SN Computer Science (2021) 2:338
https://doi.org/10.1007/s42979-021-00714-5

ORIGINAL RESEARCH

Empowering Users in Online Open Communities

Nardine Osman[1] · Ronald Chenu-Abente[2] · Qiang Shen[3] · Carles Sierra[1] · Fausto Giunchiglia[2]

Received: 23 September 2020 / Accepted: 18 May 2021 / Published online: 14 June 2021

Abstract
In this paper, we propose an architecture supporting *online open communities*, where by *open* communities, we mean communities where previously unknown people can join, possibly for a limited amount of time. The fundamental question that we address is "how we can make sure that an individual's requirements are taken into consideration by the community while her privacy is respected and the community's ethical code is not violated". The main contributions are: (i) a conceptual framework which allows to describe individual and community *profiles*, including data and norms that provide information about their owner and their requirements, and (ii) a *decentralised architecture* enabling interactions that leverage the exchange of profile information among people and communities to ensure that requirements are fulfilled and privacy is respected.

Keywords Online communities · Social relations · Context · Norms · Privacy

Introduction

The huge success of *social networks*, e.g., Facebook, Whatsapp, WeChat, has highlighted the importance of *online social relations*, where the key novelty is the possibility of enabling interactions which transcend the limitations of space and time. Thanks to social networks, it is possible for anybody to interact in real time, in writing or by talking, to virtually anybody else in the world, independent of their physical location. The implications of this success are obvious and involve a huge amplification of social relations, thus enabling the creation of large scale *online communities*. This phenomenon has been extensively studied in the literature, see, e.g., [14, 20, 53].

Following the vision described in [22], in this paper, we propose to move "from a network of computers, which in turn may be connected to people, to a network of people, whose interactions are mediated and empowered by computers" (quote from [22]). In other words, after investing in the last few decades on how the machine may be put at the service of humans in online networks (as in, for example, the Internet or the IoT), we now highlight the need to bring back the human as a critical source of providing support, and not just receiving it.

This article is part of the topical collection "Advances in Multi-Agent Systems Research: EUMAS 2020 Extended Selected Papers" guest edited by Nick Bassiliades and Georgios Chalkiadakis.

This paper is a substantially extended and revised version of [48]. The main changes are: improving and extending the introduction, considering norms as part of the profile ("Profile" section), adding a discussion on how to build and share the profiles ("Profile building and sharing" section), adding a description of the decision engine ("The decision engine" section), and changing the motivating example ("A motivating example" section).

✉ Ronald Chenu-Abente
chenu@disi.unitn.it

Nardine Osman
nardine@iiia.csic.es

Qiang Shen
shenqiang19@mails.jlu.edu.cn

Carles Sierra
sierra@iiia.csic.es

Fausto Giunchiglia
fausto@disi.unitn.it

[1] Artificial Intelligence Research Institute (IIIA-CSIC), Barcelona, Spain

[2] Dipartimento di Ingegneria e Scienza dell'Informazione, Univerità di Trento, Trento, Italy

[3] College of Computer Science and Technology, Jilin University, Changchun, China

This entails careful work on online social relations, which has its difficulties, one of the main issues relating to their quality. As discussed in, e.g., [14, 50], computer mediated interactions can be, and often are, less valuable in building and also in sustaining close relations. However, despite this, the possibility of developing high value online communities seems quite promising [19, 54]. Social relations then become more intense and can be applied for objectives which go far beyond the exchange of short messages or discussions about the current topic of interest; this being grounded in the fact that, in the real world, social relations allow people not only to interact but also to help one another using their collective strengths as the means for overcoming individual weaknesses. A collective has capabilities that are much more than the sum of the capabilities of any single member [54]. The intrinsic diversity of people (in their characteristics, knowledge, skills, competencies, and much more) is something that people use in their everyday life, often without even realising it. We ask a doctor for a diagnosis and a treatment when we are ill, we call up a plumber if a pipe leaks in our house, and look for someone speaking our language and Chinese if we need to sort out arrangements for a stay in Beijing. Scaling up this possibility to the whole size of the Internet would immensely enhance the ability to solve certain tasks, thanks to the help and support of third parties. And this applies to any type of need, ranging from a person who provides you a service (as in the plumber example), a need whose satisfaction requires some follow-up action in the real world, or a pure informational need (as in the Chinese example), where the need can be solved online.

Many social networks today seem to address this very problem. Current online social networks like Instagram and TikTok are based on enabling social relations with previously unknown people. Many are used for connecting people to solve specific human needs, like TaskRabbit (http://www.taskrabbit.com), Upwork (http://www.upwork.com), or PeoplePerHour (http://www.peopleperhour.com). Current ego networks [3, 38, 51, 61] already allow users to apply the support and skills of a large number of people.

However, existing solutions suffer from their use of rigid interaction protocols that leave their users without much control over their data and interactions. This paper's main novelty is in giving users: (1) control over their profile data, over how this data can be shared, with whom, and under which circumstances; and (2) control over how interactions are carried out within communities. The main question we address is how can we make sure that an individual will have her needs taken into consideration, leveraging the available profiles, while ensuring that her privacy is respected and the community's ethical code is not violated.

The problem of privacy online is well known and largely studied, see, e.g., [35, 36, 39, 59, 62] and has caused various studies and analyses, see, e.g., [11, 55], as well as the generation of considerable legislation,not only in Europe above all, but also worldwide [17, 18]. However, this problem grows enormously in the case of online (open) communities, given that their enablement requires sharing information which is far more sensitive than that needed in the state of the art social networks [32].

The main contribution of this paper is thus the definition and articulation of an architecture enabling and supporting online open communities, with a dedicated focus on the individual and her needs and giving special attention to privacy. Towards this end, the main components of the proposed solution are:

- A conceptual framework which allows for describing individual and community profiles, including data and norms that provide information about their owner. We argue that people and communities must build their own profile which can then be used by third parties to discover the most suitable person who can help them with the task at hand. We also argue the need to empower people and communities in selecting the visibility of the profile as a trade-off between *privacy* and *openness*. On one hand, there is a need to prevent personal information to be shared with unknown and possibly malicious people, while on the other hand, there is a need to allow for some level of personal information sharing. If nobody knows about you or about how to contact you, then no social interactions can be enabled. The solution provided is that the level of information that a person will share will depend on the *context* [10, 21, 24], e.g., the type of information itself, the goal and the people involved.
- A decentralised architecture for social networks that helps achieve the above goals by mediating social interactions through community norms. The proposed architecture empowers community members by allowing them to specify their individual rules and data that describe them, as well as to specify whom to share this information with and under what circumstances.

The paper is structured as follows. "Profile" section introduces our conceptual framework for profiles, which are composed of data and norms. "Profile building and sharing" section discusses profile building and profile sharing, a cornerstone for addressing privacy. "Architecture and operational model" section introduces the decentralised architecture addressing the conceptual framework, while "A motivating example" section provides a motivating example. The related work is discussed in "Related work" section before concluding with "Past, current and future work" section.

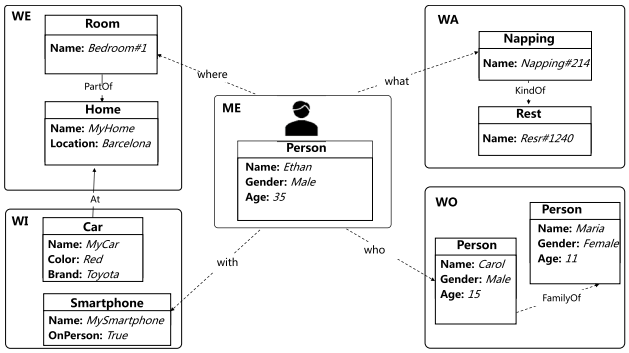

Fig. 1 An example of situational context

Profile

A profile is a description of an entity, which, in turn, we take to be a person or a community. When building a software system that supports a particular social interaction, for any entity, it is fundamental to define: (i) what particular attributes are relevant for the interaction with other entities, so that their values (i.e., *data*) should be gathered; and (ii) what rules of behaviour (i.e., *norms*) of entities affect the social interaction being modelled.

Definition 1 (*Profile*) A profile P is a pair of data D and a set N of norms: $P = (D, N)$

Next, we analyse the two components of a profile in detail.

Data

A profile will contain an ample amount of data about that entity, information needed to suitably interact in a certain open community. We illustrate here the dimensions of a person's *situational context* [24], noting that a profile of any entity may consist of multiple contexts, for instance describing the entity's physical characteristics, its competences or, even an ongoing conversation. Figure 1 shows a small example of Ethan's situational context D. We focus on the situational context for three main reasons. The first is that it highlights the privacy issues that can be raised in relation to personal information. The second is that this information is, of course, very dynamic, thus making privacy a problem which must continuously be dealt with, with the assurance that the information provided to third parties at a certain moment of time will not hold any longer than necessary. The third is that the situational context plays a crucial role in the possibility for a certain individual to engage, within a community, in a social interaction.

A situational context is composed of four main sub-contexts, WE, WA, WO, and WI, as follows:

WE is a spatial context which captures the exact location, e.g., "Home" or "Barcelona". We refer to it as the answer to "**Wh**ere are you?" in the case of a person and "**Wh**ere is the community located?" for a community.

WA is a behavioural context which captures the activity, e.g., "napping". Informally, it answers the question "**Wh**at are you doing?" for a person and "**Wh**at does the community do?" for a community.

WO is a social context which captures the social relations, or the answer to "**Wh**o are you with?" (e.g., the "family"), for a person, and "**Wh**o do you collaborate with?" for a community.

WI is object context which captures the materiality, e.g., "smartphone" or "car". It represents the object you currently have. Informally, it answers the question "**W**hat are you w**I**th?" for a person and "**W**hat **I**nfrastructure does the community have?" for a community.

We model a person's situational context D, which we refer to as the data part of the profile, as a *knowledge graph* [8, 16, 34], which we define as the union of four smaller knowledge graphs *WE*, *WA*, *WO*, *WI*.

Definition 2 (*Data*) The data D, representing the context inside of the profile, are the union of the four dimensions of situational context:

$$D = (WE \cup WA \cup WO \cup WI)$$

In this setting, we define a knowledge graph as follows:

- nodes represent entities, namely anything physical, digital, conceptual, real or imaginary which is described via a set of properties, i.e., attributes and relations (e.g., MyCar, Barcelona, Ethan);
- information about these nodes is represented as attributes, namely entity value pairs (e.g., Location (Barcelona), Gender(Ethan) = Male, Age(Ethan) = 35);
- links represent relations among entities, namely a limited set of pairs of entities describing how they relate (e.g., where (Ethan, Bedroom#1), who(Ethan, Carol), with (Ethan, MyCar), partOf (Bedroom#1, MyHome)).

Notice that a knowledge graph like the one defined above can be mapped one-to-one into a Description logic where entities (e.g., Ethan) are instances populating concepts (e.g., Person), while attributes and relations are pairs

populating, respectively, data and object properties, see, e.g., [4, 49]. This knowledge graph, in turn, can be easily represented and exported in terms of RDF triples.

We assume that a profile is continuously enriched with data coming from various sorts of streams, requiring to store the changing values of the most relevant attributes. These streams of information can be sensor data (e.g., GPS, accelerometer, giroscope, bluetooth) which are then used to learn the various types of information stored in D. Some of this information are directly provided by the user, properly asked by the system. This topic is not described here because it is out of scope. Bontempelli et al., Giunchiglia et al., Zeni et al. [9, 28, 63, 65] provide a long list of concrete examples of how this can be done. From a practical point of view, D can be considered as consisting of lifelogging data [7, 31], which can be formalised as:

$$D^t(u) = \langle D^1, D^2, D^3, \ldots, D^t \rangle, t \rightarrow +\infty$$

where $D^t(u)$ is the data profile of user u at time t, t is growing along the user's life, and the size of streaming profile is thus continuously increasing. It is worth noticing that the problem of an ever-growing profile is dealt with by implementing various forms of selective *forgetting*. The results, which are much more compact are then stored in a long-term memory. Thus, again, [9, 28, 63, 65] provide examples of the kind of learning we perform over data from a 2-week period.

Norms

Norms are rules that specify behaviour at the individual and the social level. They determine what actions are acceptable, who can an individual interact with, and under what circumstances, etc. So far normative systems have mostly focused on the action, namely on 'what' can one do; here, we focus on the other crucial aspect of interactions, namely on 'who' can one interact with, this being more and more relevant in an increasingly hyper-connected world. To achieve this, we take norms as the second component of individual and community profiles (Definition 1). Behaviour is as important in social interactions as individuals' gender, age, or relationships. For example, one individual norm can say "only seek help from people around me", while another can say "never bother me when I am napping".

Traditionally, in multiagent systems, norms have been specified through deontic operators that describe what is permitted, forbidden, or obligatory [60]. We propose a simple approach that specifies norms as *if-then* statements that specify who can perform what action, and under what condition. For instance, the above two individual norms may be specified as:

IF
$seek\_help(Person, Task)$
`and` $location(Person, City)$
`and` $friends\_around(City, List)$
THEN
$forward\_seek\_help(List, Task)$

IF
$naptime(true)$ `and` $notify(X)$
THEN
$suppress\_notification(X)$

The norm part of a profile is then taken to be a set of such if-then statements, and defined accordingly.

Definition 3 (*Norm*) A norm $n \in N$ is defined as an if-then statement: $n =$ `IF` *Condition* `THEN` *Consequent*, where *Condition* and *Consequent* are expressions, or formulae, defined as follows:

- Each atomic formula is a formula.
- If C and C' are formulae, then C `and` C' is a formula.
- If C is a formula, then $\neg C$ is a formula.

The profile may be the profile of an individual or a community, and as such and just like the data part of the profile, the norm part will also describe the rules of behaviour of the individual or the community, respectively. When norms are part of the individual profile, we refer to them as *individual norms*, and when they are part of the community profile, we refer to them as *community norms*. Notice how in this setting by 'community' we mean both an organisation as we have in the real world, e.g., the University of Trento, as well as an online group of people, more or less informally organised.

Community norms govern the behaviour of the community they are associated with, including its members. Any action (represented by a message exchange) in the peer-to-peer network of this community must be coherent with these norms. For instance, a norm in a mutual aid community that prohibits members from abusing the community by always asking for help and never offering help, or a norm that punishes those that do not fulfil their duties by suspending their memberships. We consider an action acceptable by the community when it does not violate any of the community's norms.

Community norms can be divided into a number of categories. For example, *institutional* norms can describe the rules of behaviour in the given community (following the concept of electronic institutions [15]). *Ethical* norms

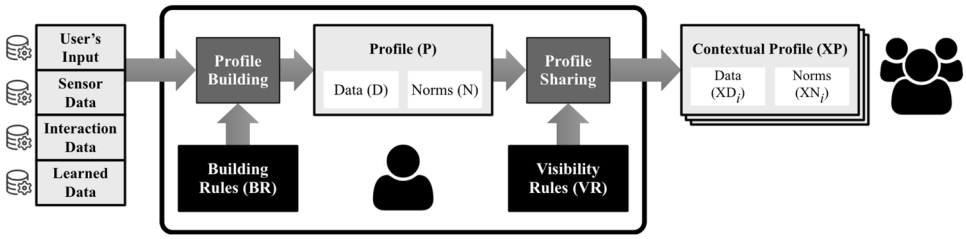

Fig. 2 How profiles are generated and shared

can describe what is considered ethical and what actions are deemed unethical, and hence, unacceptable in the community. *Incentive* norms can help provide incentives for community members to behave in a certain way, such as encouraging benevolent behaviour, say to help maintaining the community and fulfilling its objectives. And so on.

Individual norms are rules that govern the behaviour of the individual they are associated with. They represent particular aspects of the relationship of the human with her device (mobile, tablet, computer) and with the community. For instance, a prohibition to pop-up a message during a siesta. Or an obligation to filter out messages coming from people that are not in one's vicinity. Of course, individual norms may implement certain behaviour that may not be fully aligned with the community norms. Therefore, some behaviour that is deemed 'unacceptable' or even 'unethical' by the community may be codified at this level and remain unnoticed by the community, simply because individual norms represent the individual's requirements with respect to behaviour and not that of the community. In cases of conflict between community and individual norms, community norms prevail concerning actions within the community. For example, if community norms prohibit discriminating against women, then an individual norm that asks to exclude females from a given activity will be overruled by the community's norm. However, individual norms prevail when concerning actions local to one's device.[1] For instance, while community norms may prohibit discriminating against women, one's individual norm can enforce requests coming from women to be suppressed (ignored).

Last, but not least, we note that like data, norms evolve over time. While I might accept requests to play padel from anyone today, in the future, I might change my mind and restrict receiving such requests to those made by padel professionals only.

[1] Here we talk about actions local to one's device, regardless of whether the computations behind these actions (e.g. a decision to send a notification to the user) are performed locally on the same device, executed to the cloud, or a combination of both.

Profile Building and Sharing

Apart from *what* is to be represented in a profile, which was presented in the previous section, there are two other fundamental questions to be addressed by a profile management system. First, *how* is the information in a profile obtained? Second, *who* has access to it? We will address the how and who in the next subsections.

Profile Building

There are different mechanisms to obtain profile information, from simply asking the individual or the community in question (or its representative) to manually provide this information, or using sensor data that can automatically learn things (like location, busy hours, heart rate, ...), to using interaction data and learned data (e.g. observing who does one interact with often, who is usually preferred for playing padel, ...). Given the different mechanisms available for obtaining profile data, it is very important to always ensure that the associated individual or community is the one deciding which of those mechanisms to use, and under what conditions. In other words, the individual or community decides how their profile is built. This is specified through *profile building rules*. For example, one individual may decide to disable all sensor data while another might permit the GPS sensor to sense its current location, and one community might only permit its president to manually provide information about it while another might permit any of its users to do so. One community may re-use, adapt, or build on top of existing norms (for example, a new social network may re-use the institutional norms of an existing social network and adapt them to their community's particular needs), whereas another might bring its members to collaboratively specify its norms. While we always stress the need for the entity in question (whether an individual or a community) to be in control, we note that how a community reaches a decision on its profile building rules is outside the scope of this paper, which could be achieved through collective agreements or other means.

The left hand side of Fig. 2 illustrates that the profile building rules (BR), specified by the profile owner, are responsible for building the private profile from different data sources. We note that how such data is gathered is

largely beyond the scope of this paper, referring the reader to previous work on the gathering of data [64].

Profile Sharing

Once a decision is made on what data to include in a profile and established the means to gather it, the remaining fundamental question is *who* is granted access to what part of the profile. As illustrated earlier, we require that the individual or community has full access and control over their profile. To have control over who is the profile (or parts of the profile) shared with, we require individuals and communities to define *visibility rules* that determine under what circumstances someone can see part of the individual or community profile. In this respect, an individual's take on privacy (similarly for communities) will determine how she grants access to her profile. Similar to the building rules, we note that how a community reaches a decision on its visibility building rules is outside the scope of this paper, which may be through collective agreements or other means.

Our stance is that *privacy is not an absolute value*. In other words, not all communities have the same stance on what privacy is. For instance, consider the issue of revealing your ID number. Some community that aims at supporting the elderly might find it crucial to have the ID number of the people visiting the elderly at home. Another community that aims at organising political activities might find that revealing one's ID number is a blatant breech of their users' privacy. Additionally, we say that *privacy is fully contextual*. There is information that one may be willing to share with their family but not with their friends and even less with their foes. For example, one may be happy to share their exact location with friends, and maybe friends of friends, but not with strangers. Some may be happy to share their current city with strangers, while others would not even share that. Therefore, we adopt the notion of privacy being fully contextual in the sense that it depends on the current situation as well as the objectives that one wants to achieve.

The contextuality of privacy brings up the key observation that in social relations there is always a dilemma between *privacy* and *transparency*. On one hand, I may prefer that sensitive information is not made public to avoid its misuse, and on the other hand, I want others to know everything about me that is relevant for the social interaction to help achieve my objectives. This dilemma applies also to online open communities.

Our proposed solution is that the profile elements, data and norms, can be either kept private or can be shared with others. Sharing with others does not necessarily mean making it 'public' (although that would certainly be an extreme case of sharing with others), but it means that the access to the information is granted under certain circumstances. For instance, allow my friends to know my exact location when I am making a request to meet up.

The right hand side of Fig. 2 illustrates that the visibility rules (VR) are responsible for extracting, from the private profile, the profile data that may be shared with others in different contexts. We elaborate on the context and the contextual profiles shortly.

Note that to have complete control over a profile, building and visibility rules are both needed to be specified by the profile owner, whether an individual or a community (the black boxes of Fig. 2).

Private Profile

As discussed in [52] (see the related work for details), a profile will contain information about all the relevant aspects of the life of a person or a community, e.g., demographics, personality [13], competences [33], skills or investment plans, but also data which continuously change in time, even during the day, e.g., location, activities, people one is with. This *complete* set of data and norms are, by default, private to (and hence, accessible only by) the profile owner (individual or community) and the system running on the profile owner's own device (we refer to this system that is responsible for making decisions and executing actions the 'decision engine', and it is explained in further detail in "The decision engine" section). This complete and private profile is what is referred to simply as 'Profile' in Fig. 2, and has been defined in Definition 1. For instance, if 'location("Calle Enric Granados 15, 08008 Barcelona")' is part of Alice's private profile this means that the system (decision engine) running on Alice's device has permission to use Alice's location in the reasoning, but no one else can. Private norms are those that are never shared with other entities (individual or community) or devices (e.g. 'never bother me when I am taking a nap'). Their impact on behaviour is restricted to one's own device as other devices do not have access to these norms.

Shared profile

The complete profile provides a memory of the complete description of the entity in question (individual or community). Given such a memory, a shared profile is built based on current contextual needs. This is a set of attributes and norms that can be made accessible to others, both humans and the systems (decision engineers) running on their devices. The mechanism of building a shared profile is analogous to the one people use when meeting another, previously unknown, person and need to provide her with enough information for the task at hand (e.g. certain approaches [6] take inspiration in this model to implement semi-automatic systems for the sharing of information with others at different granularity based on their requests and requirements).

Similarly, one may require to abstract the contextually relevant information from the profile and create a shared profile that will preserve privacy by hiding details that are not relevant to the current situation while still containing the information that is needed for the interaction or task. For example, share my age but not my date of birth, or share the city where I live but not the exact address.

We say shared profiles are to be shared with specific people under certain conditions that define the context of the shared profile. For example, besides sharing contact information, in certain cases users may want to share with others their preferences/interests and context-specific sensor readings like number of steps [42]. A definition is provided next.

Definition 4 (*Shared Profile*) A shared profile is defined as: $XP = (P', S, C)$, where P' is the part of P that will be shared, S is the set of entities (people or organisations) that are granted access to P', and C is the condition under which this access is granted.

Note that in Definition 4, it is not necessarily the case that $P' \subseteq P$, as data may be edited before it is shared, as in editing the complete current location to only show the city. Shared profiles, as such, act as access rights, where the condition C simply specifies under what condition do the entities in S have access to the profile P' . As for notation, we note that in the remainder of this paper, we will use *XD* to refer to the data part of a shared profile and *XN* to refer to the norms of a shared profile.

A shared profile, also referred to as contextual profile in Fig. 2, is created by *visibility and abstraction rules* that we discuss next.

Visibility Rules and Abstraction

A visibility rule determines who can see what and when. For example, I may allow friends to have access to my exact location, while the rest may only have access to the city where I live. These visibility rules help generate the shared profile introduced above. To preserve privacy, and as illustrated in our location sharing example, some transformations can be applied as a set of abstraction mappings as defined in [25]. These abstraction mappings take in input an element of the input theory, in this case the knowledge graph introduced in "Data" section, and produce in output a rewrite of this element which captures the desired information hiding. Formally, these mappings are theory mappings which map a given theory into a new theory satisfying the desired constraints [26]. As discussed in detail in [25], based on the theory in [26], there are only three types of abstraction mappings, defined in terms of they operate on entities, attributes and relations, as follows:

Granularity: the granularity operator allows for substituting object wholes with one of its object parts. This is when one wants to be more specific. The opposite holds when one wants to be more vague or general. For example, as from Fig. 1, we can substitute a whole for a part,

$$\begin{gathered}\text{Granularity(entity=MyHome)}\\ \Rightarrow\\ \text{entity=Bedroom\#1}\end{gathered}$$

or, viceversa, a part with a whole,

$$\begin{gathered}\text{Granularity(Home.Location = Barcelona)}\\ \Rightarrow\\ \text{Home.Location = Catalonia}\end{gathered}$$

In the first case, *MyHome* is substituted with a bedroom inside the house, thus making the information more precise while, vice versa, in the second, *Barcelona* is substituted with *Catalonia*, this making the information more generic and less informative.

Generality: the generality operations allow the folding of concepts, attributes and relations towards more general or more specific notions (making them more implicit or specific, respectively). Thus for instance,

$$\begin{gathered}\text{Generality(concept = father)}\\ \Rightarrow\\ \text{concept = relative}\end{gathered}$$

Partiality: the partiality operation allows for the elimination of entities, attribute values and relation values from the shared profile.

$$\begin{gathered}\text{Partiality(Car\{Color=Red, Brand=Toyota\})}\\ \Rightarrow\\ \text{Car\{Color=Red\}}\end{gathered}$$

The intuition underlying these mappings is to generalise the information content of their input, thus achieving the desired level of privacy. Thus, granularity abstracts a given entity to a more general entity (in the example above, from the city of Barcelona to the region of Catalonia), generality abstracts a concept to a more general concept (in the example above, from the concept of father

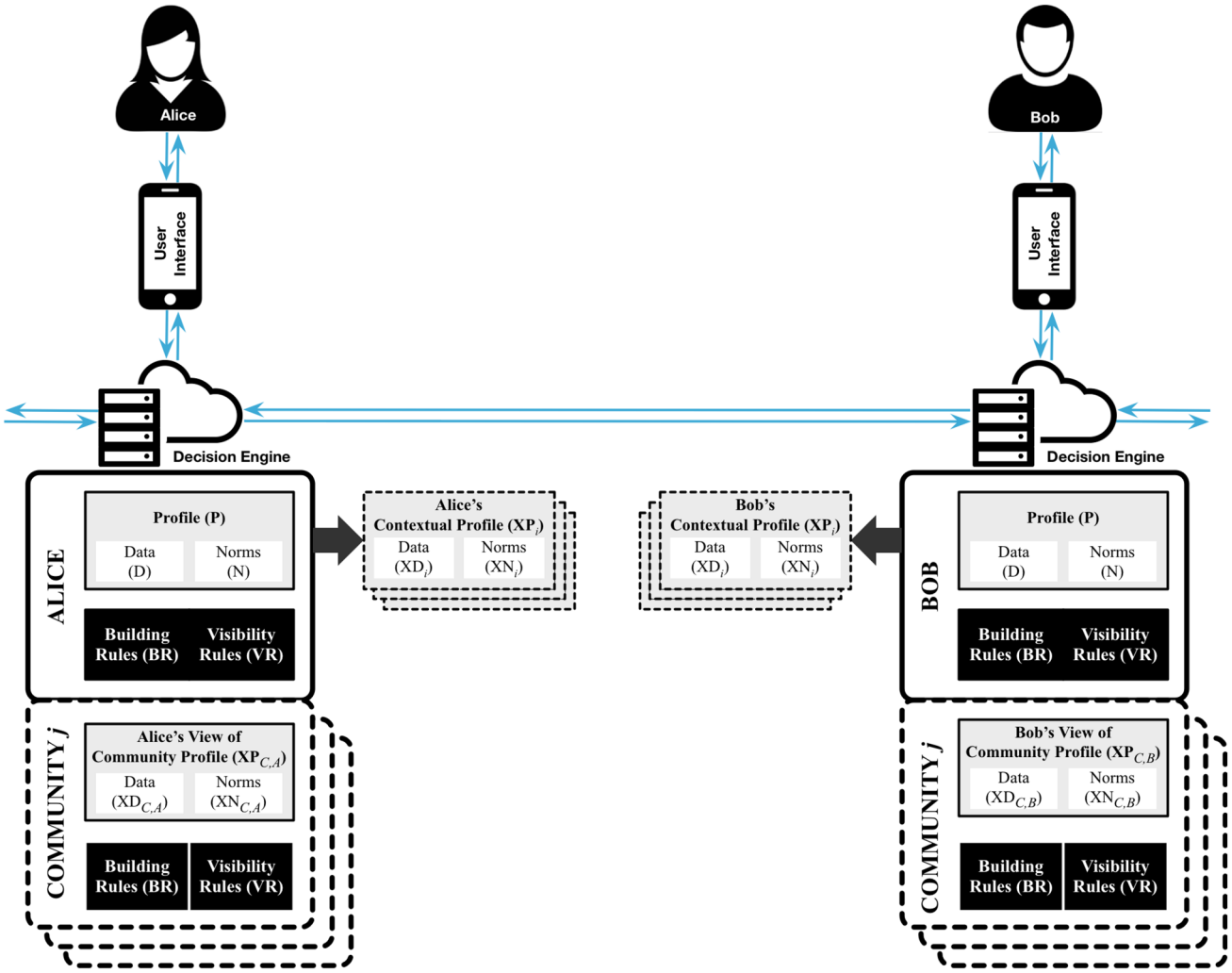

Fig. 3 Basic (decentralised) architecture

to the concept of relative) while, last but least, partiality, the most commonly used mappings, allows to forget some elements of the profile (in the example above, the brand of the car). While these three mappings are built in the system, it is up to the owner of the profile to define what is abstracted into what, for whom, under what conditions, etc.

Architecture and Operational Model

We organise this section in three parts. First, we present how the profile of individuals are organised, then we do the same for communities and, finally, we conclude with a description of the decision engine that is responsible for decision making when it comes to managing behaviour.

The Individual

In Fig. 3, the schema of the peer-to-peer architecture for our proposed normative system is presented. Each individual has a decision engine on her device that represents her and is used for interacting with others. In other words, individuals interact with each other through their decision engines, and their communication with their device's decision engine happen through a user interface. Whether the decision engine runs all or some of its computations locally or on a remote server is an implementation issue that usually depends on the complexity of the norms and their computational requirements.

As explained in the previous section, each individual has a profile, composed of data and norms. Additionally, each individual has building and visibility rules that define how the profile is built and with whom is it shared (and under

what circumstances). As such, the decision engine on an individual's device is represented as having access to these three elements: the building and visibility rules, in addition to the individual's complete profile, which is by default private. We note that each individual has the right to access and edit its own rules and profile.

For the sake of simplicity, we leave the data sources that feed the profiles outside of Fig. 3, as we choose to focus here on what is needed for individuals to interact with one another as opposed to building profiles. When interacting with others, individuals may decide to share some of its profile. Again, as presented in the previous section, these decisions are based on the visibility rules that specify what parts of the profile may be shared with whom and under what circumstances. The result is the decision engine on individual's device creating contextual profiles that share part of the profile in different contexts, sharing it with selected people under certain circumstances. The contextual profile is illustrated in Fig. 3 as XP_i, which is composed of contextual data XD_i and contextual norms XN_i, and where i represents the context. Each individual usually has a number of contextual profiles.

The Community

In addition to the individual's profile and their building/visibility rules, communities also have profiles and building/visibility rules. In other words, a community is just another entity with a profile and building/visibility rules. However, different from community members, we say any behaviour in the community should be aligned with the community norms (whereas one's behaviour does not need to be aligned with another's individual norms). As such, giving a community member access to the community's profile (or at least a selection of that profile that concerns that member's interaction in that community) is essential for ensuring that member's interaction adheres to the community's norms. The community may also provide different shared copies of its profile to different members (the $XCP_{C,i}$ in Fig. 3, where C specifies the community and i specifies the entity that this profile is shared with). For example, a community may share some sensitive profile data with its president, but not with other members. It is the community's visibility rules that will decide what data/norms can be accessed by whom (possibly, including non-members too).

In addition to the visibility rules, building rules are used to clarify who can *edit* the original community profile and how. This is because not all members are equal. Some may be given special rights in a community that allows them to edit data/norms, and sometimes they may even be allowed to edit the rules themselves (building and visibility rules). Though we must note here that data is usually much more accessible for editing than norms, because interactions usually update community's data. For example, with Alice making a new request in the community, the community's total number of requests should automatically get updated by Alice's decision engine (the entity that creates this new request).

We have considered centralised and decentralised approaches for implementing the community profile. In the former case, the community profile is saved in a central location. The verification process that grants access to the community profile adheres with the building and visibility rules is centralised. In the latter case, the community profile is located on those devices that communicate and coordinate their actions using distributed ledger technology [12]. Distributed ledger technologies are based on distributed, decentralized peer-to-peer networks where, unlike distributed databases, there is no need for a central administrator (blockchain is one successful example of distributed ledgers). In this paper, we will assume the decentralised view of the system and will consider that all decisions are local to each device as we illustrate in the following section (and in Fig. 3).

Lastly, we note that one individual may be a member of more than one community, and hence the decision engine on their device will have a number of such community profiles/rules, as illustrated by Fig. 3.

The Decision Engine

Every time individual or community profile and rules are being edited, we need to ensure that these actions abide by the building rules. When an action or event happens in a community (e.g. a message is received from another community member, the individual is asking to perform an action, a deadline has passed, ...), we need to ensure that responding to this action adheres to the given norms. For example, if the user is sending a message to other community members, should this message be forwarded, are there any other computations to be carried out, etc. We refer to the engine that reacts to such actions/events and responds accordingly as the *decision engine*.

The decision engine at each device must have both a reactive and proactive behaviour.

- *Reactive Behaviour* This allows the decision engine to react to messages received (usually representing the actions being performed), and there are two types of messages that a decision engine can receive:
  - *A message from the user interface* When a user performs an action, it is translated into a message that is sent to its decision engine through the user interface.

Upon the receipt of such a message, the decision engine needs to first verify that the message does not violate any of the norms.

If the action does not violate any of those norms, then the decision engine needs to decide what to do next, usually translated into sending messages to other entities. This decision follows from the norms that the engine would have checked, and sometimes taking into account some relevant profile data.

- *A message from another decision engine* As with the previous case, the decision engine needs to first verify that the message does not violate any of the community norms. This re-checking upon receipt ensures that the sender's decision engine has not been manipulated to cheat. If the message violates any of the community norms, then it may either be discarded, or if the community norms require sanctioning, then the appropriate sanctions should be executed.

  However, if the action obeys the community norms, then the decision engine needs to decide what to do next, which is usually translated into sending messages to other entities and/or the user interface. As above, this decision takes into consideration the community and individual norms.

- *Proactive Behaviour* This allows the decision engine to proactively perform actions as required by the norms. For example, incentivising norms might remind a user to complete their profile, if this has been neglected for some time, or remind the user of how much their contribution to their community is valued, if they have not been active lately. A norm suppressing messages when one is sleeping might send these messages when the alarm goes off to wake the user. While external events might trigger reactive behaviour, we argue that internal events trigger proactive behaviour (e.g. reaching a timeout).

The decision engine is triggered when an action is performed, and we view all actions as messages. For example, the user pressing a button is translated into a message from the user to its decision engine. Messages may be of two types, those received by decision engine from its associated user (in other words, one's actions are translated to messages that are sent to the user's own decision engine), and those received by other decision engines. Notice that here we put the restriction that no one can send messages directly to another user's decision engine, before having their message passing through their own decision engine first. Another issue to note is that we assume interactions happen in communities. As such, each message is associated with a given community.

The decision engine may also be triggered when an event happens (e.g. an alarm goes off, or a timeout is reached). In this case, the decision engine will require a list of relevant events that may trigger it and their associated communities (when applicable). For example, an alarm that marks that a community's deadline is near will be associated with that specific community, but an alarm that wakes up a person might not be associated with any community.

Figure 4 illustrates the behaviour of the decision engine. If triggered by receiving a message, it extracts all the norms relevant for that message, that is, the associated community norms, the individual norms associated with the user of this decision engine, and other norms that have been associated with this specific message (e.g. if one wants others to know that she is only looking for people in her vicinity, then this norm gets attached with the message). If triggered by an event, the relevant norms to be checked by the decision engine are then the individual norms associated with decision engine's user, and the norms of the community associated with the event, if any.

After compiling the set of relevant norms, the decision engine checks the norms one by one in order to see assess the consequences with respect to the triggering message or event. For example, does it need to perform some computations? Send some information back to its user? Forward the incoming message (if any) to another decision engine? Set a timer to perform some action at a later time? These consequences are usually specified by the norms. However, after compiling the complete set of consequences, and before executing them, the decision engine needs to make sure that these consequences do not have consequences themselves. As such, it goes into a loop (see the loop in Fig. 4) to check the consequences of the consequences, and will continue to repeat this until there are no new consequences arising. When that is reached, the compiled set of consequences is executed, and the job of dealing with the triggering message or event is done.

Properties of the decision engine. In what follows, we present a few properties of our decision engine's algorithm.

The first property is a property about the decision engine's algorithm itself, namely, its finiteness. Despite having a loop, the algorithm always comes to an end after a finite number of steps.

Property 1 (Finiteness) *The decision engine's algorithm will always terminate after a finite number of steps.*

Proof Sketch For any message/event triggering the decision engine, the decision engine will check the norms one by one and compile a set of consequences C'. The decision engine then loops to check the consequences of C' and so on. The decision engine exits this loop when there are no more new consequences to consider.

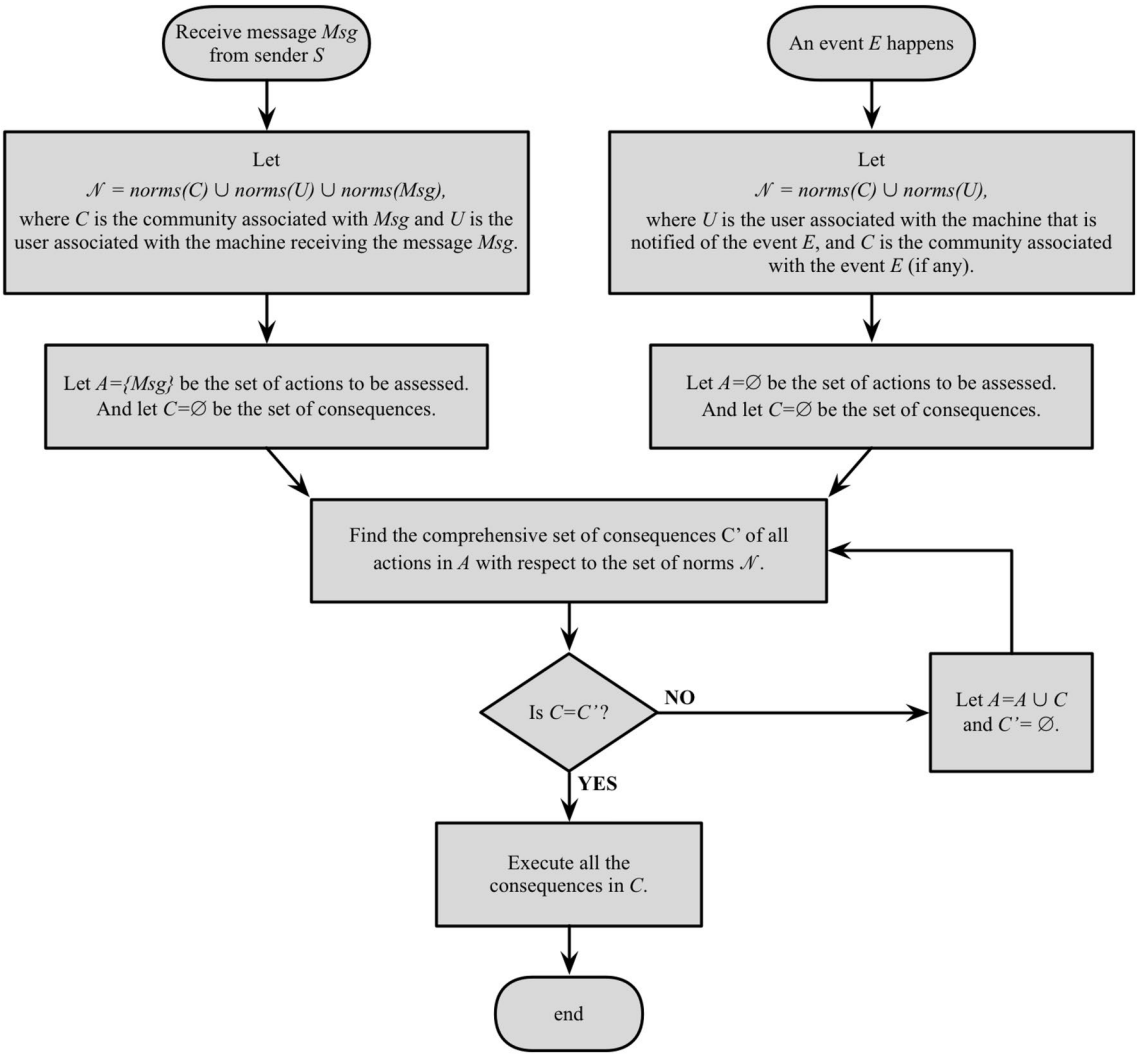

Fig. 4 An illustration of the decision engine's algorithm

As the set of norms is finite, it is then inevitable that the set of consequences will also be finite. With a finite set of consequences, the decision engine is guaranteed to eventually exit this loop and terminate its execution.

The second property is a property about community behaviour. It states that with our proposed normative-based system, norms are a necessary condition for any behaviour to emerge in a community. If the set of norms is empty for a given community, then nothing can happen in that community.

Property 2 (Necessity of Norms) *Norms are a necessary condition for any behaviour to emerge in a community*.

Proof Sketch Following the algorithm of Fig. 4, for every message or event that will trigger the norm engine, the set of norms to be evaluated will be retrieved. Now let us assume that this set of norms is empty: $\mathcal{N} = \emptyset$. The algorithm will try to go through existing norms one by one to check their relevance with respect to the triggering message/event and extract the corresponding consequences when appropriate. However, as the set of norms is empty, then there are no norms to check and the set of consequences will be empty too: $C = \emptyset$. With no consequences, the triggering message/event will result in no actions to be performed. With no actions performed, no behaviour can emerge in the community, regardless of the triggering messages and events.

The third property is about the propagation of messages in a community. It essentially states that as long as the norms require the forwarding of messages to all adjacent nodes, and there are no other norms that condition this

forwarding, then any two nodes connected by a path can send messages to each other.

Property 3 (Reachability) *If there exists a norm that requires sending and forwarding messages to all neighbouring nodes unconditionally, then a message sent by node n_0 can propagate through a path $P = \{n_0, n_1, ..., n_l\}$ of any length $l \in \mathbb{N}^*$.*

Proof Sketch Say there is a norm that states that any message to be sent shall be sent to all neighbouring nodes:

```
IF
    to_send(N,M)
    and neighbours(N,NN)
THEN
    send(N,NN,M)
```

where `to_send(N,M)` states that node `N` wants to send the message `M`, `neighbours(N,NN)` states that the set of all neighbouring nodes of `N` is `NN`, and `send(N,NN,M)` states that a message `M` is to be sent from node `N` to the set of neighbouring nodes `NN`.

Also, say there is a norm that states that any received message is to be forwarded to all neighbouring nodes:

```
IF
    received(N,N',M)
    and neighbours(N',NN)
THEN
    send(N',NN,M)
```

where `received(N,N',M)` states that a message `M` has been received by `N'` from `N`.

And say there exists no other norm that conditions the above behaviour: the behaviour of sending and forwarding messages to all neighbouring nodes.

Now we show that a message sent by n_0 will propagate through a path $P = \{n_0, n_1, ..., n_l\}$ of any length $l \in \mathbb{N}^*$.

First, we note that when node n_0 sends a message, the decision engine of Fig. 4 will send this message all neighbouring nodes of n_0, including node n_1, and that is in accordance with the consequences of the first norm presented above. As such, we show that a message propagates through a path of length 1.

Second, we note that if a message propagates through a path P of length m then it will propagate through a path P of length $m + 1$. This is because if a node n_m receives a message, the decision engine will result in sending the message to all neighbouring nodes of n_m, which include the node n_{m+1}. And that is in accordance with the consequences of the second norm presented above. As such, we show that if a message propagates through a path of length m, then it will propagate through a path of length $m + 1$.

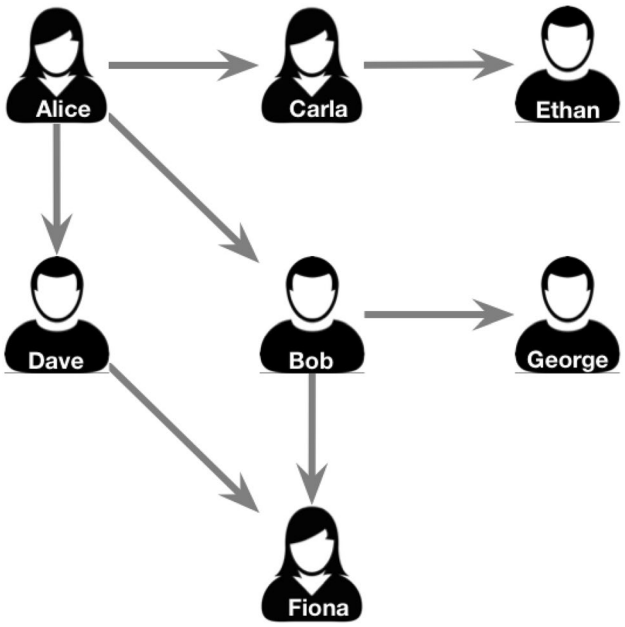

Fig. 5 The social network of the WeNet example

Given that a message is guaranteed to propagate through a path of length 1, and given that if a message propagates through a path of length m then it will propagate through a path of length $m + 1$, by inductions, we can then say that a message can propagate though a path of any length $l \in \mathbb{N}^*$.

There are other properties of interest that we leave for future work. For example, while Property 3 is based on the norm that all nodes will forward a message to all other neighbouring nodes (that is, the probability of forwarding a message to all neighbouring nodes is 1), it would be interesting to show that reachability decreases as the probability of forwarding a message to neighbouring nodes goes below 1. Such a property helps, for example, assess the impact of privacy on reachability. We know that norms usually make heavy use of profile data. As such, the more the data is private, then the less effective the norms can be. And if reachability is affected by such norms, then reachability will certainly decrease with the increase of privacy. The proof of such interesting properties, however, will be experimental proof, where one can make use of simulations to verify the property in question. As mentioned earlier, this is left for future work.

A Motivating Example

In this example, we will specify the interaction between a number of people in an open community of mutual help. The community is inspired with the WeNet use case in mind, an open community allowing one to find help with everyday tasks, such as picking up one's child from school, finding some friends to dine with, or finding some people to play padel with. It finds help by propagating help requests through one's social network. In our specific example, we keep the community's social network very limited for the sake of simplicity. Figure 5 presents the social network associated with our example, where a link from node n to n' represents that n' is a friend of n. The relevant community and individual profiles associated with the nodes of Fig. 5 are presented in Fig. 6.

The community profile that defines a community contains data about the community (Lines 1–5 of Fig. 6), such as the list of members of the community, the list of suspended accounts, etc. The profile also contains the norms that specify the rules of interaction. These rules help shape community behaviour, and in our example, they attempt to increase collaboration. The first rule, or community norm (Lines 6–13), restricts continuous requests for help if the requester has not been volunteering himself (it essentially does not permit 5 consecutive requests without making any offer to help). The second community norm (Lines 14–18) enforces a strict penalty on volunteers that commit to helping others and then fail to go through with their commitment, by suspending their participation in the community for 24 h (and suspended accounts cannot make new requests for help: Lines 19–22).

In addition to community norms that govern community behaviour, individuals may also have their own norms, also saved as part of their profiles. For example, Alice has a norm that states that only people closeby (in the same city) may receive her requests (Lines 34–37). Ethan has a private norm that states that notifications are to be suppressed when he is napping (Lines 55–58). Fiona, a professional padel player, has a private norm that states that requests to play padel are to be ignored if they come from novice players (Lines 64–67). In addition to individual norms, profiles hold data about the user, like their current location, their competency in padel, whether they own a car or not, etc. Not all individual profiles specify all data attributes, this is up to the user to decide what to save in their profile. It is also up to the user to decide what profile information to share and with whom. To keep the example simple, we keep the visibility rules out of Fig. 6, and we present the shared data and norms.

In this example, Alice is sharing her location, her level at playing padel (novice), and her norm that requires that only people in the same city may receive her request (Lines 38–43). Notice that we use 'all' in the notation for shared norms or data (e.g. $XN_{alice,all}$ or $XD_{alice,all}$) to state that this part of the profile is to be shared with everyone. Bob, Ethan, Fiona and George are sharing the city where they are located, which is extracted from their private location, and whether they have a car or not (Lines 46-47, 59–60, 68–69, and 72–73, respectively). Carla is sharing her location city (Line 49). Dave is not sharing any information about himself.

Now say Alice is looking for someone to play padel with tonight. For this specific request, she might add an additional norm, such as they must have a car to drop her off later at night. This additional norm is shared with everyone in the specific context of this request (see $XN_{alice,all,requestId}$ on lines 75–79 of Fig. 6). The norm essentially states that whoever receives a request from Alice, they should not be notified about the request if they do not have a car.

Alice will attempt to send her request on the WeNet platform, with the new request-related norm embedded. The objective of WeNet is to start propagating her request within her social network, starting from her friends, to her friends of friends, and so on. This is achieved with each decision engine that receives the request, starting with her own decision engine, deciding whether it needs to send its user a notification about this request or not, and whether it should forward it to friends or not. Decisions of a decision engine are made by checking community norms, the individual norms of the associated user, the requester's shared norms, and any request-related norm associated with the specific request.

The steps for propagating the request and finding a volunteer is described next by the reaction of the different decision engines at the different stages of the request propagation. Note that the interaction here is asynchronous. Of course, *some* actions will happen in a specific order. For example, Alice's decision engine must first kick of the propagation of the request before other decision engines can start receiving messages and reacting to them. Or Ethan's decision engine must first receive the request from Carla's before it can react to it. However, we do not know for sure whether Carla's decision engine will receive the request before Dave's or Fiona's, for example. As such, the steps below describing the reaction of the different decision engines does not have a specific order (keeping in mind, of course, that a decision engine must receive a message before reacting to it).

– *Alice's decision engine—initiating the propagation:* Receiving Alice's request, Alice's decision engine first checks whether it violates any community norms. As the number of consecutive requests made since Alice's last offer for help is 3 (Line 25), it fulfils the first community norm (Lines 6–13) and does not break any other com-

```
1  D_community = { members(alice, bob, carla, dave, ethan, fiona);
2               suspended{};
3               consecutive_requests(alice,3);
4               consecutive_requests(bob,5);
5               ...   }
6  N_community = {IF
7                   attempt_new_request(Requester,Request,RRNorms) and consecutive_requests(Requester,X) and X<5
8               THEN
9                   new_request(Requester,Request,RRNorms);
10              IF
11                  attempt_new_request(Requester,_,_) and consecutive_requests(X) and X=5
12              THEN
13                  message(Requester, "You may not request help as you first need to offer help.");
14              IF
15                  committed_to_offer_help(X,Request) and failed(X,Request)
16              THEN
17                  suspend(X) and message(Requester, "Your account is suspended for 24 hours because you failed to fulfil your
18                                                    commitment.");
19              IF
20                  attempt_new_request(Requester,_,_) and suspended(Requester,true)
21              THEN
22                  message(Requester, "You may not make new requests as your account has been suspended for 24 hours.")   }
23 XD_community,alice = {
24              suspended(alice,false);
25              consecutive_requests(alice,3);
26              ...   }
27 XD_community,bob = {
28              suspended(bob,false);
29              consecutive_requests(bob,5);
30              ... }
31 ...
32 D_alice = {   location("Calle Enric Granados 15, 08008 Barcelona");
33              competency(padel,novice)  }
34 N_alice = {   IF
35                  new_request(alice,Request,_) and city_location(alice,City)
36              THEN
37                  friends_in_city(City,Friends) and forward_request(alice,Request,Friends)   }
38 XD_alice,all = {location(alice, "Barcelona"};
39              competency(alice,padel,novice)   }
40 XN_alice,all = {IF
41                  new_request(alice,Request,_) and city_location(alice,City)
42              THEN
43                  friends_in_city(City,Friends) and forward_request(alice,Request,Friends)   }
44 D_bob = {    location("Carrer de Verdi, 32, 08012 Barcelona");
45              has(car,true)   }
46 XD_bob,all = { location(bob, "Barcelona");
47              has(bob,car,true)   }
48 D_carla = {   location("Passeig de Gràcia, 43, 08007 Barcelona")   }
49 XD_carla,all = {location(carla, "Barcelona")   }
50 D_dave = {    location("Pg. de Sant Joan, 152, 08037 Barcelona");
51              has(car,true)   }
52 D_ethan = {   location("Carrer de Balmes, 197, 08006 Barcelona");
53              has(car,true);
54              competency(padel,professional)   }
55 N_ethan = {   IF
56                  naptime(true) and notify(Request)
57              THEN
58                  supress_notification(Request)   }
59 XD_ethan,all = {location(ethan,"Carrer de Balmes, 08006 Barcelona");
60              has(ethan,car,true)   }
61 D_fiona = {   location("Av. de Sarrià, 45, 08029 Barcelona");
62              has(car,true);
63              competency(padel,intermediate)   }
64 N_fiona = {   IF
65                  new_request(Requester,Request) and request_type(play_padel) and ¬ competency(Requester,padel,professional)
66              THEN
67                  ¬ notify(fiona,Request)   }
68 XD_fiona,all = {location(fiona, "08029 Barcelona");
69              has(fiona,car,true)   }
70 D_george = {  location("9 Bywater St, London SW3 4XD, UK");
71              has(car,true)   }
72 XD_george,all = {location(george, "London"),
73              has(george,car,true)   }
74
75 XN_alice,all,requestId = {
76              IF
77                  receive_request(alice,R) and ¬ has(X,car)
78              THEN
79                  ¬ notify(X,R)  }
```

Fig. 6 WeNet example: individual and community profiles (data and norms)

munity norm (for instance, she is not breaking the norm on Lines 19–22, because her account is not suspended —Line 24). Individual and request-related norms are also not broken. As such, Alice's decision engine decides to propagate the message to her friends Bob, Carla and Dave.

- *Carla's decision engine:* Checking the relevant norms, Carla's decision engine decides that Carla should not be notified of the request, and simply forwards the message to Carla's friends (in this simplified network, just Ethan). This is because Carla does not have any information on whether she owns a car or not, and the request-related norm requires the recipient of the request to own a car.
- *Dave's decision engine:* Dave's decision engine also suppresses sending the notification to Dave and simply forwards the request to Dave's friends (in this simplified network, Fiona). This is because Dave's location and car ownership are kept private when there is a shared norm from the requester (Alice) requiring Dave to be in Barcelona and a request-related norm requiring Dave to have a car.
- *Bob's decision engine:* Unlike Carla and Dave, Bob fulfils all requirements. He is in the same city as Alice (Barcelona) and has a car. As such, he receives a notification about Alice's request, and his decision engine forwards the request to his friends (in this simplified network, Fiona and George).
- *Fiona's decision engine:* Fiona's decision engine receives the request from both Bob and Dave's decision engines, but Fiona has a private norm that ignores invitations to play padel if they come from novice players. As Alice's shared profile with Fiona states that she is novice at padel, Fiona's decision engine does not send a notification about Alice's request to Fiona.
- *George's decision engine:* George's decision engine receives the request from Bob, but as George is currently in London, his decision engine does not send him the notification about Alice's request (as it break's Alice's shared norm that requires being in the same city as Alice).
- *Ethan's decision engine:* Ethan's decision engine receives the request from Carla's, but a notification to Ethan is momentarily suppressed as Ethan is taking a nap and he has a private norm that requires suppressing notifications when napping. Finally, when Ethan wakes up from his nap, he receives Alice's request and he accepts.
- *Alice's decision engine—finding a volunteer:* Alice's decision engine receives Ethan's acceptance, and it decides to notify Alice about this. Alice now has one volunteer to play padel with that fulfils her requirements.

For the sake of simplicity, the reader will notice that the example has been extremely simplified. For example, we do not explain how the predicate "suspend(X)" (Line 17) manages the list of suspended accounts (Line 2), or how the predicate "city_location(_)" (Line 35) extracts the city from a given location. The objective of this example is to illustrate how our proposed system ensures the interaction between people adheres to both community norms and individual ones without jeopardising people's privacy. It also illustrates the impact of private and public information (whether it was concerning data or norms) on both local and external decisions processes. For instance, we note that private individual information (data or norms) are better suited to control local behaviour, whereas shared individual information are better suited for controlling the behaviour (or decision process) on others' decision engines. For example, to see how shared norms can have an impact on other decision engines: notice that all decision engineers are aware of Alice's norm of restricting notifications to those in the same city, but only the impact of a private norm is local to the decision engine of the private norm's owner only. For example, Fiona's private norm filters the notifications sent to Fiona concerning padel requests to those that come from professional padel players. No one needs to know Fiona's restriction. And if a requester does not share their expertise on padel with Fiona, then their request will never get to Fiona, without them being aware of this.

Also note that for privacy reasons, not sharing some information assumes that the information does not exist. For instance, Dave fulfils Alice's requirements as he has a car and he is in the same city. And Dave's decision engine is fully capable of confirming this as it has access to his private data. But by notifying Dave of Alice's request, Alice can automatically deduce that Dave is in the same city (if he accepts). And as such, Dave's privacy concerning his location would be broken. For this reason, Dave's decision engine assumes that private data are not used for actions that have implications outside Dave's decision engine.

And again for privacy reasons, the community only shares the account suspension information with the account holder only: each community member can only know whether their own account has been suspended or not.

Related Work

The main idea that our proposed architecture is built upon is the profile, which is composed of the more traditional data element, as well as the more novel norms element. As such, in this section, we present the related work in the fields of both profiles and normative systems.

The issue of how to define meaningful profiles has been extensively studied in various sub-fields of Artificial Intelligence, see e.g. [29, 56]. Two are the main differences with the notion of profile presented here. The first is that, in all

this previous work, what is being profiled is the *user* while in this work, we profile *people*, i.e., their overall behaviour, independently of whether this behavior involves machines. As a matter of fact, the profile in "Profile" section is mainly focused on people's everyday life properties. The second is that, in all the previous work, the profile is built by the system, largely independently from the user, for instance, in order to provide the most relevant product [40, 44]. In the work described here, the profile is built under the total control of the entity being profiled, and with the goal, not to enable a better system behaviour, but rather to be applied by other people, as a key ingredient for enabling better social interactions. It is important to point out also that some of the approaches used to represent and manage the knowledge in the profiles are based on semantic web techniques [8, 16, 23].

The notion of profile presented here shares some basic principles with the work on *contextual privacy* [5, 46]. In this work, agents are associated with a set of attributes which describe them, i.e., their *profile*. Key elements of profiles are *roles*, namely properties that characterise the way something, e.g., an agent, participates in some course of action. In this work, agents may hold multiple roles in parallel and usually hold them for some limited amount of time; thus, for instance, an agent can be at the same time a doctor and the recipient of a message. As from [46], agents interact via *communication actions* where each communication action consists of a sender, a recipient and a message, and the *context* of a communication action is the sets of roles that the involved agents have in that communication actions. Many of the ideas are common: we both profile people, rather just users, and we both have the idea of having the profile, in our case, the public profile, defined in terms of the current interaction context. We also have a notion of role, where we take roles as described in [30, 43], which seems very similar to their notion. The key difference is that the work described in [5, 46] is foundational and focused on the basic principles while here we propose an approach that uses context-driven profiles to adapt what is shared to the different contexts.

The notion of (lifelong) management of personal data is discussed in [52]. This work, which is rather general and focused on basic principles, provides useful guidelines for how to store, maintain and use personal data. Of specific interest is the notion of *partial identity*, where a partial identity is the description of a person within a certain (situational) context. Thus, a person may have a partial identity at work, another when shopping, another when in vacation and so on. Furthermore, these partial identities evolve and change in time following the dynamics of the life of a person. Many of the long-term issues described in this work (e.g., the minimisation of data made available to third parties) are implemented in the private profile, as implemented inside iLog, and also via the implementation of the public profile. As a matter of fact our notion of public profile can be seen as an implementation of the idea of partial identity. Related and motivated by the ideas in [52] is the work on *PPL*, for *Primelife Policy Language* [2, 58]. The idea of norms which can be circulated together with data and which can be used to define how these data should be used maps directly to the PPL notion of *sticky policies*. With respect to the general idea of sticky policies, the type of norms that we have considered in this paper are limited to the management and circulation of the personal profile.

Another important aspect of our profiles is their inclusions of norms, allowing the proposed system to act as a normative system. Normative systems have attracted considerable attention in the multi-agent systems community as one approach to maintaining the autonomy of agents while ensuring community goals and aspirations are fulfilled. Relevant work in this field is the work on electronic institutions [16] that help organise collective activities by restricting interactions to abide by some established conventions (which may be understood as norms). While normative systems have excelled at addressing issues such as coordination and cooperation [1], they have left a number of open challenges. In this paper, we deal with two such issues which are crucial in the design of open communities. The *first* is how to reconcile individual goals with community goals. A number of approaches have been studied to take the individual into consideration, such as norm synthesis techniques that would help norms evolve based on individuals' behaviour [45], or norm evolution that would allow the individuals to reason about norms through argumentation [47]. But what about individual norms that one is not willing to share with their fellow community member? The *second* is that in such an open environment the individual privacy should be protected. For instance one would not want to let others know of the blatant contradiction between community and individual norms.

Also relevant to our work is the work in agent-based simulations [41], where a theory of agent behaviour for specific contexts is needed to model agent behaviour. While behavioural models are usually used to model agents, normative models may also be used for developing a heuristic model of behaviour. However, like others, these do not provide solutions to the two issues we raise above.

Past, Current and Future Work

This paper has proposed a decentralised architecture for normative systems that introduces individual norms, while ensuring the privacy of people. These ideas and architecture, including the decision engine of "The decision engine" section are being developed, and continuously evolving, as part of the WeNet project. The implementation of WeNet's

platform is based on the adaptation and integration of two pre-existing systems. The first is the uHelp app [37], which provides the mechanism for matching and connecting suitable people, and is currently being modified to mediate community interactions through norms. The second is the *iLog system* [64], the core of the private profile component, as originally specified in [23, 24, 57]. Considerable effort has been devoted to the development of techniques for learning profile data from sensor data and human-machine interactions, and this has implemented as part of the work described in [27, 57, 63].

Our current next steps are an extension of the existing WeNet platform aimed at introducing different types of norms and corresponding different types of profiles. In fact, as illustrated above, norms can be used to specify the rules of interaction in a community, but also to introduce more specialised rules, such as rules specifying what is considered ethical and unethical, or rules specifying how to motivate people to act in a certain way. Working on incentives and linking them with norms is an ongoing work, which we hope to report on next.

One aspect that has not been analysed in this paper and left for future work is the conflict resolution mechanism. Having people specify their own norms will probably result in conflicting rules, and a mechanism will be needed to address such conflicts.

Last, but not least, we have illustrated with the example of "A motivating example" section the impact of sharing (or not) data and norms. Our next steps include plans to formally explore the properties of our proposed system, especially when it comes to understanding private versus shared profile data and norms.

Acknowledgements This research is being funded by the European Union Horizon 2020 FET Proactive project "WeNet - the Internet of Us", grant agreement Number 823783 (https://www.internetofus.eu/). WeNet also provided the main motivations and problem scenarios at the basis of this work. We thank all our WeNet colleagues for the many insightful discussions on the general notion of social relations. Special thanks to Ivano Bison, Matteo Busso, Daniele Miorandi and Marcelo Dario Rodas Britez.

Funding Open access funding provided by Università degli Studi di Trento within the CRUI-CARE Agreement.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Andrighetto G, Governatori G, Noriega P, van der Torre LWN, editors. Normative multi-agent systems. vol. 4. Dagstuhl Publishing; 2013.
2. Angulo J, Fischer-Hübner S, Pulls T, Wästlund E. Towards usable privacy policy display & management-the primelife approach. In: HAISA. 2011. p. 108–18.
3. Arnaboldi V, Conti M, Passarella A, Pezzoni F. Analysis of ego network structure in online social networks. In: 2012 international conference on privacy, security, risk and trust and 2012 international confernece on social computing. 2012. p. 31–40.
4. Baader F, Horrocks I, Sattler U. Description logics. In: Handbook on ontologies. Springer; 2004. p. 3–28.
5. Barth A, Datta A, Mitchell JC, Nissenbaum H. Privacy and contextual integrity: framework and applications. In: 2006 IEEE symposium on security and privacy (S&P'06). IEEE; 2006. p. 15.
6. Bilogrevic I, Huguenin K, Agir B, Jadliwala M, Hubaux JP. Adaptive information-sharing for privacy-aware mobile social networks. In: Proceedings of the 2013 ACM international joint conference on pervasive and ubiquitous computing, UbiComp '13, New York, NY, USA. 2013. Association for Computing Machinery; p. 657–66.
7. Blum M, Pentland A, Troster G. Insense: interest-based life logging. IEEE Multimed. 2006;13(4):40–8.
8. Bonatti PA, Decker S, Polleres A, Presutti V. Knowledge graphs: new directions for knowledge representation on the semantic web (dagstuhl seminar 18371). Dagstuhl Rep. 2018;8:29–111.
9. Bontempelli A, Teso S, Giunchiglia F, Passerini A. Learning in the wild with incremental skeptical gaussian processes. In: Proceedings of the 29th international joint conference on artificial intelligence (IJCAI). 2020.
10. Bouquet P, Giunchiglia F. Reasoning about theory adequacy. A new solution to the qualification problem. Fundamenta Informaticae. 1995;23(2, 3, 4):247–62, .
11. Brough M, Literat I, Ikin A. "Good social media?": underrepresented youth perspectives on the ethical and equitable design of social media platforms. Soc. Media + Soc. 2020;6(2):205630512092848.
12. Burkhardt D, Werling M, Lasi H. Distributed ledger. In: 2018 IEEE international conference on engineering, technology and innovation (ICE/ITMC). IEEE; 2018. p. 1–9.
13. Chittaranjan G, Blom J, Gatica-Perez D. Mining large-scale smartphone data for personality studies. Pers. Ubiquitous Comput. 2013;17(3):433–50.
14. Cummings JN, Butler BS, Kraut RE. The quality of online social relationships. Commun ACM. 2002;45(7):103–8.
15. d'Inverno M, Luck M, Noriega P, Rodriguez-Aguilar JA, Sierra C. Communicating open systems. Artif Intell. 2012;186:38–94.
16. Ehrlinger L, Wöß W. Towards a definition of knowledge graphs. In: SEMANTiCS (Posters, Demos, SuCCESS). 2016.
17. European Commission. White paper on artificial intelligence: a European approach to excellence and trust. 2020.
18. European Parliament. General data protection regulation (regulation (eu) 2016/679): legislative act. 2016.

19. Faraj S, Jarvenpaa SL, Majchrzak A. Knowledge collaboration in online communities. Organ Sci. 2011;22(5):1224–39.
20. Fernback J. Beyond the diluted community concept: a symbolic interactionist perspective on online social relations. New Media Soc. 2007;9(1):49–69.
21. Giunchiglia F, Maltese V. Domains and context: first steps towards managing diversity in knowledge. J Web Semant. 2012;12:53–63.
22. Giunchiglia F. A diversity-aware internet, when technology works for people, 2020. https://ec.europa.eu/digital-single-market/en/news/diversity-aware-internet-when-technology-works-people.
23. Giunchiglia F, Bignotti E, Zeni M. Human-like context sensing for robot surveillance. Int J Seman Comput. 2017;12(01):129–48.
24. Giunchiglia F, Bignotti E, Zeni M. Personal context modelling and annotation. In: 2017 IEEE international conference on pervasive computing and communications workshops (PerCom Workshops). IEEE; 2017. p. 117–22.
25. Giunchiglia F, Fumagalli M. Teleologies: objects, actions and functions. In: ER—international conference on conceptual modeling. ICCM; 2017. p. 520–34.
26. Giunchiglia F, Walsh T. A theory of abstraction. Artif Intell. 1992;57(2–3):323–89.
27. Giunchiglia F, Zeni M, Big E. Personal context recognition via reliable human-machine collaboration. In: 2018 IEEE international conference on pervasive computing and communications workshops (PerCom Workshops). IEEE; 2018. p. 379–84.
28. Giunchiglia F, Zeni M, Gobbi E, Bignotti E, Bison I. Mobile social media usage and academic performance. Comput Hum Behav. 2018;82:177–85.
29. Golemati M, Katifori A, Vassilakis C, Lepouras G, Halatsis C. Creating an ontology for the user profile: method and applications. In: Research challenges in information science (RCIS 2007). 2007. p. 407–12.
30. Guarino N, Welty C. Evaluating ontological decisions with ontoclean. Commun ACM. 2002;45(2):61–5.
31. Gurrin C, Smeaton AF, Doherty AR. Lifelogging: personal big data. Found. Trends Inf. Retr. 2014;8(1):1–125.
32. Hartswood M, Jirotka M, Chenu-Abente R, Hume A, Giunchiglia F. Privacy for peer profiling in collective adaptive systems. In: Privacy and identity management for the future internet in the age of globalisation. 2015. p. 237–52.
33. Hoffmann T. The meanings of competency. J Eur Ind Train. 1999.
34. Hogan A, Blomqvist E, Cochez M, d'Amato C, de Melo G, Gutierrez C, Gayo JE, Kirrane S, Neumaier S, Polleres A, Navigli R. Knowledge graphs. 2020.
35. Houghton DJ, Joinson AN. Privacy, social network sites, and social relations. J Technol Hum Serv. 2010;28(1–2):74–94.
36. Jiang Z, Heng CS, Choi BC. Research note—privacy concerns and privacy-protective behavior in synchronous online social interactions. Inf Syst Res. 2013;24(3):579–95.
37. Koster A, Madrenas J, Osman N, Schorlemmer M, Sabater-Mir J, Sierra C, Fabregues A, De Jonge D, Puyol-Gruart J, Garcia-Calvés P. u-Help: supporting helpful communities with information technology. In: Proceedings of the 2013 international conference on autonomous agents and multi-agent systems, AAMAS '13, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems; p. 1109–10.
38. McAuley JJ, Leskovec J. Learning to discover social circles in ego networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, editors. Advances in neural information processing systems, vol. 25. Curran Associates, Inc., 2012. p. 539–47.
39. Levin S. Facebook told advertisers it can identify teens feeling 'insecure' and 'worthless'. Guardian. 2017.
40. Logesh R, Subramaniyaswamy V, Vijayakumar V. A personalised travel recommender system utilising social network profile and accurate GPS data. Electron Govern Int J. 2018;14(1):90–113.
41. Macal CM, North MJ. Tutorial on agent-based modelling and simulation. J Simul. 2010;4(3):151–62.
42. Martinez-Villaseñor MDL, Gonzalez-Mendoza M, Hernandez-Gress N. Towards a ubiquitous user model for profile sharing and reuse. Sensors. 2012;12(10):13249–83.
43. Masolo C, Vieu L, Bottazzi E, Catenacci C, Ferrario R, Gangemi A, Guarino N. Social roles and their descriptions. In: KR. 2004. p. 267–77.
44. Mishra V, Arya P, Dixit M. Improving mobile search through location based context and personalization. In: 2012 international conference on communication systems and network technologies. 2012. p. 392–6.
45. Morales J, López-Sánchez M, Esteva M. Using experience to generate new regulations. In: Proc. of IJCAI '11. 2011. p. 307–12.
46. Nissenbaum H. Privacy as contextual integrity. Wash. L. Rev. 2004;79:119.
47. Oren N, Luck M, Miles S, Norman TJ. An argumentation inspired heuristic for resolving normative conflict. Unknown Publisher. 2008.
48. Osman N, Sierra C, Chenu-Abente R, Qiang S, Giunchiglia F. Open social systems. In: 17th European conference on multi-agent systems (EUMAS), Thessaloniki, Greece, 2020.
49. Pan JZ, OWL Working Group. Owl 2 web ontology language document overview: W3c recommendation 27 October 2009, 2009.
50. Parks MR, Roberts LD. Making moosic: the development of personal relationships online and a comparison to their offline counterparts. Soc Pers Relatsh. 1998;15:517–37.
51. Perry BL, Pescosolido BA, Borgatti SP. Egocentric network analysis: Foundations, methods, and models, vol. 44. Cambridge University Press; 2018.
52. Pfitzmann A, Borcea-Pfitzmann K. Lifelong privacy: privacy and identity management for life. In: IFIP primelife international summer school on privacy and identity management for life. Springer; 2009. p. 1–17.
53. Plant R. Online communities. Technol Soc. 2004;26(1):51–65.
54. Scekic O, Miorandi D, Schiavinotto T, Diochnos DI, Hume A, Chenu-Abente R, Truong HL, Rovatsos M, Carreras I, Dustdar S, Giunchiglia F. Smartsociety—a platform for collaborative people-machine computation. In: SOCA. 2015.
55. Schelenz L, Reinhardt K, Gjuraj D. Developing a conceptual and ethical framework of diversity: deliverable 9.1 for the project wenet-the internet of us. 2019.
56. Schiaffino S, Amandi A. Intelligent user profiling. In: Bramer M, editor. Artificial intelligence an international perspective: an international perspective. Berlin: Springer; 2009. p. 193–216.
57. Shen Q, Teso S, Zhang W, Xu H, Giunchiglia F. Multi-modal subjective context modelling and recognition. In: 2020 international workshop modelling and reasoning in context (ECAI Workshops). 2020.
58. Trabelsi S, Sendor J, Reinicke S. Ppl: primelife privacy policy engine. In: 2011 IEEE international symposium on policies for distributed systems and networks. IEEE; 2011. p. 184–5.
59. Tufekci Z. Facebook's surveillance machine. New York Times. 2018.
60. Verhagen HA, Neumann MA, Singh MP. Normative multiagent systems: foundations and history. In: Chopra A, van der Torre L, Verhagen H, Villata S, editors. Handbook of normative multiagent systems. College Publications; 2018. p. 3–25.
61. Wasserman S, Faust K, et al. Social network analysis: methods and applications, vol. 8. Cambridge: Cambridge University Press; 1994.
62. Xu W, Zhou X, Li L. Inferring privacy information via social relations. In: IEEE international conference on data engineering workshop. 2008.

63. Zeni M, Bignotti E, Giunchiglia F. Combining crowdsourcing and crowdsensing to infer the spatial context. In: 2018 IEEE international conference on pervasive computing and communications workshops (PerCom Workshops). IEEE; 2018. p. 28–33.
64. Zeni M, Zaihrayeu I, Giunchiglia F. Multi-device activity logging. In: ACM international joint conference on pervasive and ubiquitous computing. ACM; 2014. p. 299–302.
65. Zeni M, Zhang W, Bignotti E, Passerini A, Giunchiglia F. Fixing mislabeling by human annotators leveraging conflict resolution and prior knowledge. Proc ACM Interact Mobile Wearable Ubiquitous Technol. 2019;3(1):32.