{A_2} \cup P_{B_1} \cup_{i=1}^{5} r_i$
$r_1 = \{\rightarrow \mathit{hasMedium}(\texttt{:pr},\texttt{:medium})\}$
$r_2 = \{\rightarrow \mathit{locatedIn}(\texttt{:nice},\texttt{:france})\}$
$r_3 = \{\rightarrow \mathit{isBornIn}(\texttt{:bob},\texttt{:nice})\}$
$r_4 = \{\rightarrow \mathit{hasGeoTag}(\texttt{:medium},\texttt{:eiffel})\}$
$r_5 = \{\rightarrow \mathit{locatedIn}(\texttt{:eiffel},\texttt{:france})\}$
$\mathcal{A} = \{as_1,as_2\}$
$as_1 = \text{not}(\mathit{rejects}(\texttt{:alice},\texttt{:pr}))$
$as_2 = \mathit{takenAbroadOf}(\texttt{:medium},\texttt{:bob})$
$\mathcal{C} = \{c_1,c_2\}$
$c_1 = (\text{not}(\mathit{rejects}(\texttt{:alice},\texttt{:pr}))=\mathit{rejects}(\texttt{:bob},\texttt{:pr}))$
$c_2 = (\mathit{takenAbroadOf}(\texttt{:medium},\texttt{:bob})=\mathit{takenHomeCountryOf}(\texttt{:medium},\texttt{:bob}))$

Table 6.4. Arguments derived from Scenario 2.

$f_1 : \{\} \vdash^{r_1} \mathit{hasMedium}(\texttt{:pr},\texttt{:medium})$
$f_2 : \{\} \vdash^{r_2} \mathit{locatedIn}(\texttt{:nice},\texttt{:france})$
$f_3 : \{\} \vdash^{r_3} \mathit{isBornIn}(\texttt{:bob},\texttt{:nice})$
$f_4 : \{\} \vdash^{r_4} \mathit{hasGeoTag}(\texttt{:medium},\texttt{:eiffel})$
$f_5 : \{\} \vdash^{r_5} \mathit{locatedIn}(\texttt{:eiffel},\texttt{:france})$
$a_1 : \{\text{not}(\mathit{rejects}(\texttt{:alice},\texttt{:pr}))\} \vdash \text{not}(\mathit{rejects}(\texttt{:alice},\texttt{:pr}))$
$b_1 : \{\mathit{takenAbroadOf}(\texttt{:medium},\texttt{:bob})\} \vdash \mathit{takenAbroadOf}(\texttt{:medium},\texttt{:bob})$
$b_2 : \{\mathit{takenAbroadOf}(\texttt{:medium},\texttt{:bob})\} \vdash^{P_{B_1} \cup r_1} \mathit{rejects}(\texttt{:bob},\texttt{:pr})$
$a_2 : \{\} \vdash^{I_{A_1} \cup I_{A_2} \cup_{i=2}^{5} r_i} \mathit{takenHomeCountryOf}(\texttt{:medium},\texttt{:bob})$
$a_3 : \{\} \vdash^{I_{A_2} \cup_{i=2}^{3} r_i} \mathit{hasHomeCountry}(\texttt{:bob},\texttt{:france})$

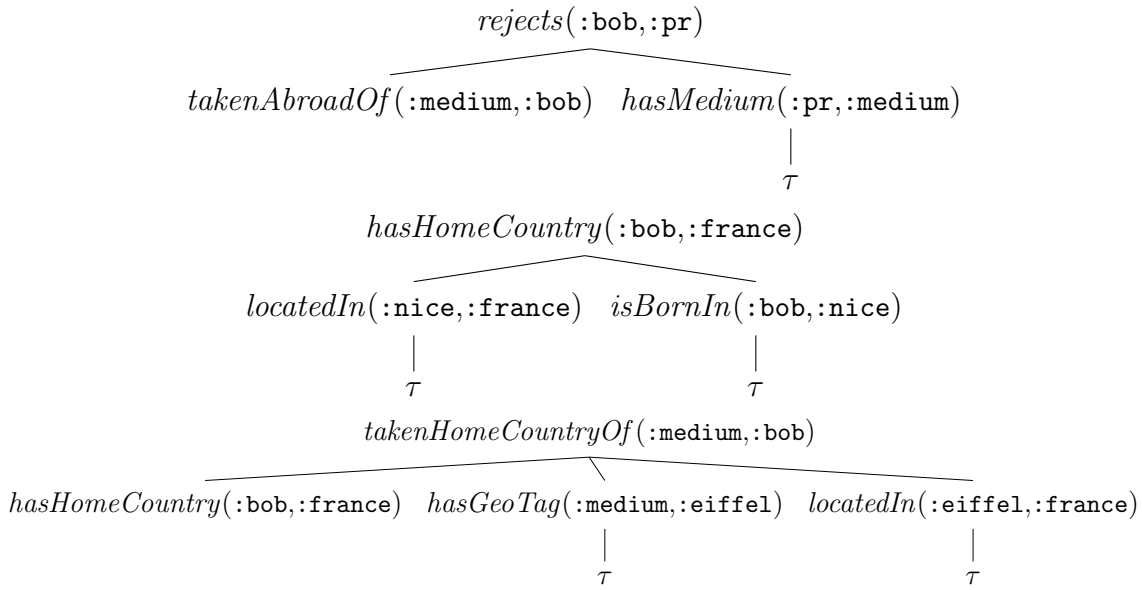


Figure 6.1. Deduction trees for the arguments b_2, a_3 and a_2 in Scenario 2 respectively.

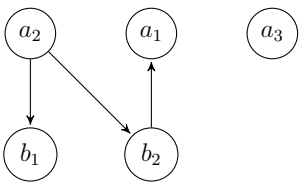


Figure 6.2. Attacks between arguments for Scenario 2.

Figure 6.2 represents the attack relations between arguments. b_2 attacks a_1, since it is the contrary of as_1 (not(*rejects*(`:alice`,`:pr`))), which supports a_1. In return, a_2 attacks b_1 and b_2, since it is the contrary of as_2 (*takenAbroadOf*(`:medium`,`:bob`)), which supports both b_1 and b_2. In this point, we can find the credulously justified admissible argument sets based on the attack relations (i.e., $\{\}$, $\{a_3\}$, $\{a_2\}$, $\{a_2, a_3\}$, $\{a_1, a_2\}$, $\{a_1, a_2, a_3\}$). `:alice` queries *abagraph* with its initial assumption as_1 (not(*rejects*(`:alice`,`:pr`))). as_1 creates the argument a_1, which is a justified argument with the argument a_2. Therefore, *abagraph* returns that the assumption is valid. Since the assumption claims that the request should not be rejected, the post is shared.

Scenario 3: Bob wants to share a picture of Alice in the social network. Alice does not want to share any picture in the leisure context and she thinks that the picture is in the leisure context. However, Bob believes that it is taken in a conference location at a conference date. Thus, he claims that the picture is in the conference context. Alice opposes Bob by showing the geotag of the picture.

Table 6.5. Semantic rules in Scenario 3 as SWRL rules.

I_{B_1}:	*isInContext*(?postRequest, ?context), *hasMedium*(?postRequest, ?medium), *includesLocation*(?medium, Antalya), `City`(Antalya), *hasDateTaken*(?medium, ?date), `dateTime`(date), *equal*(?date, "2016-03-01T00:00:00Z") → `Conference`(?context) A post is in the conference context, if its medium is taken at 3rd of January in Ankara.
I_{A_1}:	*differFrom*(?location1, ?location2), *hasGeoTag*(?medium, ?location1) → *notIncludeLocation*(?medium, ?location2) A medium does not include a location, if its geotag shows another location.
P_{A_1}:	`Leisure`(?context), *isInContext*(?postRequest, ?context) → *rejects*(`:alice`, ?postRequest) Alice rejects a post request, if it is in the leisure context.

Table 6.5 shows the SWRL rules of Alice and Bob. Alice thinks that the post is in the *Leisure* context. He rejects the request by following P_{A_1}. On the other hand, Bob knows the date the picture is taken and thinks it is taken in Antalya. Following

I_{B_1}, he comes to conclusion that the post is in the $Conference$ context. Alice knows the geotag of the picture, which is not Antalya. She uses I_{A_1} to oppose Bob's claim. Table 6.6 represents the specification of Scenario 3.

Table 6.6. ABA specification of Scenario 3.

$\mathcal{R} = I_{A_1} \cup I_{B_1} \cup P_{A_1} \cup_{i=1}^{8} r_i$
$r_1 = \{\rightarrow isInContext(\texttt{:pr},\texttt{:context})\}$
$r_2 = \{\rightarrow hasGeoTag(\texttt{:medium},\texttt{:izmir})\}$
$r_3 = \{\rightarrow differFrom(\texttt{:izmir},\texttt{:antalya})\}$
$r_4 = \{\rightarrow hasMedium(\texttt{:pr},\texttt{:medium})\}$
$r_5 = \{\rightarrow \texttt{City}(\texttt{:antalya})\}$
$r_6 = \{\rightarrow hasDateTaken(\texttt{:medium},\texttt{:date})\}$
$r_7 = \{\rightarrow \texttt{dateTime}(\texttt{:date})\}$
$r_8 = \{\rightarrow equal(\texttt{:date},\text{“2016-03-01T00:00:00Z”})\}$
$\mathcal{A} = \{as_1,as_2,as_3\}$
$as_1 = \text{not}(rejects(\texttt{:bob},\texttt{:pr}))$
$as_2 = \texttt{Leisure}(\texttt{:context})\}$
$as_3 = includesLocation(\texttt{:medium},\texttt{:antalya})$
$\mathcal{C} = \{c_1,c_2,c_3\}$
$c_1 = (\text{not}(rejects(\texttt{:bob},\texttt{:pr}))=rejects(\texttt{:alice},\texttt{:pr}))$
$c_2 = (\texttt{Leisure}(\texttt{:context})=\texttt{Conference}(\texttt{:context}))$
$c_3 = (includesLocation(\texttt{:medium},\texttt{:antalya})=notIncludeLocation(\texttt{:medium},\texttt{:antalya}))$

Figure 6.3 shows the derivation trees for arguments a_2, b_3 and a_3 respectively. a_2 and b_3 are assumption arguments, since they contains assumptions in their body. a_3 is a fact argument, since it has only facts ($hasGeoTag(\texttt{:medium},\texttt{:izmir})$, $differFrom(\texttt{:izmir},\texttt{:antalya})$) in its body.

Figure 6.4 represents the attack relations between arguments. a_2 attacks b_1, since it is the contrary of as_1 ($\text{not}(rejects(\texttt{:bob},\texttt{:pr}))$), which supports b_1. Then, b_3 attacks a_1 and a_2, since it is the contrary of as_2 ($\texttt{Leisure}(\texttt{:context})$), which supports both a_1 and a_2. At last, a_3 attacks b_3, since it is the contrary of as_3 ($includesLo$-

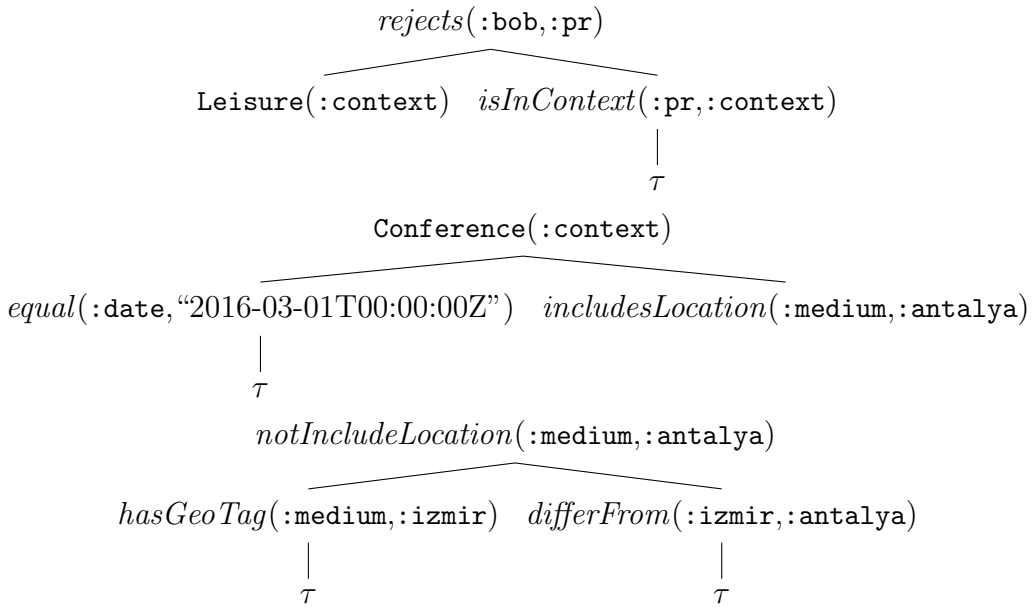


Figure 6.3. Deduction trees for the arguments a_2, b_3 and a_3 in Scenario 3.

Table 6.7. Arguments derived from Scenario 3.

$f_1 : \{\} \vdash^{r_1} \mathit{isInContext}(\texttt{:pr},\texttt{:context})$
$f_2 : \{\} \vdash^{r_2} \mathit{hasGeoTag}(\texttt{:medium},\texttt{:antalya})$
$f_3 : \{\} \vdash^{r_3} \mathit{differFrom}(\texttt{:izmir},\texttt{:antalya})$
$f_4 : \{\} \vdash^{r_4} \mathit{hasMedium}(\texttt{:pr},\texttt{:medium})$
$f_5 : \{\} \vdash^{r_5} \texttt{City}(\texttt{:antalya})$
$f_6 : \{\} \vdash^{r_6} \mathit{hasDateTaken}(\texttt{:medium},\texttt{:date})$
$f_7 : \{\} \vdash^{r_7} \texttt{dateTime}(\texttt{:date})$
$f_8 : \{\} \vdash^{r_8} \mathit{equal}(\texttt{:date},\text{“2016-03-01T00:00:00Z”})$
$b_1 : \{\text{not}(\mathit{rejects}(\texttt{:bob},\texttt{:pr}))\} \vdash \text{not}(\mathit{rejects}(\texttt{:bob},\texttt{:pr}))$
$a_1 : \{\texttt{Leisure}(\texttt{:context})\} \vdash \texttt{Leisure}(\texttt{:context})\}$
$b_2 : \{\mathit{includesLocation}(\texttt{:medium},\texttt{:antalya})\} \vdash \mathit{includesLocation}(\texttt{:medium},\texttt{:antalya})$
$a_2 : \{\texttt{Leisure}(\texttt{:context})\} \vdash^{P_{A_1} \cup r_1} \mathit{rejects}(\texttt{:alice},\ \texttt{:pr})$
$a_3 : \{\} \vdash^{I_{A_1} \cup_{i=2}^{3} r_i} \texttt{notIncludeLocation}(\texttt{:medium},\ \texttt{:antalya})$
$b_3 : \{\mathit{includesLocation}(\texttt{:medium},\texttt{:antalya})\} \vdash^{I_{B_1} \cup_{i=4}^{8} r_i \cup r_1} \texttt{Conference}(\texttt{:context})$

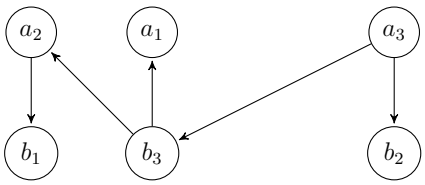


Figure 6.4. Attacks between arguments for Scenario 3.

cation(`:medium`,`:antalya`)), which supports b_3. The calculated credulously justified admissible sets of this attack relations are $\{\}$, $\{a_3\}$, $\{a_2, a_3\}$, $\{a_1, a_3\}$, $\{a_1, a_2, a_3\}$. When `:bob` queries *abagraph* with its initial assumption as_1 (not(*rejects*(`:bob`,`:pr`))), *abagraph* returns that the assumption is not valid. Therefore, the post is not shared.

Scenario 4: Bob wants to share a picture of Carol. He consults Carol before sharing it. Carol does not want any picture of her in the *Politics* context to be shared in the social network. Since she believes that the picture is in the *Politics* context, she rejects Bob's request. However, Bob thinks that the picture is unrelated with the politic events. Thus, it is in the *Unpolitic* context. Carol proves that a politic event time and location matches with the time and location that the picture indicates.

Table 6.8. Semantic rules in Scenario 4 as SWRL rules.

I_{B_1}:	*isInContext*(?postRequest, ?context), *hasMedium*(?postRequest, ?medium), `PoliticsEvent`(`:p2013`), *unrelatedTo*(?medium, `:p2013`) → `Unpolitic`(?context) A post is in Unpolitic context, if it is not related to the politic event `:p2013`
I_{C_1}:	*hasDateTaken*(?medium, ?date1), *hasEventDate*(?event, ?date2), *equal*(?date1, ?date2), `dateTime`(?date1), `dateTime`(?date2), *hasGeoTag*(?medium, ?location1), *hasEventLocation*(?event, ?location2), *SameAs*(?location1, ?location2) → *relatedTo*(?medium, ?event) A medium is related to an event, if they indicate the same date and location
P_{C_1}:	`Politics`(?context), *isInContext*(?postRequest, ?context) → *rejects*(`:carol`, ?postRequest) Carol rejects a post request, if the post is in the *Politics* context

Table 6.9. ABA specification of Scenario 4.

$\mathcal{R} = I_{B_1} \cup I_{C_1} \cup P_{C_1} \cup_{i=1}^{11} r_i$

$r_1 = \{\rightarrow \mathit{isInContext}(\texttt{:pr},\texttt{:c})\}$

$r_2 = \{\rightarrow \mathit{hasMedium}(\texttt{:pr},\texttt{:m})\}$

$r_3 = \{\rightarrow \texttt{PoliticsEvent}(\texttt{:p2013})\}$

$r_4 = \{\rightarrow \mathit{hasDateTaken}(\texttt{:m},\texttt{:date1})\}$

$r_5 = \{\rightarrow \mathit{hasEventDate}(\texttt{:p2013},\texttt{:date2})\}$

$r_6 = \{\rightarrow \mathit{equal}(\texttt{:date1},\texttt{:date2})\}$

$r_7 = \{\rightarrow \texttt{dateTime}(\texttt{:date1})\}$

$r_8 = \{\rightarrow \texttt{dateTime}(\texttt{:date2})\}$

$r_9 = \{\rightarrow \mathit{hasGeoTag}(\texttt{:m},\texttt{:taksim})\}$

$r_{10} = \{\rightarrow \mathit{hasEventLocation}(\texttt{:p2013},\texttt{:taksim})\}$

$r_{11} = \{\rightarrow \mathit{SameAs}(\texttt{:taksim},\texttt{:taksim})\}$

$\mathcal{A} = \{as_1,as_2,as_3\}$

$as_1 = \text{not}(\mathit{rejects}(\texttt{:bob},\texttt{:pr}))$

$as_2 = \texttt{Politics}(\texttt{:c})$

$as_3 = \mathit{unrelatedTo}(\texttt{:m},\texttt{:p2013})$

$\mathcal{C} = \{c_1,c_2,c_3\}$

$c_1 = (\text{not}(\mathit{rejects}(\texttt{:bob},\texttt{:pr}))=\mathit{rejects}(\texttt{:carol},\texttt{:pr}))$

$c_2 = (\texttt{Politics}(\texttt{:c})=\texttt{Unpolitic}(\texttt{:c}))$

$c_3 = (\mathit{unrelatedTo}(\texttt{:m},\texttt{:p2013})=\mathit{relatedTo}(\texttt{:m},\texttt{:p2013}))$

Table 6.8 shows the SWRL rules of Bob and Carol. Carol thinks that the post is in the *Politics* context. She rejects the request by following P_{C_1}. Bob is aware of only one politic event, which is `:p2013`. He thinks that the picture is unrelated with the `:p2013`. By following I_{B_1}, he claims that the post is in the *Unpolitic* context. Carol knows that the date and the location of the medium and `:p2013` matches. Thus, she follows I_{C_1} and comes to the conclusion that the medium is related to a politic event.

Table 6.9 represents the specification of Scenario 4. In Bob's ontology, `:p2013` is actually defined as an instance of ProtestEvent content. ProtestEvent content is a subclass of the PoliticsEvent content. Bob does not have the direct information about `:p2013` being a political event or he does not have an additional rule to infer this. Instead, he benefits from the ontological structure and infers that the `:p2013` is a political event.

Table 6.10. Arguments derived from Scenario 4.

$f_1 : \{\} \vdash^{r_1} \mathit{isInContext}(\texttt{:pr},\texttt{:c})$
$f_2 : \{\} \vdash^{r_2} \mathit{hasMedium}(\texttt{:pr},\texttt{:m})$
$f_3 : \{\} \vdash^{r_3} \texttt{PoliticsEvent}(\texttt{:p2013})$
$f_4 : \{\} \vdash^{r_4} \mathit{hasDateTaken}(\texttt{:m},\texttt{:date1})$
$f_5 : \{\} \vdash^{r_5} \mathit{hasEventDate}(\texttt{:p2013},\texttt{:date2})$
$f_6 : \{\} \vdash^{r_6} \mathit{equal}(\texttt{:date1},\texttt{:date2})$
$f_7 : \{\} \vdash^{r_7} \texttt{dateTime}(\texttt{:date1})$
$f_8 : \{\} \vdash^{r_8} \texttt{dateTime}(\texttt{:date2})$
$f_9 : \{\} \vdash^{r_9} \mathit{hasGeoTag}(\texttt{:m},\texttt{:taksim})$
$f_{10} : \{\} \vdash^{r_{10}} \mathit{hasEventLocation}(\texttt{:p2013}, \texttt{:taksim})$
$f_{11} : \{\} \vdash^{r_{11}} \mathit{SameAs}(\texttt{:taksim},\texttt{:taksim})$
$b_1 : \{\text{not}(\mathit{rejects}(\texttt{:bob},\texttt{:pr}))\} \vdash \text{not}(\mathit{rejects}(\texttt{:bob},\texttt{:pr}))$
$c_1 : \{\texttt{Politics}(\texttt{:c})\} \vdash \texttt{Politics}(\texttt{:c})$
$b_2 : \{\mathit{unrelatedTo}(\texttt{:m},\texttt{:p2013})\} \vdash \mathit{unrelatedTo}(\texttt{:m},\texttt{:p2013})$
$c_2 : \{\texttt{Politics}(\texttt{:c})\} \vdash^{P_{C_1} \cup r_1} \mathit{rejects}(\texttt{:bob},\texttt{:pr})$
$b_3 : \{\mathit{unrelatedTo}(\texttt{:m},\texttt{:p2013})\} \vdash^{I_{B_1} \cup_{i=1}^{3} r_i} \texttt{Unpolitic}(\texttt{:c})$
$c_3 : \{\} \vdash^{I_{C_1} \cup_{i=4}^{11} r_i} \mathit{relatedTo}(\texttt{:m},\texttt{:p2013})$

Figure 6.5 shows the derivation trees for arguments c_2 and b_3 respectively. Both of them are assumption arguments, since c_2 contains as_2 (`Politics(:c)`) and b_3 contains as_3 (*unrelatedTo*(`:m,:p2013`)) in its body. For clarity, we do not present the derivation three of c_3. It is a fact argument, since it only contains the facts in its body.

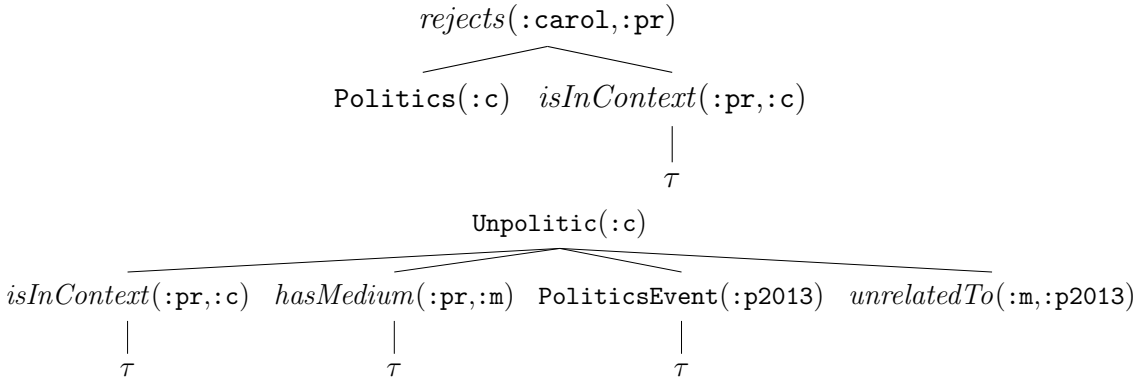


Figure 6.5. Deduction trees for the arguments c_2, and b_3 in Scenario 4.

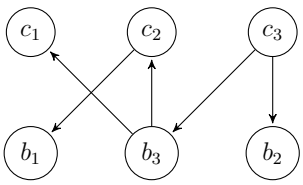


Figure 6.6. Attacks between arguments for Scenario 4.

Figure 6.6 represents the attack relations between arguments. c_2 attacks b_1, since it is the contrary of as_1 (not(*rejects*(`:bob,:pr`))), which supports b_1. Then, b_3 attacks c_1 and c_2, since it is the contrary of as_2 (`Protest(:c)`), which supports both a_1 and a_2. At last, c_3 attacks b_3 and b_2, since it is the contrary of as_3 (*unrelatedTo*(`:m,:p2013`)), which supports b_3. The credulously justified admissible sets for this argumentation are $\{\}$, $\{c_3\}$, $\{c_2, c_3\}$, $\{c_1, c_3\}$ and $\{c_1, c_2, c_3\}$. `:bob` queries *abagraph* with its initial assumption as_1 (not(*rejects*(`:bob,:pr`))) and *abagraph* returns that the assumption is not valid. Therefore, the post is not shared.

6.2. User Expectations and PriArg

We implemented PriArg to help users to manage their privacy in online social networks (see Chapter 5). To make this possible, its results should be overlapped with the expectations of users. We have conducted a personal interview and a survey to

evaluate the output of PriArg. We first asked subjects for their personal information such as their age, gender, frequency to use a social network and if they have privacy concerns in online social networks. Then, we provided them Form 1, Form 2 and Form 3 of Scenario 1, which are described in Chapter 2. We omitted the Form 1 as it does not involve argumentation and cannot be compared with PriArg. We asked subjects what do they think about sharing the post. Note that users do not consider themselves as Alice or Bob, but answer the questions objectively. At the end of the surveys, we compared the user results with PriArg.

6.2.1. Personal Interviews

In the personal interviews, we worked with 36 subjects (9 females and 27 males) from the Bogazici University. 33 of these subjects (91.6%) are aged between 18 and 35 while remaining 3 subjects are aged between 35 and 55. 30 of the subjects (83.3%) use the social networks at least once in a day and are familiar with the concepts. 28 of the subjects (77.8%) express that they have privacy concerns in online social networks. We have conducted the survey in person. Thus, participants had chance to ask questions whenever they felt necessary.

Table 6.11. Personal Interview and Online Survey results.

Form	Personal Interviews (36 participants)		Online Survey (68 participants)	
	Share	Not Share	Share	Not Share
2	5.55%	**94.44%**	7.35%	**92.65%**
3	**52.77%**	47.22%	20.59%	**79.41%**
4	2.77%	**97.22%**	7.35%	**92.65%**

In the second part of the survey, we explain the subjects the privacy constraints and the knowledge of both Alice and Bob. We use either “think” or “know” while giving the new information. In that way, we emphasize the difference between an assumption and a fact. After explaining a scenario, we ask participants to decide whether to share the post or not. During the experiment, participants were warned several times to only use the provided information in the scenarios. However, we saw a great tendency for subjects to come to the conclusions based on unavailable

information. These unavailable information were mostly their hypotheses about the scenarios, own experiences in the social networks or personal privacy rules. We also ask them follow-up questions to understand the reasons behind their decision. Thus, we were able to warn them to only use the provided information. Table 6.11 shows the personal interview results.

For Scenario 1 Form 2, 34 subjects (94.44%) claimed that the picture should not be shared considering Bob's rules and information. One subject used her own inference rule and claimed that a wristband is not a strong evidence to relate the picture with a festival. This exemplifies how users can include their own ideas instead of following the presented information. Another subject emphasized that Bob does not know that the wristband is unique. Instead, he thinks so. Thus his argument is not strong enough and the picture can be shared. Form 3 provided really interesting and valuable results. When subjects asked for a decision, 8 of them focused on the difference between the words "know" and "think". They told that if Alice's information is not absolute, then her argument is not strong even though she can rebut Bob's argument. When they were asked whether their thoughts would change if Alice's information would have been absolute, they said yes. On the other hand, 28 of people did not care the distinction between an assumption and a fact. This is an important result, since it shows the attitude of people regarding privacy and the assumption based argumentation. In real life, we do not also have all of the facts to make a decision. Instead, we come to conclusions based on our assumptions and absolute knowledge (i.e., facts). We make these assumptions based on our experience and knowledge about the world. Hence, even though assumptions can be rebutted, they differ from a random proposition. 19 (52.77%) of the remaining 28 subjects claimed that the picture can be shared with the new information while 8 of them claimed it should not be shared and one of them claimed it cannot be decided. Note that none of these eight subjects did strictly follow Alice's and Bob's rules to come to this decision. They made their own inferences and assumptions based on their experience. They claimed that the ordinariness of the wristband is not important, Bob's friends may have been in this concert etc. The same problem occurred with the subject who could not decide. In Form 4, 35 subjects (97.22%) claimed that the picture should not be shared. Only one subject claimed the

opposite. When she asked for the reason, she claimed Bob's rule is not strong enough to relate the picture with a festival.

6.2.2. Online Survey

We have created an online survey to reach more participants with different backgrounds. The online survey is created using QuestionPro [42] and stayed online for three days. We announced the survey on Facebook. In the first part of the survey, we obtained the personal information of the participants as in the personal interview. According to these information, 50 of the 68 participants are females while 18 of them are males. 6 of them (8.82%) are aged between 18 and 25, 53 of them (77.94%) has an age from 25 to 45 and 9 of them (13.23%) are older than 45. 64 of the participants (91.18%) use the online social networks at least once in a day. Thus, they are familiar with the concept. 62 of the participants (91.18%) express that they have privacy concerns in social networking sites.

Table 6.12. Dialogue between Alice and Bob for Form 2, Form 3, and Form 4 of Scenario 1.

A: I want to share this picture. Would it be ok for you? **B:** There is a wristband, which was given at Oktoberfest. I do not think that this wristband would be found somewhere else. I do not want my festival pictures to be shared.
A: This wristband might not be given at the Oktoberfest. One might find such a wristband in *Gifty*. In another words, one might not be able to understand that you are in the Oktoberfest.
B: I am not able to access the web site of *Gifty*. I think the shop is closed. So, I do not think that the wristband would be purchased from there.

In the second part of the survey, we provide participants a dialogue between Alice and Bob for each form of Scenario 1. Then, we ask them to decide whether to share the post or not. Again, participants do not consider themselves as neither Alice nor Bob but answer the questions objectively. Table 6.12 shows the dialogues for each form.

The first line of the scenario shows the dialogue between Alice and Bob for Scenario 1 Form 2. First and the second line together shows the dialogue for Scenario 1 Form 3. At last, the complete dialogue shows the Scenario 1 Form 4.

Table 6.11 shows the results for online survey. In Form 2 and Form 4, participants show strong consensus to not to share the post. 63 of them (92.65%) decide to not to share it in both forms. Form 3 shows interesting results. 54 of the participants (79.41%) still do not want the picture to be shared even though Alice could rebut Bob's argument.

6.2.3. Comparison of User Results with PriArg

Both the personal interview and the survey results shows us how user attitude can change with the new information. In PriArg, we also enable agents to change their idea with the new information. This is a valuable result, since it shows us the importance of a fine grained discussion. We believe Assumption-based Argumentation is a good fit to mimic this behaviour.

Our second observation is that the user results are mostly in line with the PriArg results. In Form 2 and Form 4, both the personal interview and the online survey results have a consensus on not to share the picture. PriArg also gives the same result, since Bob can provide arguments to convince Alice. However, there is a contradiction between the personal interviews and the online survey for Form 3. Personal interview results show that the picture can be shared while online survey results tell otherwise. Remember that during the personal interviews, even though participants are warned several times, they insist to use unavailable information. We believe that we face the same situation in online surveys. Participants have a tendency to not share the picture to make sure that nobody gets harmed. Thus, they put more restrictions to the post than Bob does. However, when a participant accepts Bob's constraints as they are and only follows them, she comes to conclusion that Bob's claim is not valid any more. Therefore the post can be shared. PriArg returns the same result.

6.2.4. Qualitative Evaluation

There are different approaches in the literature to protect the privacy of users in online social networks. However, there are no public data sets or reported results for us to compare PriArg with these approaches. Hence, we define the necessary qualifications for a privacy protection mechanism. Then, we compare PriArg with other approaches through these qualifications. We focus on three approaches from the literature as we find them the most related ones to PriArg.

PriNego is an agent-based negotiation system where users are represented with agents [15]. When an agent wants to share a post, it makes a post request to the relevant agents. Relevant agents either accept the request or reject it by providing reasons. If a post is rejected, the post owner tries to modify the post according to the rejection reasons or provide an alternative medium. For Scenario 1, when `:bob` tells `:alice` to not the share the post, `:alice` would immediately accept it or propose an alternative medium to share.

CoPE is a privacy management system, which enables all relevant users to put restrictions on a post [4]. It is implemented as a Facebook application. Each relevant user is identified as a co-owner. Co-owners put restrictions for each post and result is found based on a voting mechanism. For Scenario 1, Bob and Alice would have resolve the conflict manually. Bob would choose the co-owners and Alice would choose her friends as the audience of the post. Then, the tie between the votes would be broken in the favour of the user who originally start the discussion (i.e., Alice).

FaceBlock protects the privacy of users when their picture is taken by a Google Glass [43]. It uses SWRL rules to define the privacy policies of users. If a Google Glass is detected in the environment, FaceBlock checks whether any of the privacy rules are fired. If so, it sends a disallow message to the Google Glass. Then, the Glass obscures the face of the user in all of its pictures. For Scenario 1, `:bob` would send `:alice` a disallow message. Then, `:alice` would obscure the face of `:bob`.

Table 6.13. Comparison of privacy criteria.

	PriArg	CoPE	PriNego	FaceBlock
Automation	✓	✗	✓	✓
Concealment	✗	✓	✓	✓
Persuasion	✓	✗	✗	✗
External consultation	✓	✗	✗	✗

Automation expresses the ability of an application to run without human intervention. We believe that automation is an important constraint for privacy, since the number of contents to deal with can be extremely high. In our application, users are represented via agents that act on behalf of them. Thus, we meet the automation constraint. PriNego is also an automated system, since it enables agents to act on behalf of their users. FaceBlock uses its SWRL rules and a reasoner to automatically detect a fired privacy rule. It prevents the privacy violation by sending a message to Google Glass. However in CoPE, users themselves evaluate each related post in the social network. Hence, there is no automatic mechanism in the CoPE to protect the privacy of users.

Concealment refers to the ability of the system to keep the privacy constraints of users private. Hence, a user cannot be aware of the privacy constraints of another user. In PriArg, users have to provide their SWRL rules to each other to express their argument. Consider Form 3 of Scenario 1, `:alice` can attack `:bob`'s argument, since `:bob` provided its rules and explained what are the reasons behind its argument. `:alice` would have not know how to convince `:bob`, if `:bob` would not provide its reasons. In PriNego, agents do not challenge each other's claims. Thus, they do not need to know the structure of a claim. Agents do not reveal their privacy constraints but provide reasons behind their decision. A specific audience, date or location can be the rejection reason. When FaceBlock detects a Google Glass, it reasons in its ontology. If a user has a concern, FaceBlock sends a simple allow or disallow message instead of sending the complete rules. In CoPE, users define whom to show the content directly instead of providing any reasons. Therefore, privacy constraints of users are not revealed.

Persuasion refers to whether an agent can change another agent's decision in the system. An agent's decision, which is taken in the lack of knowledge or incorrect information can change when rebutting information arrives. Thus, an agent can persuade another agent by questioning and attacking its claims. We believe that persuasion is an important constraint, since it helps agents to come to an agreement. In PriArg, agents provide each other counter arguments with new information. Hence, they are able to persuade each other. In PriNego, requesting agent does not try to persuade the other agents that its claim is true. Instead, it modifies the post, proposes an alternative one or shares it anyway. FaceBlock or CoPE do not have a persuasion mechanism. They simply follow the constraints of relevant users.

External Consultation expresses the ability of an agent to obtain information from outside. Users need information to support their claims and to rebut the claims of other users. Therefore, it is useful to be able to obtain external information when necessary. PriArg enables agents to obtain external rules and instances. There is no such system in PriNego, CoPE and FaceBlock.

7. DISCUSSION

We propose an approach to protect privacy of users in OSNs. Each user is represented by an agent, which is fully aware of its user's privacy constraints and knowledge about the social network through its ontology. When a post wants to be shared, an argumentation process starts between agents to decide whether the post can be shared or not. If sharing the post does not violate any user's privacy, the post is shared. Otherwise, there occurs a conflict between agents and each agent tries to provide arguments to persuade the other agent that its claim is true. PriArg also enables agents to ask for external knowledge when they do not have enough information in their ontology. Thus, agents can benefit from other agents' information to protect their privacy. When argumentation ends, we calculate the justified argument sets to decide whether the post can be shared or not.

Detecting conflicts between different parties, resolving them and coming to an agreement is an important concern in various domains. In the literature, there are different methods to resolve conflicts. Wishart *et al.* propose PRİMMA-Viewer, a collaborative approach for users to create a privacy policy in OSNs [3]. A user who uploads a post into a social network defined as the owner of the post. Owner specifies the initial privacy policy of the post and actions that the audience of the post can perform. The owner can also nominate another user as a co-owner. Owner and co-owners collaboratively create the privacy policy. A privacy policy consists of a resource, possible actions that can be performed with the resource, strong and weak conditions proposed by users. Strong conditions cannot be touched while weak conditions can be deleted by other users. If a conflict arises because of a strong condition, either co-owners respect the strong condition or alter the post as the condition owner is not related to the post. Each user is able to create new weak or strong conditions. When a user makes a request, Facebook directs the request to PRİMMA-Viewer. Then, Policy Decision Point (PDP) in the PRİMMA-Viewer evaluates the request by using the IRIS reasoner. PDP sends its decision to the Policy Enforcement Point (PEP) and Facebook displays views that is decided by the PRİMMA-Viewer. PRİMMA-Viewer enables all

related users to manage their content. Even though it enables users to resolve conflicts, it cannot be done automatically which is a burden for users. In addition, authors do not mention a system that automatically proposes policies. Thus, users have to create a policy from the scratch for each post. A co-owner can only be assigned by the owner of the post. It is not possible for a user to request to be a co-owner. Another issue is that a privacy policy of a user become known by all of the users.

Another method to resolve conflicts between different parties is majority voting. Squicciarini *et al.* [4] developed a collaborative privacy management system (CoPE) for Facebook. There are two stakeholders in CoPE as content-owner and co-owner. A content-owner is the user who uploads the post into the social network. A co-owner is the user whose information is revealed by the post of the content-owner. As opposed to the most current OSNs, CoPE enables co-owners to control the privacy settings of the content. Authors call the shared content as privacy-content and audience of the post as content-viewers. The requirements for a collaborative privacy management system that CoPE provides are following: (1) A content-owner can assign other users as co-owners; (2) a co-owner gets warned for a related privacy-content; (3) a privacy-content related user should be able to make a request to be a co-owner; (4) A co-owner should be able to manage the privacy settings of a privacy-content. CoPE has two main contributions. First, it prevents privacy leakages by following privacy settings. Second, it provides all stakeholders to collaboratively manage the settings by using a simple voting system. Each stakeholder can set the audience as some-friends, public or co-owner only. The privacy preference for the content is set according to the majority of votes. If the majority voted for co-owner only, no content-viewers apart from co-owners can see the content. Otherwise, the content-viewers are computed by joining the preferred audience and by removing the not preferred audience of each stakeholder. In this way, it is guaranteed that each stakeholder contributed to the privacy settings even though their preferences might be restricted. If viewers set is empty, only the stakeholders can see the content. CoPE has a user interface with several features. In addition to the aforementioned features, CoPE enables users to set default privacy settings for all pictures. It also enables co-owners to track the content-viewers who has seen their privacy-contents. A drawback of CoPE is, during the calculation of joint

preferences, co-owners become aware of each other's audience preferences. In addition to that, managing the privacy settings of post is not automatic. Thus, users handle it by themselves. Authors conducted a survey-based user study to evaluate CoPE. Results showed them OSN users like the idea of collaborative privacy management and they can adapt such a tool such as CoPE to manage their privacy.

Apart from majority voting, negotiation is a useful method to handle conflicts and come to an agreement. Such and Rovatsos developed a system that automatically detects and resolves conflicts between different policies [16]. Agents' privacy policies consist of a set of exceptions and intimacy thresholds for specific relationship types. Agent intimacies are considered to be the relationship strengths between agents. Only users who have a higher intimacy value than a predefined intimacy threshold can access the posts. Authors assumed that intimacies between users are available, since there exists tools that accurately estimate them [44] [45] [46]. Agents indicate the users who are able to access to the posts with action vectors. The system takes the privacy policy of agents and their intimacies with other agents. Then, it checks whether the agents' action vectors are equal. If they are not equal, a conflict is detected. Thus, a negotiation process starts between agents to come to an agreement. Authors use a one-step negotiation protocol that is proven to be stable and efficient in the literature. During the negotiation, each agent calculates the utility for each possible action vector that induced by a policy. A vector utility for an agent is calculated by using the number of exceptions its policy involves and its policy's distance to the agent's preferred policy. At the end, each agent propose an action vector and the one that maximizes the product of the utilities is chosen. Authors also studied the performance of such negotiation mechanisms and proposed heuristics to decrease their cost.

Mester *et al.* also developed a negotiation system to protect the privacy of users in OSNs [15]. Each user is represented with an agent and each agent has an ontology that describes the user's constraints and knowledge. When an agent wants to share a post in an OSN, it sends a post request to the other relevant agents. Receiver agents evaluate the post in their ontology. If the post does not violate the privacy of any relevant agents, the post is shared. Otherwise, rejecting agents provide reasons to

not to share the post. Thus, the requester agent can modify the post as it complies with all privacy constraints or it can propose an alternative post to share. Negotiation continues until the post is shared or the limited number of iterations is reached. This work is similar to ours as we also use agents to represent users in OSNs and we benefit from ontologies. However, in both works of Such and Rovatsos and Mester *et al.* agents are not able to question the arguments of the other users. Agents may have incorrect or missing information in their ontology. Thus, their arguments may lead to an unfair decision or even to a privacy violation. Agents need to be able to question and persuade the other agents to prevent this situation.

It is also possible to solve privacy conflicts for images by modifying the images. FaceBlock is developed to protect the privacy of users when their picture is captured by a Google Glass [43]. FaceBlock users share their privacy policy with the FaceBlock application in the Glass to express whether they want to be seen in a picture or not. First, FaceBlock creates an eigenface of the user by using her photo. An eigenface is the mathematical representation of a user's face [47]. Then, whenever a Glass is detected by the user's device, FaceBlock sends the user's eigenface and her policy to the Glass. FaceBlock in the Glass checks every picture it takes and obscures the matching faces. FaceBlock benefits from Semantic Web Technologies to be able to create more fine-grained privacy policies. It benefits from five types of context-aware semantic policies. Location-aware policies enable users to define privacy rules based on locations. Similarly, identity-aware policies make it possible for users to define rules for specific users or user groups and time-aware policies enable users to define rules based on time. Currently some smart devices are able to detect activities. This feature make it possible for users to create activity-aware policies. At last, composite policies include a combination of aforementioned context-aware policies. Users define privacy policies with Semantic Web Rule Language (SWRL) rules and FaceBlock's context ontology. Then, a Description Logics reasoner is used to infer if a privacy policy is triggered with the current context. Privacy policies are shared among users via peer-to-peer networks. Glass only receives a policy including an allow or disallow message. Thus, there is no privacy leakage that reveals the identity and the context model of the user. However, FaceBlock has no mechanism for users to discuss their privacy policies in case of having

conflicting policies. Thus, it is also not possible to persuade the other agent.

Another powerful method to make decisions and handle inconsistencies is argumentation. Muller and Hunter propose an argumentation-based approach for decision making [48]. Their system is able to generate decisions. It uses argumentation to analyse and document decisions. The system uses formal logic to create arguments. Decision making process takes place in a structured way by using arguments, rules and counter arguments. This structured information can be transformed into a decision document. Thus, it is also possible to see the reasons behind the decisions. In the current approaches, a mathematical value is calculated for each decision by considering different criteria. Then, the most favourable decision is chosen. In addition, documentation is made in an unstructured way. However, there are several problems with these current approaches. First, multi criteria decision making (MCDM) only gives an output instead of showing the reasoning process. Second, MCDM provides the best decision among only the predefined decisions. Thus, it gives the local optimum. Another problem is, there is no standard for documentation which prevents to develop tools that supports decision documentation. This also prevents the analyses of decisions that are made before. Another downside of having unstructured documentation is the complexity of finding a particular information in the document. To solve these problems, authors propose an argumentation decision framework (ADF). Arguments support or attack a given set of decisions to find the best decision. ADF meets the aforementioned requirements that MCDM cannot meet. First, the preferred decisions are supported by arguments. Thus, it is possible to see the reasoning process. Since ADF is able to recommend decisions, the decision maker does not need to enumerate possible decisions manually. In addition, it is also possible to automatically analyse decisions with ADF, since rules in the argumentation provide domain knowledge. ADF also enables standardization, since it uses a formal language, which can be easily converted to an ontology and vice versa.

Kakas and Moraitis proposed an argumentation based decision making system for agents [49]. They created a modular agent architecture as each module represents a capability of an agent. These modules have their own deliberation processes and

they reason independently. The behaviour of an agent is decided by intersecting the decisions of these modules. Different roles of an agent in an environment and the context of its interactions may change its decisions. Thus, argumentation theory of the agent consists of three levels. In the first level, object-level decision rules are defined. In the second and third level, the object-level decision rules are used for different roles and contexts. This structure gives modularity to the argumentation theory so that when a rule wants to be modified, making a change in the associated level is enough. Authors also consider the situation when an agent does not have enough information in its knowledge base to produce an argument. In this case, agents produce arguments with abducible predicates. This supporting information can be acquired by the environment observer agent itself or as a result of interactions with other agents. Modularity of the architecture enables agents to be adaptable. One of the modules is personality, which creates a decision policy based on the motivations and needs of an agent. Authors benefit from five main classes that are categorized in Cognitive Psychology to show the needs and motivations of an individual. These classes are Psychology, Safety, Affiliation, Achievement and Learning. Authors assume that there is an ordering between these needs and agents make a decision by considering them. Each goal of an agent is linked to one of the five motivations. Thus, agents also check if a motivation is critical while choosing a goal. In this way the personality and the behaviour of the agents can be designed. Personality module plays an important role as it can change the overall decisions of the agent by influencing the other modules. However, personality module also can lead to dilemmas, since an agent's personal goals may be different from its professional goals. Authors proposed three ways to handle these dilemmas. First, the agent can suspend the decision. Second, the agent can try to find supporting information to strengthen or weaken the goals. At last, the agent can try to evaluate the supporting information in the external environment. There exist an implementation of this framework. PriArg is different from the work of Kakas and Moraitis as it is different from the work of Muller and Hunter. PriArg enables distributed argumentation where more than one agents uses argumentation to come a decision. On the other hand, both of these works uses a central argumentation system where an agent makes a decision by itself. Further, agents in both work do not use

decision making for privacy domain.

Another work that benefits from both ontology and argumentation is OAF (Ontology-based Argumentation Framework), which is an extension of DeLP(Defeasible Extended Logic Programming) [50]. OAF enables a decision making system for breast cancer. Authors use real clinical trials to export defeasible rules and to create an ontology. Then, they create arguments by using the ontology, facts and rules. Since some arguments may conflict each other, they try to find the warranted and unwarranted arguments. In this point, they also take the power of the arguments into account by considering the preference order. As in this work, we also use both ontology and argumentation in our research. However, our agents make decisions in privacy domain.

In addition to decision making approaches for one agent, there are other works that using argumentation for decision making with multiple agents. Fan *et al.* propose a multi-agent decision framework [36]. The framework uses argumentation-based dialogue so the agents can exchange information, discuss on decisions, goals and attributes and choose decision. An ABA-dialogue consist of utterances whose content can be a claim, a rule, an assumption or a contrary. Each agent has a decision framework, which involves a set of decisions, a set of attributes, a set of goals, a goal-attribute table and a decision-attribute table. A decision-attribute table shows if a decision has an attribute and a goal-attribute table shows if a goal is satisfied by an attribute. Decision frameworks are combined to create the joint decision framework. Since there are two different parties, conflicts between the decision frameworks may occur. These conflicts can be resolved in two ways. First, if the public trustworthiness of the agents exist, the conflict is resolved in the support of more trusted agent. Otherwise, the conflict is resolved by using three different joint decision frameworks. In sceptical joint decision framework, both agents must believe that a decision-attribute pair or a goal-attribute pair must be true to make the same pair true in the joint decision framework table. In credulous joint decision framework, it is enough for one agent to say that a decision-attribute pair or a goal-attribute pair is true to make the same pair true in the joint decision framework table. In the fair joint decision framework, a decision-attribute pair or a goal-attribute pair is true if and only if one agent believes that it is true and

the other agent either believes the same thing or does not have a knowledge about it. After having a decision framework, each agent can construct its own ABA framework. Then, union of the two agents' frameworks are used to compute the selected decision. Authors implemented a dialogue system with JADE and Grapharg. This works differs from ours as we benefit from ontologies to represent the knowledge base of agents. Agents can infer new information from the existing knowledge by following SWRL rules. Further, they use decision making for medical domain whereas we use it for privacy domain.

Fogues *et al.* propose to use a privacy recommendation tool that would help users to set privacy constraints to a post considering all of the related users' privacy constraints [51]. For that aim, agents that acts on behalf of the users, negotiate through arguments and try to convince each other that their privacy constraints are necessary and should be taken into account. This work is similar to ours, since they also try to find the optimal privacy settings for a post by letting agents negotiate each other by using arguments. However, they do not define how an agent decide what is suitable for it. We propose every agent to have an ontology so they can reason and decide to the privacy constraints of a post and provide arguments to support their claim. Furthermore, we also provided an algorithm for agents to carry the argumentation process and decide. Note that none of the aforementioned works above except the work of Kakas *et al.* enable agents to ask for external knowledge as PriArg does. Thus it is not possible for an agent to learn new information from other agents.

In our future work, we would like improve PriArg as it satisfies the concealment aspect. One approach may be revealing the privacy constraints partially as the reasons behind the rules are hidden from the others. Another future work would be explaining users the reasons behind a decision of a discussion. Currently, the explanation that PriArg provides may not be easily understandable for users. One approach may be extracting information from the dialectical trees of arguments and providing a plain text for a more understandable explanation. At last, it would be interesting to bring PriArg a trust aspect. In personal interviews, we find out how users consider the difference between an assumption and a fact for privacy. The result presents the importance of

absolute information for users. It would be interesting to study how an agent's attitude changes when it obtains uncertain information from other agents that are trusted in different levels.

REFERENCES

1. Ellison, N. B. *et al.*, "Social network sites: Definition, history, and scholarship", *Journal of Computer-Mediated Communication*, Vol. 13, No. 1, pp. 210–230, 2007.

2. Maleewong, K., C. Anutariya and V. Wuwongse, "A collective intelligence approach to collaborative knowledge creation", *Semantics, Knowledge and Grid, 2008. SKG'08. Fourth International Conference on*, pp. 64–70, IEEE, 2008.

3. Wishart, R., D. Corapi, S. Marinovic and M. Sloman, "Collaborative Privacy Policy Authoring in a Social Networking Context", *Proceedings of the IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*, pp. 1–8, Washington, DC, USA, 2010.

4. Squicciarini, A. C., H. Xu and X. L. Zhang, "CoPE: Enabling Collaborative Privacy Management in Online Social Networks", *Journal of the American Society for Information Science and Technology*, Vol. 62, No. 3, pp. 521–534, 2011.

5. Stross, R., "How to lose your job on your own time", `http://www.nytimes.com/2007/12/30/business/30digi.html?ex=1356670800&en=55ef6410d3cac28e&ei=5088&partner=rssnyt&emc=rss`, 2007, accessed at May, 2016.

6. Grasz, J., "Forty-five Percent of Employers Use Social Networking Sites to Research Job Candidates, CareerBuilder Survey Finds", `http://www.careerbuilder.com/share/aboutus/pressreleasesdetail.aspx?id=pr691&sd=4/18/2012&ed=4/18/2099`, 2012, accessed at May, 2016.

7. Maternowski, K., "Campus police use Facebook", `https://badgerherald.com/news/2006/01/25/campus-police-use-fa/`, 2006, accessed at May, 2016.

8. Shachtman, N., "Exclusive: U.S. Spies Buy Stake in Firm That Monitors Blogs, Tweets", `https://www.wired.com/2009/10/`

exclusive-us-spies-buy-stake-in-twitter-blog-monitoring-firm/, 2009, accessed at May, 2016.

9. Thomas, K., C. Grier and D. M. Nicol, "unfriendly: Multi-party privacy risks in social networks", *Privacy Enhancing Technologies*, pp. 236–252, Springer, 2010.

10. Westin, A. F., "Privacy and freedom", *Washington and Lee Law Review*, Vol. 25, No. 1, p. 166, 1968.

11. Kökciyan, N., "Privacy Management in Agent-Based Social Networks:(Doctoral Consortium)", *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 2019–2020, International Foundation for Autonomous Agents and Multiagent Systems, 2015.

12. Eecke, P. V. and M. Truyens, "Privacy and social networks", *Computer Law & Security Review*, Vol. 26, No. 5, pp. 535 – 546, 2010, http://www.sciencedirect.com/science/article/pii/S0267364910001093, accessed at May, 2016.

13. Stewart, M. G., "How giant websites design for you (and a billion others, too)", https://www.ted.com/talks/margaret_gould_stewart_how_giant_websites_design_for_you_and_a_billion_others_too, accessed at January, 2016.

14. Squicciarini, A. C., F. Paci and S. Sundareswaran, "PriMa: a comprehensive approach to privacy protection in social network sites", *Annals of Telecommunications/Annales des Télécommunications*, pp. 1–16, 2013.

15. Mester, Y., N. Kökciyan and P. Yolum, "Negotiating Privacy Constraints in Online Social Networks", F. Koch, C. Guttmann and D. Busquets (Editors), *Advances in Social Computing and Multiagent Systems*, Vol. 541 of *Communications in Computer and Information Science*, pp. 112–129, Springer International Publishing, 2015.

16. Such, J. M. and M. Rovatsos, "Privacy Policy Negotiation in Social Media", *CoRR*, Vol. abs/1412.5278, 2014, `http://arxiv.org/abs/1412.5278`, accessed at May, 2016.

17. "Abagraph", `http://www.doc.ic.ac.uk/~rac101/proarg/abagraph.html`, accessed at May, 2016.

18. Wooldridge, M. and N. R. Jennings, *Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages Amsterdam, The Netherlands August 8–9, 1994 Proceedings*, chap. Agent theories, architectures, and languages: A survey, pp. 1–39, Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.

19. Franklin, S. and A. Graesser, "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", *Intelligent agents III agent theories, architectures, and languages*, pp. 21–35, Springer, 1996.

20. Jennings, N. R., K. Sycara and M. Wooldridge, "A roadmap of agent research and development", *Autonomous agents and multi-agent systems*, Vol. 1, No. 1, pp. 7–38, 1998.

21. Schmidt, A., M. Beigl and H.-W. Gellersen, "There is more to context than location", *Computers & Graphics*, Vol. 23, No. 6, pp. 893–901, 1999.

22. Horrocks, I., P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean *et al.*, "SWRL: A semantic web rule language combining OWL and RuleML", *World Wide Web Consortium Member submission*, Vol. 21, p. 79, 2004.

23. Fang, L. and K. LeFevre, "Privacy wizards for social networking sites", *Proceedings of the 19th international conference on World wide web*, pp. 351–360, ACM, 2010.

24. Mugan, J., T. Sharma and N. Sadeh, *Understandable learning of privacy preferences through default personas and suggestions*, Institute for Software Research Technical Report CMU-ISR-11-112, Carnegie Mellon University, Pittsburgh, PA,

2011.

25. Amgoud, L. and C. Cayrol, "Inferring from inconsistency in preference-based argumentation frameworks", *Journal of Automated Reasoning*, Vol. 29, No. 2, pp. 125–169, 2002.

26. Dunne, P. E., A. Hunter, P. McBurney, S. Parsons and M. Wooldridge, "Weighted argument systems: Basic definitions, algorithms, and complexity results", *Artificial Intelligence*, Vol. 175, No. 2, pp. 457–486, 2011.

27. Amgoud, L. and H. Prade, "Using arguments for making and explaining decisions", *Artificial Intelligence*, Vol. 173, No. 3, pp. 413–436, 2009.

28. Dung, P. M., "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games", *Artificial intelligence*, Vol. 77, No. 2, pp. 321–357, 1995.

29. Dung, P. M., R. A. Kowalski and F. Toni, "Assumption-based argumentation", *Argumentation in Artificial Intelligence*, pp. 199–218, Springer, 2009.

30. Modgil, S. and H. Prakken, "The ASPIC+ framework for structured argumentation: a tutorial", *Argument & Computation*, Vol. 5, No. 1, pp. 31–62, 2014.

31. García, A. J. and G. R. Simari, "Defeasible logic programming: Delp-servers, contextual queries, and explanations for answers", *Argument & Computation*, Vol. 5, No. 1, pp. 63–88, 2014.

32. Besnard, P. and A. Hunter, "Constructing argument graphs with deductive arguments: a tutorial", *Argument & Computation*, Vol. 5, No. 1, pp. 5–30, 2014.

33. "CaSAPI,", `http://www.doc.ic.ac.uk/~ft/CaSAPI/`, accessed at May, 2016.

34. "Proxdd", `http://www.doc.ic.ac.uk/~rac101/proarg/proxdd.html`, accessed at May, 2016.

35. Toni, F., "A tutorial on assumption-based argumentation", *Argument & Computation*, Vol. 5, No. 1, pp. 89–117, 2014.

36. Fan, X., F. Toni, A. Mocanu and M. Williams, "Dialogical Two-agent Decision Making with Assumption-based Argumentation", *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pp. 533–540, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2014.

37. "Protege", `http://protege.stanford.edu/`, accessed at April, 2016.

38. Horridge, M. and S. Bechhofer, "The OWL API: A Java API for OWL Ontologies", *Semantic Web*, Vol. 2, No. 1, pp. 11–21, 2011.

39. Sirin, E., B. Parsia, B. C. Grau, A. Kalyanpur and Y. Katz, "Pellet: A practical OWL-DL reasoner", *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 5, No. 2, pp. 51–53, 2007.

40. "SICStus", `https://sicstus.sics.se/`, accessed at April, 2016.

41. Gansner, E. R. and S. C. North, "An open graph visualization system and its applications to software engineering", *SOFTWARE - PRACTICE AND EXPERIENCE*, Vol. 30, No. 11, pp. 1203–1233, 2000.

42. "QuestionPro", `http://www.questionpro.com/`, 2005, accessed at May, 2016.

43. Pappachan, P., R. Yus, P. K. Das, T. Finin, E. Mena and A. Joshi, "A Semantic Context-aware Privacy Model for Faceblock", *Proceedings of the 2nd International Conference on Society, Privacy and the Semantic Web - Policy and Technology*, PrivOn, pp. 64–72, 2014.

44. Fogués, R. L., J. M. Such, A. Espinosa and A. Garcia-Fornes, "BFF: A tool for eliciting tie strength and user communities in social networking services", *Information*

Systems Frontiers, Vol. 16, No. 2, pp. 225–237, 2014.

45. Gilbert, E., "Predicting tie strength in a new medium", *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pp. 1047–1056, ACM, 2012.

46. Gilbert, E. and K. Karahalios, "Predicting Tie Strength with Social Media", *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pp. 211–220, ACM, New York, NY, USA, 2009.

47. Sirovich, L. and M. Kirby, "Low-dimensional procedure for the characterization of human faces", *Josa a*, Vol. 4, No. 3, pp. 519–524, 1987.

48. Muller, J. and A. Hunter, "An argumentation-based approach for decision making", *IEEE 24th International Conference on Tools with Artificial Intelligence (ICTAI)*, Vol. 1, pp. 564–571, 2012.

49. Kakas, A. and P. Moraitis, "Argumentation based decision making for autonomous agents", *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 883–890, ACM, 2003.

50. Williams, M. and A. Hunter, "Harnessing Ontologies for Argument-Based Decision-Making in Breast Cancer", *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, Vol. 2, pp. 254–261, Oct 2007.

51. Fogues, R., P. Murukanniah, J. Such, A. Espinosa, A. Garcia-Fornes and M. Singh, "Argumentation for multi-party privacy management", *The Second International Workshop on Agents and CyberSecurity (ACySe)*, pp. 3–6, 5 2015.

APPENDIX A: APPLICATION

1. What is your age?
    a. 18-24
    b. 25-34
    c. 35-44
    d. 45-54
    e. 55-64
    f. 65+

2. What is your gender?
    a. Female
    b. Male

3. How often do you use social networking sites?
    a. At least once in a day
    b. At most three times in a week
    c. At most once in a week
    d. Others

4. Do you check who can see the content before you share it in a social network?
    a. Yes
    b. No

Figure A.1. Demographic questions for Personal Interview and Online Survey

Warnings

1. Please only use the information that are provided below. Do not use your own privacy constraints.
2. Please do not consider yourself as neither Alice nor Bob but answer the questions objectively.
3. In each step, use all of the information that are provided you in the previous steps.

Scenario

Alice and Bob are friends in a social network. Alice wants to share a picture of Bob. The picture shows Bob with a wristband, which is obtained from the Oktoberfest. They involve in a discussion and try to decide whether they can share the picture or not.

1. Bob does not want any picture, which signals his attendance to a festival to be shared in a social network. He states that if a picture includes an object, which is unique and is taken from a festival, then the picture signals his attendance to the festival. Bob knows that the wristband is taken from the Oktoberfest and he thinks that the wristband is unique.

   Can this picture be shared?
   a. Yes
   b. No

2. As a response, Alice tells that if an object can be found in a shop, then it is not unique. She also tells that she thinks the wristband can be found in a shop named Gifty.

   Can this picture be shared?
   a. Yes
   b. No

3. Then, Bob tells that if the web site of a shop cannot be reached, then the shop is closed. He checks the website of Gifty and realizes that it cannot be reached.

   Can this picture be shared?
   a. Yes
   b. No

Figure A.2. Provided scenarios for Personal Interview