Patterns of Inquiry in a Community Forum for Legal Compliance with Privacy Law

Sarah Santos*, Sara Haghighi†, Sepideh Ghanavati*, Travis D. Breaux†, Thomas B. Norton‡
*Software and Societal Systems Department, Carnegie Mellon University, Pittsburgh, Pennsylvania
{ssantos, tdbreaux}@andrew.cmu.edu
†School of Computing and Information Science, The University of Maine, Orono, Maine
{sara.haghighi, sepideh.ghanavati}@maine.edu
‡Center on Law and Information Policy, Fordham University School of Law, New York, New York
{tnorton1}@law.fordham.edu

***Abstract*—Software developers have variable access to legal advice depending on the size and maturity of their organization. To supplement direct access to legal counsel, some developers seek advice through public forums, such as Reddit. This paper presents an exploratory case study to analyze questions posted on Reddit that concern compliance with the EU General Data Protection Regulation (GDPR). We analyze seven posts, selected using stratified sampling, then extract and code facts within the post to discover a taxonomy of discourse patterns, which describes how developers reason over legal privacy requirements. We report results on types of facts, their conclusory statements, and the main focus areas of developers when seeking legal advice.**

***Index Terms*—requirements, legal compliance, forums**

I. Introduction

Software developers must comply with various regulations, such as the EU General Data Protection Regulation (GDPR) [30]. In large organizations, developers work within a privacy program managed by in-house legal counsel that helps them assess privacy and legal risk or implement code design reviews to ensure compliance. Small organizations, including start-ups, often rely on hired outside counsel to set up their privacy posture (e.g., write privacy notices, recommend third parties, etc.). In this setting, access to knowledge and best practices can be inconsistent and incomplete, leaving developers to seek advice on public forums [9], [27], [29], [39].

In this paper, we study discussions on the Reddit forum r/GDPR to examine how developers talk about ambiguous legal requirements [2], [26]. These forums show how interactive discussions among developers and subject matter experts can help developers build their understanding of how to interpret legal terminology, what legal requirements mean in practice, and how to avoid non-compliant edge cases in design. In general, we focus on posts by those who are likely developers or who are asking software development-related questions, with a specific design description in their posts. The discourse patterns offer insight into the information needs of developers and the structure of arguments in support of or against compliance claims. Using stratified sampling, we randomly select seven posts from the r/GDPR subreddit, which cover a range of GDPR topics including consent, exemptions, retention, de-identification, user access to their data, and third-party practices to conduct our case study analysis. We extract facts presented in Reddit discourse and code them with a grounded theory approach. Our findings present taxonomies of fact types and systems and an analysis of how facts are organized to understand legal compliance.

The results show that, on average, conclusory statements make up ∼18.9% of a Reddit post's summary, suggesting that developers provide minimal conclusory inferences about "why" and "how" their software could satisfy a requirement. Instead of inferential statements, developers focus on providing descriptive premises, which is ∼81.1% of a typical post's summary. 30.4% of facts are about the interface between software and its broader context, 34.2% are solely about software design, and 35.4% are purely about the wider world, including the legal context.

II. Background and Related Work

A. Grounded Theory

Grounded theory is an inductive research process in which theory emerges from the analysis of data [38]. In this research approach, analysts often code a dataset, a process called *coding* [32], and then interpret the codes to create a new artifact (a theory) that describes or explains the data [38]. In requirements engineering, this method has been used to discover requirements artifacts that generalize and originate in the data, such as personas [19] and use cases [43].

Alternatively, grounded theory has been used to construct models or taxonomies that explain phenomena observed in the data [3], [7]. In these examples, the theory aims to generally describe or explain how phenomena produce engineering behavior, whether this covers the communication methods of developers [7] or the authorship of legal requirements [3].

In software engineering, Stol et al. argue that grounded theory methods should meet specific reporting standards [37]. In this paper, we employ *open coding* [32], a commonly used technique in conjunction with developing a grounded theory. Similar to [3], [7], we are interested in understanding the concepts and relations that describe patterns in a dataset. This effort is exploratory and does not lead to a substantive theory.

B. Satisfiability Arguments

The notion of satisfiability has been applied to requirements engineering through prior work on formal methods and model checking of formalized requirements. Cimatti et al. [6] studied automatically verifying requirements using satisfiability procedures from satisfiability modulo theory. Zahid et al. [44] surveyed various studies of formal methods for requirements in industrial-control systems. Other research efforts focus on translating natural language requirements into formal logic [4].

Goals, which describe requirements to be achieved, maintained, or avoided [8], can be satisfied using formal arguments. In goal modeling, satisfaction is realized through logical AND/OR refinement and goal decomposition links to sub-goals, and through obstacles and their countermeasures [22], quantitatively and qualitatively [1], [25]. A goal's satisfiability is calculated from the total or partial satisfiability of sub-goals, which relies on various numerical probabilities and tolerances. Differences in satisfiability can be used to prioritize among multiple design alternatives. Van Zee et al. use argument schemes to capture stakeholder discussions about the acceptability of goal model elements [45]. Building on this work, we explore *legal* requirements' satisfiability for software.

Departing from formal methods to compute satisfiability, we aim to understand informal, natural language arguments for or against the satisfiability of legal requirements. In this respect, stakeholders must assess whether a particular design description satisfies the intent of a legal requirement. We draw on the study of logical arguments in fields like philosophy, law, and linguistics. Toulmin [40] defines three essential elements of an argument: 1) claim, a central assertion the argument aims to prove, 2) data or evidence, a supporting reason backing a claim, and 3) warrant, an assumption linking the data to the claim. Searle [34] offers additional nuance to Toulmin's data with his distinction of brute versus institutional facts: brute facts exist independently of any social institution (e.g., the Earth is 93 million miles from the Sun), whereas institutional facts rely on some human context or social reality (e.g., she is a student at Carnegie Mellon University). This distinction of fact types adds additional properties to the supporting evidence in Toulmin's argument scheme. We build on these ideas when analyzing the reasoning roles of sentences in developer discussions. We combine Toulmin's framework with concepts from legal reasoning, namely the use of premises and conclusions for legal arguments [24].

We also draw on discourse relations from linguistics, which provide a framework to analyze sentence composition and dependencies between sentences [35]. We are specifically interested in logical dependencies between sentences assembled to argue for a claim of legal satisfiability. We aim to annotate units of discourse, sourced from a corpus of developer discussions, with reasoning properties rooted in patterns of argument from Toulmin [40], Searle [34], and Levi [24].

III. Method and Approach

We employ a case study research design that is both exploratory to answer "what" questions and explanatory to answer "how" questions [42]. Our research questions are:

RQ1: What types of information do developers provide when asking questions about compliance on online forums?

RQ2: How do developers arrange this information to structure an argument for or against requirements satisfiability?

A. The Case: r/GDPR

We select the r/GDPR forum on Reddit, which is highly active (i.e., over 52 million active daily users in 2019 [20]), covers a diverse range of GDPR topics, and presents rich descriptions of software features under GDPR. The platform invites contributors to ask and answer a range of questions about GDPR, from how to handle edge cases in data subject access requests [14], to the legality of scraping anonymous public data [15], to learning general best practices for data privacy [11]. These discussions are technical in both legal and software domains, providing insight into how GDPR concepts are and should be implemented in practice. Contributors include data controllers and processors, which the GDPR [30] defines, respectively, as the persons who determine "the purposes and means of the processing of personal data" and who process "personal data on behalf of the controller." The data controller may or may not be the party collecting the data from a data subject, whereas the data processor is often the organization representing the developer who is seeking clarification about whether an envisioned or implemented feature is GDPR compliant.

As an example, one post [13] with a title, "Is this legal?", contains a picture of a paywall banner that raises legal concerns. The banner appears after a user rejects cookies and only allows access if the user either pays a fee or registers an account with only five days of cookie-free access. GDPR requires consent to be freely given, which means there is no external pressure compelling a user to give consent [12]. These pressures can include a range of negative consequences for refusing consent, e.g., a power imbalance, detriment, conditionality, etc. [10]. The paywall banner's restrictions highlight potential detriments, e.g., a financial burden via the fee, which may prevent freely given consent. The example illustrates either a gray area or a potential non-compliant situation requiring further legal analysis. Developers can use this legal analysis to inform design decisions affecting a feature's legal compliance.

B. Data Preparation

We selected seven posts by different authors from r/GDPR that were posted between March and August 2023. The posts were selected using stratified sampling from GDPR-related topics: consent, de-identification, retention, data subject access, disclosure to third parties, and legal exemptions.

Reddit users exhibit varying stylistic differences in tone and style. To standardize the tone and style and generate text with simpler, more atomic statements, we summarize both the title and body for each post using `gpt-4-0613` with default parameter settings (temperature= 0.7), a system message of "*You are a helpful assistant.*", and this prompt template:

```
Please summarize the following scenario:

{post_title} {post_body}
```

The summaries normalize differences in point-of-view to a third-person voice rather than a first-person voice specific to a role (e.g., a web developer versus a website owner). They also help reduce various emotive phrases, which are typically irrelevant to evaluating factual evidence for legal compliance (e.g., "I'm sorry if this might seem prosaic...", "I'm not entirely clear...", "guidance would be much appreciated...", etc.).

Here is an original post with its generated summary. Note how the summary is more concise, has a neutral tone, and standardizes the point of view to third-person voice:

***Example** 1 **(Generated summary sample)**:*
Original Reddit Post

```
Hello everyone
I need some help with GDPR compliance for my
website. Here's the situation: my website is
hosted in Europe and it contains a third-party
integration with LaunchDarkly, a company based
outside of Europe. While the data sent to
LaunchDarkly does not include any personal
information, users' browsers still establishes a
 connection to their servers, which could
potentially reveal IP addresses.
As the website owner, I'm wondering if I have
any obligation to obscure these IP addresses,
even though I don't process or store them. I'm
not entirely clear on what GDPR requires in this
 situation, so any advice or guidance would be
much appreciated. Thanks in advance!
```

Generated Summary

```
The website owner is seeking help regarding GDPR
 compliance for their website. The website
integrates with a third-party company called
LaunchDarkly, which is based outside of Europe.
Although the data sent to LaunchDarkly does not
contain personal information, users' browsers
establish a connection to their servers,
potentially revealing IP addresses. The website
owner wants to know if they are obligated to
hide these IP addresses, even though they do not
 process or store them. They are unsure about
the requirements of GDPR in this situation and
are seeking advice and guidance.
```

The seven generated summaries served as our target dataset for manual analysis, available *in this public workbook*.

C. Analysis Method

Our method aims to discover the components that comprise a *satisfiability argument*, which we define as a unit of logical discourse that presents evidence, as premises, to support a conclusion about whether a software system does or does not satisfy some legal requirements. Our analytical objective is to discover patterns of constructing arguments for or against a claim about whether a software feature satisfies a legal requirement. The method proceeds in two steps: (1) we extract atomic facts from Reddit post summaries; (2) we open code these facts based on their role in supporting an argument.

1) Extracting Facts: For each sentence in a generated summary, we manually extract *facts*, where a fact is an atomic simple sentence with a single subject, verb, and (optional) object. A complex or compound sentence may include additional clauses, which we itemize into separate facts during this analytical step. Note that facts are not necessarily descriptive truths, but comprise a mix of indicative statements about "what is" and optative statements about "what should be" [18]. We do not exclude facts present in the source sentence, even if they are not about the software's design or legal context.

A single sentence in a summary can yield one or more facts. For example, consider the following sentence from which we extract three facts, $F1$, $F2$, and $F3$:

***Example** 2 **(Sample source sentence)**:*

```
    The website owner is seeking help regarding
GDPR compliance for their website.
```

Facts:

- **F1** There is a WEBSITE.
- **F2** WEBSITE is owned by an entity, OWNER.
- **F3** OWNER wants help with GDPR compliance for WEBSITE.

During fact extraction, we also extract *variables*, which are the nominal agents involved in data processing or controlling, typically expressed as a noun or noun-phrase. Example 2 shows the extracted variables, WEBSITE and OWNER.

2) Coding Facts: We employed open coding [32] to create a shallow taxonomy of the fact types expressed by Reddit users. The taxonomy aims to clarify what phrases constitute the minimal unit of information needed to construct compliance questions and answers, and how those units compose to answer compliance questions. The coded units are then analyzed in a second step to identify patterns of facts.

For each extracted fact, we begin by coding the *fact type*, which indicates its role in a satisfiability argument. We began without an initial set of codes and iteratively reviewed the list of facts to discover these types. When we discovered a new type, we iteratively reviewed the existing fact types to ensure there were no overlaps. Where an overlap was possible, we reviewed the coded facts to decide if the previously identified facts should be re-coded using the newly discovered type. After coding 48 facts, the list of fact types had saturated, i.e., every fact subsequently reviewed fit one of the previously identified fact types, and no new fact types were discovered. We report the list of fact types in Section IV.

Next, we performed a second, higher-level round of analysis to code *reasoning properties* of fact types, which describe a type's role in logical reasoning. We define *reasoning properties* as any feature of a fact type that describes how that type is used in or appears in an argument. We identify two specific reasoning properties for a fact type: 1) its reasoning role, and 2) its presentation style. A type's *reasoning role* captures how the type is used in an argument. It is the logical function that a fact type plays in an argument, such as to provide evidence or infer a conclusory statement from provided evidence. Reasoning roles are determined by a fact's dependencies and relations to other facts in a given argument. A type's *presentation style* describes how the type appears in an argument, such as explicitly stated or implicitly deduced.

When coding reasoning properties, we applied these to fact types, rather than individual facts themselves. Our goal was to identify broader reasoning patterns that remained consistent at a higher level of abstraction above individual facts. We then arranged fact types as "leaf" nodes in a taxonomic structure, wherein parent nodes represent reasoning properties inherited by child nodes. Each reasoning property corresponds to a layer in the taxonomy (see Figure 1).

Finally, we performed a third round of analysis at the fact level to tag each fact with its relevant *system*, i.e., the physical or conceptual entity that a fact belongs to. We then arranged the possible systems in a taxonomic structure, which is separate but parallel to the taxonomy of reasoning properties. Figure 2 in Section IV shows the taxonomy.

IV. Results

We now present our results, including the taxonomy of fact types and their frequencies, and observed discourse patterns.

A. Taxonomy of Fact Types

The coding process described in Section III-C2 yielded six atomic types of facts (i.e., the "leaf" nodes of our taxonomy of reasoning properties). Each fact type is defined in part by a combination of higher-level reasoning properties. We first formalize the principles of reasoning properties and then apply those properties to define the fact types.

Our taxonomy organizes each fact type under two top-level classes of reasoning roles: *premise* and *conclusion*. Drawing from Levi's work on legal reasoning [24], as well as argument patterns on the LSAT [16], these categories serve our goal of understanding developers' reasoning patterns to construct satisfiability arguments. We define a *premise* as a descriptive statement taken to be true and that is used as evidence to support a claim or contextualize a query, i.e., to substantiate a *conclusion*. For example, the following fact is a premise: *"OWNER does not process nor store IP_ADDRS."* This premise is used to contextualize the following query, which forms a "conclusion" of the source summary: "*Must OWNER obscure IP_ADDRS?*"

We define a *conclusion* as an inferential statement whose logical validity is not presumed or known and that is supported or contextualized by other facts (premises). An example of a conclusion in the form of a claim is: "*COMPANY claims it has immunity from GDPR, given Fact 26.*" Conclusions can also appear in the form of a query: "*How can REQUESTS processing be simplified?*", or as implications derived from other premises, such as: *"APP is not developed by a company."* This implication follows from two premises: *"APP is developed by a group of individuals, DEVS."*, and *"DEVS are not a company."*. During the coding process, we identify premises that directly support each conclusion fact.

Note that conclusions can support other conclusions. In other words, intermediate conclusions act as premise inputs for "downstream" conclusions. Formally, the conclusion class can have a recursive relationship to itself. In our analysis, we found that implications are most commonly intermediate conclusions, serving as evidence for some final claim or query.

We further organize facts into two presentation styles: *explicit* facts appear directly in the source sentence; and *implicit* facts are derived by using either 1) logical deduction for *implications*, or 2) critical analysis for *assumptions*, to identify missing information needed to justify why premises are relevant to and sufficient in supporting a conclusion [40]. During the coding process, we identify explicit facts that directly support each implied fact. Finally, the leaves show the six atomic fact types, which we define as follows:

1) ***Fixed Fact***: An indicative statement in the source text that is taken to be true in the scope of a satisfiability argument.
   - E.g.: "*There is a person entity, PERSON, who works with COMPANY.* "
2) ***Hypothetical Fact***: A descriptive statement explicitly stated in the text but, unlike a fixed fact, qualified with a modal verb, such as "may", "might", etc. These modal verbs represent multiple possible states for a given system.
   - E.g.: "*The databases may be stored in the US.*"
3) ***Implication***: An implied descriptive statement not appearing in the source text but that is logically derived from explicitly stated facts.
   - E.g.: "*APP is based in the EU.*"
   - Given: "*APP is developed by a group of individuals, DEVS.*" and "*DEVS are all based in the EU.*"
4) ***Assumption***: An implied descriptive statement not appearing in the source text that provides a missing detail required to connect a fixed or hypothetical fact to the argument's larger claim. It is analogous to a *warrant* in Toumlin argument structures [40].
   - E.g.: "*3RD_PARTY_DATA is sent by WEBSITE.*"
   - Given: "*There is a WEBSITE.*", "*WEBSITE contains a third-party integration with LaunchDarkly, 3RD_PARTY_LD.*", and "*Data, 3RD_PARTY_DATA, is sent to 3RD_PARTY_LD.*"
5) ***Claim***: A conclusory, inferential statement in the source text whose logical validity relies on support from premise facts.
   - E.g.: "*COMPANY claims it has immunity from GDPR, given Fact 26.*"
   - Given: "*COMPANY is involved in crime prevention in various centers.*"
6) ***Query***: A question statement in the source text to inquire about an unknown claim.
   - E.g.: "*Must the BANNER include a 'decline' button?*"

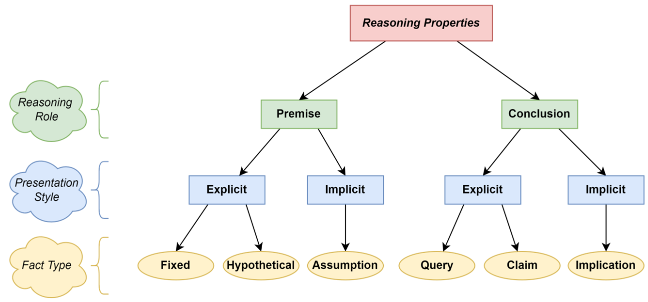

Fig. 1. Taxonomy of Reasoning Properties

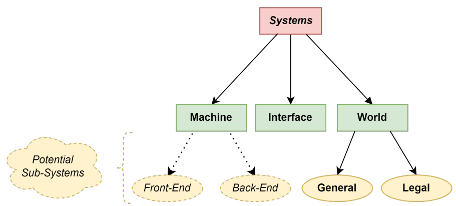

Fig. 2. Taxonomy of Systems

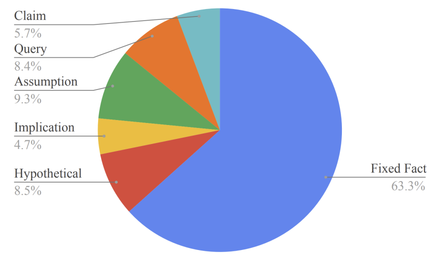

Fig. 3. Average fact type composition in a Reddit post summary

In parallel to reasoning properties, we observe that facts can be organized into three systems: the *machine*, which directly relates to a software system; the *world*, which concerns the environment into which the software is situated, including the broader legal context; and the *interface*, which captures interactions between the software and its larger world environment. These systems are modeled on the foundational distinctions introduced by Jackson [18]. The *interface* system refers to the collection and sharing of data about the world, e.g., personal data, but not its processing, which occurs within the *machine*.

During open coding, we identified one sub-system under *world* – the *legal* system, which we denote as *world::legal*. *world::legal* applies to any fact whose purpose is compliance related, such as understanding whether certain actions are compliant with a particular regulation. We present examples for each system below:

- **machine**: "*STORE uses Google Tag Manager and Google Analytics 4 to track data.*"
- **world::general**: "*Inaccurate tracking may cause ineffective ads.*"
- **world::legal**: "*Is it legal to send TOKEN from the EU to the US under GDPR?*"
- **interface**: "*SAAS collects personal data, DATA.*"

Our two taxonomies define the top-level classes from which facts inherit characteristics, where a characteristic is some reasoning property or system. Our taxonomy is visually represented as a rooted tree, where each node is a class. Directed edges between class nodes point to sub-classes. During open coding, each fact is assigned one fact type class and one system class, such that the characteristics of these classes also apply to the given fact.

For example, take the following fact: "*CONNECTION can reveal user IP addresses, IP_ADDRS.*". We assign it the fact type of *hypothetical fact* under the *interface* system. The two classes of hypothetical fact and interface produce a combined sub-class of a *hypothetical interface fact*. Given our taxonomies, this fact inherits the reasoning properties of *explicit* and *premise*, while possessing *interface* characteristics about interactions between software and the wider world.

B. Fact Type Frequencies

1) Reasoning Property Frequencies: In our analysis, we found that developers provide minimal conclusory inferences about "why" and "how" their software could satisfy a requirement. Instead of inferential, interpretive statements (e.g., conclusion facts), developers focus on providing descriptive information (e.g., premise facts). Premises make up ∼81.1% of a typical post summary, whereas conclusion facts are only ∼18.9% of an average summary. Figure 3 breaks down frequencies by fact type for a typical post summary.

2) System Frequencies: We notice that ∼34.2% of all facts are about software design (*machine*), and ∼35.4% are about the wider world (*world*), which includes GDPR and the legal context (∼13.9% of total fact types). ∼30.4% of all facts are about the *interface* between software (machine) and its broader context (world). The *interface* facts capture the boundary between legal and software systems. Separating this interface boundary into its own system, distinct from pure legal (world) and software (machine) systems, allows us to analyze how fact topics flow from one system to another. We discuss examples of how facts flow through systems in Section V. Figure 4 shows frequencies of systems by fact type and reasoning role.

System	Fact Type						Reasoning Role	
	Fixed Fact	*Hypothetical Fact*	*Assumption*	*Implication*	*Query*	*Claim*	*Premise*	*Conclusion*
Machine	40.8%	0.0%	44.4%	25.0%	40.0%	0.0%	*36.8%*	*18.2%*
Interface	36.7%	66.7%	11.1%	0.0%	20.0%	0.0%	*33.8%*	*9.1%*
World::General	14.3%	33.3%	44.4%	75.0%	20.0%	0.0%	*23.5%*	*9.1%*
World::Legal	8.2%	0.0%	0.0%	0.0%	20.0%	100.0%	*5.9%*	*63.6%*

Fig. 4. System frequency by fact type

C. Fact Coordination Patterns

We describe how facts are arranged in a satisfiability argument, focusing on how premises are used to draw conclusions. We visualize a satisfiability argument as an *argument graph*, where each node is a variable, or a descriptive feature about a variable, presented by some fact. We identify "edges" between variables and feature "nodes". These edge patterns represent common relations that connect fact types, reasoning properties, and systems.

1) Shared Variables Connect Premises: This edge pattern connects variables from premise facts with a shared variable. The edge is defined by a common variable, which refers to a noun phrase entity, that is shared by multiple premise facts.

The edge pattern works as follows: An initial premise $declaration$ declares some variable X. Subsequent premises, $subsequent_i$, share X and add additional descriptive features about X. This information further restricts what real-world entity X can refer to, akin to a restrictive relative clause in linguistics. A relative clause is defined as a dependent clause sharing a noun phrase with the main independent clause [21]. A restrictive relative clause adds information about the shared noun phrase that limits the meaning of that noun phrase [21]. The edge introduced here is functionally similar to a relative clause, so we call this edge type a *relative clause edge*.

Here is a basic demonstration of relative clause edges, building on Example 1 from Section III:

1) **declaration** = *"There is a WEBSITE."*
2) **subsequent_1** = *"WEBSITE is hosted in Europe."*
3) **subsequent_2** = *"WEBSITE is owned by an entity, OWNER."*

2) Conclusions Connect Unshared Variables: The next edge pattern connects variables from premise facts without any explicitly shared variable. The edge is notable since it also has a reasoning role, namely: conclusions. We found that conclusions (claims, queries, and implications) operate as explicit edges tying together entities, and features about entities, that were introduced in other, disparate facts.

The edge pattern works as follows: Various variables $A, B, ..., Z$ are declared then described by a series of $descriptor^i_{XY}$ premises, which is the ith premise about variables X and Y. For example, we extract the premise: *"There is a person entity, PERSON, who works with COMPANY."* In this case, $X =$ PERSON and $Y =$ COMPANY.

There are implicit edges connecting entities from premises with a shared variable (relative clause edges). For instance, there are relative clause edges between $descriptor^1_{XY}$, $descriptor^2_{XZ}$, and $descriptor^i_{XZ}$, which all share variable X.

However, there are initially no edges connecting entities from premises without a shared variable. E.g., some $descriptor^i_{XY}$ is not connected to some $descriptor^i_{AZ}$.

A conclusion fact, $conclude_{XA}$, is introduced that references multiple variables, X and A, from unconnected facts. This $conclude_{XA}$ introduces a new edge between X and A with additional features that X introduces for A, or vice versa.

As a concrete example, take this conclusion (a claim), $claim\_company\_exempt$: *"COMPANY claims that EXEMPTS applies to itself."* In our argument graph, this claim appears as an edge connecting the entities for COMPANY and EXEMPTS, which are two variables introduced in earlier premises, namely: *"There is a COMPANY."* and *"There are exemptions, EXEMPTS, to UK GDPR related to Crime and Taxation."* The claim adds potentially new information about *COMPANY* by relating it to *EXEMPTS*, and vice versa.

3) T-Shaped Network of Edges: We can visually plot the ways that edges sharpen the contour of the "variable topology" by using a T-shaped diagram, which we call argument graphs. Relative clause edges typically add *depth* to a variable in a given system (Figure 5). Conclusion edges often add *breadth* across variables in different systems (Figure 6). We define *depth* as additional knowledge about a variable within a single system, whereas *breadth* is additional knowledge about relations between different systems. We discuss the significance of these diagrams in Section V-D below.

Our T-shaped argument graphs resemble a transposed version of Rosch's vertical and horizontal structures for categories [31]. To use Rosch's terms, high-level basic objects are plotted as breadth in our graph, whereas levels of abstraction for basic objects are plotted as depth. A variable's depth, which is produced through relative clause edges that add descriptive features about a variable, echoes Tversky's definition of objects as collections of compiled features [41]. These variable features are extracted during fact extraction and compiled through relative clause edges, deepening the level of abstraction for that variable.

V. DISCUSSION

A. Missing Information Can Inform Interactive Discourse

Implied facts highlight overlooked premises omitted by a developer but that may be necessary to fully support a conclusion. As described in Section IV, we identified two types of *implicit* facts: *assumptions* and *implications*. These facts do not explicitly appear in the original text and represent unstated premises or valid logical deductions, respectively.

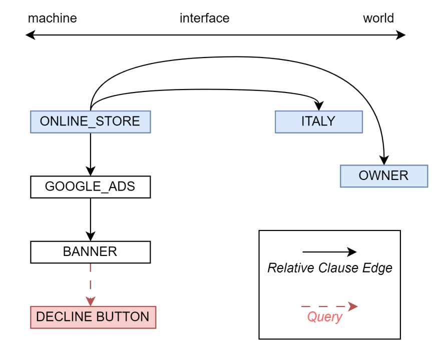

Fig. 5. Relative Clause Edges

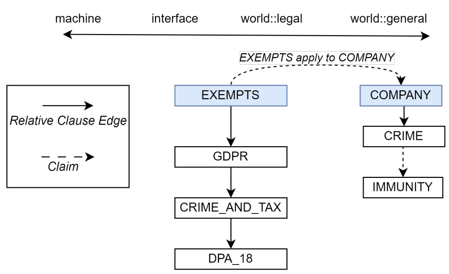

Fig. 6. Conclusion Edges

For instance, in Example 2, we have the source sentence: *"The website owner is seeking help regarding GDPR compliance for their website."* We identified one assumption from this: *"WEBSITE is hosted in Europe."* This assumption is an obvious but necessary premise for the applicability of GDPR. Otherwise, without this assumption, the other facts that relate to GDPR may be irrelevant or insufficient in supporting the argument's claim or query. Similarly, implications may provide crucial logical premises for evaluating requirements' satisfiability. Because these two types are unstated in the original summary, they are the research analyst's interpretations of the summarized posts. The analysts' ability to infer or deduce the existence of implicit facts can vary due to differences in training or other experience in formal logic and reasoning.

We find that implicit facts are occasionally uncovered in a Reddit post through engagement with responders. The Reddit data extends beyond the original author's post to include responses by responders. These responders, like research analysts, may have variable levels of expertise: some responders in our dataset describe themselves as being former data protection authorities (experts), while others may be fellow developers offering their own opinions. Implicit facts could be important sources of interrogation to assess whether the posters or respondents also made these assumptions or held these implications. By extending our method and taxonomy to facts unearthed in the comments, we can better understand how to prompt developers for missing information.

B. Hypothetical Reasoning

Hypothetical facts can help pinpoint design decisions that alter the legal compliance of a software system. A hypothetical fact presents possible alternatives. Each alternative offers a different potential premise with different implications for a satisfiability argument. These alternatives parallel the speculative reasoning that a legal analyst may do to evaluate the compliance risks of software systems. For example, we identified the following hypothetical fact, HF:

***Example** 3 (Hypothetical fact, HF):*

```
    DEVS may place databases in the EU.

```

When constructing a satisfiability argument given HF, a legal analyst may branch the argument into two separate analyses: one where HF is true (databases are in the EU), and another analysis where HF is false (databases are not in the EU). This branch analysis is an exploration of the design state space, where different regions in the space have different legal implications. Hypothetical facts can inform how to prompt developers and legal experts alike for information needed to accurately traverse the design and legal state space.

Hypothetical facts also relate to the idea of missing information introduced above, in the context of assumptions and implications. All three fact types are opportunities to further engage discussants in question-answering to explore the design state space. All three types have an element of uncertainty that must be clarified through further refinement from other facts, or through exhaustive evaluation of all possibilities. Missing information and implied facts present future opportunities to engage discussants in question-answering (Q+A), potentially in future work on AI-powered Q+A tools.

C. Gap between Premise and Conclusion

We identify a key pattern of flawed logic when coordinating facts in a satisfiability argument. Ultimately, we want to know what evidence is needed to answer a query or test a claim. We also want to identify how much of this necessary evidence is missing from the provided premises. We equate the missing evidence with a logical flaw. The missing evidence creates a gap between the premises and conclusions such that the premises do not sufficiently connect to conclusions. Thus, the premises fail to answer a query or support a claim through logical reasoning.

The system of a premise vs that of a conclusion can influence the relevance and effectiveness of premises. For example, in one post summary, the first five premises establish a design description of an online store in Italy that uses Google Analytics. The main query is about whether a decline button is needed to satisfy cookie consent requirements. To answer this question, we need evidence connecting the premises, which are in the machine and interface systems, to some world::legal claim in the same system as the conclusion. However, the poster attempts to answer this query with several facts in the world::general system, which assert that declining cookies leads to wasted money. These facts fail to close the gap between the world::legal query and machine/interface premises.

As a result, the premises do not answer the query. Argument graphs can help identify these gaps in logical reasoning.

D. Inflection Points in Argument Graphs

Argument graphs, introduced in Section IV-C3, can help a developer make design decisions in two main ways: evaluating argument strength and finding inflection points.

Firstly, the graph may illuminate the strengths or weaknesses of a developer's satisfiability argument. For instance, the analysis may yield a diagram with an incorrect conclusion edge between some variable X and some feature A. This conclusion edge represents the developer's understanding of the requirement and the software system. However, that understanding may be flawed, as described in Section V-C. The visual representation of this flawed reasoning, expressed as an incorrect edge, reduces time spent clarifying relationships expressed in textual descriptions.

Besides argument evaluation, the graph can assist in identifying "inflection points", which are certain pivotal facts about software design that trigger non-compliance. For instance, take the following hypothetical fact: "DEVS may place databases in the EU." This is an inflection point that could decide whether international transfers take place, which would trigger various GDPR requirements for data transfers to third countries [30].

An inflection point may also produce contradictory conclusions for fulfilling a legal requirement. In one example, we have several contradictory claims about whether a company is exempt from certain GDPR requirements: *"COMPANY claims that EXEMPTS applies to itself."* vs *"PERSON claims EXEMPTS is for law enforcement."* and *"COMPANY is not law enforcement."* This inflection point raises design decisions about what the company's software can and cannot legally do if the company is not exempt. Namely, the company's software may not be able to perform certain functionalities if the exemptions do not apply.

E. Trace Accountability through Systems

An argument graph allows us to visually trace the inflection points where stakeholders in different systems are accountable for legal compliance. Analyzing the flow of facts through systems can help organizations identify which parties, e.g., developers or lawyers, are responsible for inflection points that trigger non-compliance. This can pinpoint accountability for who should resolve issues in legal requirements satisfiability.

F. Threats to Validity

Construct validity is the correctness of operational measures used to collect data, build theory, and report findings from the data [42], and the extent to which an observed measurement fits a theoretical construct [36]. In this study, we examined Reddit posts to develop a taxonomy of fact types that contribute to a legal argument. The taxonomy concepts saturated after coding 48 facts, which shows the reliability and minimality of the taxonomy to cover the space of fact types within the post. Because the labels are assigned by a single analyst, a threat to validity is whether other analysts would assign the same label to the same statement or phrase.

To mitigate this threat, we conducted a study to measure the inter-rater reliability of the labeling task by three analysts. All analysts specialize in computer science research. Our initial Fleiss Kappa values are: 0.65 for fact types, 0.21 for systems, and 0.49 for reasoning roles. After this round of coding, the analysts met to discuss and improve the code definitions; then they re-coded the data. This resulted in improved Kappa values for all categories: 0.72 for fact types, 0.47 for systems, and 0.61 for reasoning roles. Using Landis and Koch's interpretation scale [23], we report substantial agreement for fact types and reasoning roles and moderate agreement for systems.

Internal validity is the extent to which measured variables cause observable effects in the data [42]. The patterns identified from the fact types rely on a two-step process: itemizing the facts, and labeling the facts.

External validity determines the scope of environmental phenomena or domain boundaries to which the theory and findings generalize [42]. This study examined only seven Reddit posts, while there are hundreds of such posts to analyze. Therefore, the results are only limited to this dataset, and analysis of additional posts is needed to improve generalizability. Our stratified sampling process relied on opportunistic decisions to pick Reddit posts with a variety of question topics. We selected these from a pool of $\sim$50 posts published in the same time window. Extending this window may improve generalizability. Lastly, our methodology is time-intensive and spans multiple steps, some of which require manual interpretation. These factors pose a threat to scaling our approach. Finding fact coordination patterns may prove particularly difficult to automate. This analysis involved reasoning over discourse-level linguistic units, which is an open challenge in structured commonsense reasoning tasks for NLP [46].

VI. Conclusion

Our case study provides a taxonomy for annotating reasoning properties in developer discourse regarding software and legal requirements. Additional studies are needed to extend the annotated dataset beyond a single case study. Ultimately, these empirical findings can inform the design of chat-based interfaces in Q+A tools to check requirements satisfiability. By annotating Reddit discussions using grounded theory, we aim to understand the underlying reasoning process used by developers to assess the legal compliance of their software.

More generally, our taxonomy of fact types can be used for prompt engineering in AI systems that check legal compliance. Future studies can explore optimal arrangements of fact types in a prompt, as a way to maximize the accuracy of language models when evaluating legal requirements satisfiability. Although the manual interpretations of our approach may prove difficult to automate, our study can serve as a catalyst for fostering meaningful discourse about GDPR requirements.

Acknowledgment

This research was funded by NSF Awards #2007298, #2217572, #2217573, and NSF CAREER #2238047.

References

[1] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, E. Yu, "Evaluating goal models within the goal-oriented requirement language," *International Journal of Intelligent Systems, Special Issue:Goal-driven Requirements Engineering*, 25(8): 841-877, 2020.

[2] J. Bhatia, T. D. Breaux, J. R. Reidenberg and T. B. Norton, "A Theory of Vagueness and Privacy Risk Perception," 2016 IEEE 24th International Requirements Engineering Conference (RE), Beijing, China, 2016.

[3] T.D. Breaux, M.W. Vail, A.I. Anton. "Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations." *14th IEEE International Requirements Engineering Conference*, pp. 49-58, 2006.

[4] I. Buzhinsky. Formalization of natural language requirements into temporal logics: a survey. 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), 1, 400-406. 2019.

[5] California Consumer Privacy Act of 2018, Cal. Legis. Serv, 2018.

[6] A. Cimatti, M. Roveri, A. Susi, and S. Tonetta. "Formalization and Validation of Safety-Critical Requirements". *Electronic Proceedings in Theoretical Computer Science*, 20:68–75, Mar. 2010. Open Publishing Association.

[7] D. E. Damian, D. Zowghi, "The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization," *IEEE Joint International Conference on Requirements Engineering*, pp. 319-328, 2002.

[8] A. Dardenne, A. Van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Science of Computer Programming*, 20(1): 3–50, 1993.

[9] Z. Delile, S. Radel, J. Godinez, G. Engstrom, T. Brucker, K. Young, S. Ghanavati, "Evaluating Privacy Questions From Stack Overflow: Can ChatGPT Compete?",. In 2023 IEEE 31st International Requirements Engineering Conference Workshops (REW) (pp. 239-244). IEEE, 2023, September.

[10] EDPB, 'Guidelines 05/2020 on consent under Regulation 2016/679', 4 May 2020 (Version 1.1)

[11] https://www.reddit.com/r/gdpr/

[12] GDPR, Chapter 2, Article 7

[13] https://www.reddit.com/r/gdpr/comments/1bfgnmt/is_this_legal/

[14] https://www.reddit.com/r/gdpr/comments/14kmiuj/making_a_request_for_data_that_i_can_access_on_my/

[15] https://www.reddit.com/r/gdpr/comments/1bmosqu/scraping_involving_public_pii_data_which_are/

[16] https://www.thinkinglsat.com/articles/argument-parts-and-indicators

[17] R. Hoda, "Socio-Technical Grounded Theory for Software Engineering," *IEEE Transactions on Software Engineering*, 48(10): 3808-3832, 2022.

[18] M. Jackson. "The world and the machine," 17th *International Conference on Software Engineering*, pp. 283-292, 1995.

[19] S. Jantunen, D.C. Gause, "Using a grounded theory approach for exploring software product management challenges," Science of Computer Programming, 152: 1-37, 2008.

[20] J. Kastrenakes. (December 1, 2020). "Reddit reveals daily active user count for the first time: 52 million". The Verge.

[21] E. Keenan, Relative Clauses, In T. Shopen (Ed.), Language Typology and Syntactic Description (Vol. 2, pp. 141-170), Cambridge: Cambridge University Press, 1985.

[22] A. van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications,* Wiley, 2009.

[23] J. R. Landis and G. G. Koch, "The Measurement of Observer Agreement for Categorical Data," Biometrics, vol. 33, no. 1, pp. 159–174, 1977.

[24] Levi, E. H. "An Introduction to Legal Reasoning," University of Chicago Law Review: Vol. 15: Iss. 3, Article 2, 1948.

[25] E. Letier, A. van Lamsweerde. "Reasoning about partial goal satisfaction for requirements and design engineering." *12th ACM SIGSOFT International Symposium on Foundations of Software Engineering,* pp. 53–62, 2004.

[26] A. K. Massey, R. L. Rutledge, A. I. Antón and P. P. Swire, "Identifying and classifying ambiguity for regulatory requirements," 2014 IEEE 22nd International Requirements Engineering Conference (RE), Karlskrona, Sweden, 2014.

[27] M. Prybylo, S. Haghighi, S. T. Peddinti, S. Ghanavati, " Evaluating Privacy Perceptions, Experience, and Behavior of Software Development Teams", 2024, https://doi.org/10.48550/arXiv.2404.01283.

[28] L. Michaelis. *Sign-Based Construction Grammar*. The Oxford Handbook of Linguistic Analysis, Oxford University Press, 2015.

[29] J. Parsons, M. Schrider, O. Ogunlela, and S. Ghanavati, "Understanding Developers Privacy Concerns Through Reddit Thread Analysis," in *Joint Proceedings of REFSQ-2023 Workshops, Doctoral Symposium, Posters and Tools Track and Journal Early Feedback co-located with the 28th International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2023), Barcelona, Catalunya*, 2023.

[30] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), 2016, OJ L 119/1.

[31] E. Rosch, Principles of Categorization, E. Rosch and B. Lloyd (eds), Cognition and categorization, 27-48, Hillsdale, NJ: Lawrence Erlbaum, 1978.

[32] J. Saldana, The Coding Manual for Qualitative Researchers, SAGE Publications, 2015.

[33] S. Santos, T. Breaux, T. Norton, S. Haghighi and S. Ghanavati, Requirements Satisfiability with In-Context Learning, accepted to the 32nd IEEE Requirements Engineering Conference, Reykjavik, Iceland.

[34] J. R. Searle, Constitutive rules (Version 1). Università degli studi di Sassari, 2018.

[35] D. Schiffrin, Discourse Markers. Cambridge: Cambridge University Press, 1987.

[36] W.R. Shadish, T.D. Cook, and D.T. Campbell. *Experimental and Quasi-experimental Designs for Generalized Causal Inference.* Houghton-Mifflin Company, Boston, Massachusetts, 2002.

[37] K-J. Stol, P. Ralph, B. Fitzgerald. "Grounded theory in software engineering research: a critical review and guidelines." *8th International Conference on Software Engineering*, pp. 120-131, 2016.

[38] A. Strauss, J. Corbin, *Basics of Qualitative Research.* Thousand Oaks, CA, USA: Sage, 1990.

[39] M. Tahaei, K. Vaniea, and N. Saphra, "Understanding privacy-related questions on Stack Overflow", in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2020.

[40] S. Toulmin, *The Uses of Argument*, 2nd ed. Cambridge University Press, 2003.

[41] A. Tversky, Features of similarity, Psychological Review, 84(4), 327–352, 1977.

[42] R.K. Yin. *Case study research*, 6th ed., In Applied Social Research Methods Series, v.5. Sage Publications, 2018.

[43] D. Wurfel, R. Lutz, S. Diehl. "Grounded requirements engineering: An approach to use case driven requirements engineering," *Journal of Systems and Software*, 11: pp. 645-657, 2016.

[44] F. Zahid, A. Tanveer, M.M.Y. Kuo, et al. A systematic mapping of semi-formal and formal methods in requirements engineering of industrial Cyber-Physical systems. J Intell Manuf 33, 1603–1638 (2022). https://doi.org/10.1007/s10845-021-01753-8

[45] M. van Zee, F. Bex, S. Ghanavati. "RationalGRL: A framework for argumentation and goal modeling," *Argument and Computation*, 12(2): 191-245, 2021.

[46] L. Zhou, W. Wang, T. Liu, and J. Zhang, "Cross-Modal Fusion for Image and Text Retrieval using Transformer-based Models," *arXiv preprint arXiv:2210.07128*, 2022.