Justification Logic, Inference Tracking, and Data Privacy

Thomas Studer

Abstract

Internalization is a key property of justification logics. It states that justification logics internalize their own notion of proof which is essential for the proof of the realization theorem. The aim of this note is to show how to make use of internalization to track where an agent's knowledge comes from and how to apply this to the problem of data privacy.

Keywords: justification logic, internalization, knowledge tracking, data privacy

1 Introduction

Justification logics [3] are epistemic logics that include explicit justifications for an agent's knowledge and they allow to reason with and about these justifications. The first justification logic, the *logic of proofs*, has been developed by Artemov [1, 2] to provide S4 with a provability semantics. Since then justification logics have been applied to a variety of problems. For instance, these logics have been used to create a new approach to the logical omniscience problem [5], to study self-referential proofs [8], and to investigate the role of the announcement as a justification in public announcement logics [7].

Instead of statements A *is known*, denoted by $\Box A$, justification logics reason about justifications for knowledge by using constructs $t : A$ that stand for t *is a justification for* A. In those statements, the evidence term t can be viewed as an informal justification for A or a formal mathematical proof of A depending on the application.

The structure of terms in a given justification logic corresponds to the axiomatization of that theory so as to guarantee the property of internalization: for each derivation $\mathcal{D}$ of a theorem A of the logic in question, there is

a step-by-step construction that transforms $\mathcal{D}$ into a term $t_{\mathcal{D}}$ in such a way that $t_{\mathcal{D}} : A$ is also a theorem of the logic. Therefore, the term $t_{\mathcal{D}}$, describes the reasons, according to the logic, why A must hold. This suggests that we can think of a term t in a formula $t : A$ as an explicit reason that justifies the assertion A.

The aim of the present note is to show how to make use of internalization for inference tracking. Assume that a formula A is derivable from a theory Δ. Internalizing a derivation of A from Δ gives a term $t_{\mathcal{D}}$ which basically is a blueprint of that derivation. In particular, we can read off from the evidence term $t_{\mathcal{D}}$ which axioms of Δ have been used in the derivation of A. Artemov [4] considers an example of evidence tracking where the structure of evidence terms allows to discern factive and non-factive justifications.

We are going to use inference tracking to study certain data privacy issues. A user of an information system usually has only limited access to the data stored in the system. This is controlled by assigning to the user a view definition which is a restricted set of queries that the user is allowed to issue. The only way the user can get information about the data stored in the system is via the queries provided by the view definition. There are two problems that need to be addressed in this approach.

1. What can the user infer from the information he may gain by issuing the queries? That means in particular, is privacy preserved or is it possible to infer sensitive information from the answers to the queries?

2. If privacy is not preserved, that is if the view definition leaks sensitive information, how can the user's access rights be restricted in order to keep the secrets.

We will see in this note that internalization and inference tracking provide means to approach these two problems.

In the next section we introduce the justification logic J, which is the justified counterpart of the modal logic K, and we recall the internalization property for J. Section 3 presents our running example and illustrates how inference tracking works. Then, in Section 4, we give a formal definition of the problem of data privacy and study it from the point of view of justification logic. We conclude the paper in Section 5.

2 Justification Logic and Internalization

Definition 1 (Language)**.** We fix countable sets $\mathsf{Cons} = \{c_1, c_2, \ldots\}$ of *constants*, $\mathsf{Vars} = \{x_1, x_2, \ldots\}$ of *variables*, and Prop of *atomic propositions*. The

language of J consists of the *terms* $t \in \mathsf{Tm}$ and the *formulas* $A \in \mathsf{Fml}$ formed by the following grammar

$$\begin{array}{rcl} t & ::= & x \mid c \mid (t \cdot t) \mid !t \\ A & ::= & p \mid \neg A \mid (A \to A) \mid t : A \end{array}$$

where $x \in \mathsf{Vars}$, $c \in \mathsf{Cons}$, $p \in \mathsf{Prop}$. We define the connectives $\wedge$ and $\vee$ as usual. To say that a term $t \in \mathsf{Tm}$ is *ground* means that t does not contain variables

Often the language of justification logic also includes a binary term operator $+$. However, for the purpose of this paper we do not need this operator and, therefore, dispense with it.

Definition 2 (Deductive System)**.** The *axioms of* J consist of all Fml-instances of the following schemes.

1. All classical propositional tautologies
2. $t : (A \to B) \to (s : A \to t \cdot s : B)$ (application)

A *constant specification* $\mathcal{CS}$ is any subset

$$\mathcal{CS} \subseteq \bigcup \{c : A \mid c \in \mathsf{Cons} \text{ and } A \text{ is an axiom of } \mathsf{J}\}.$$

A constant specification $\mathcal{CS}$ is called *axiomatically appropriate* if for each axiom A of J there is a constant $c \in \mathsf{Cons}$ such that $c : A \in \mathcal{CS}$.

The *deductive system* $\mathsf{J}(\mathcal{CS})$ is the Hilbert system consisting of the above axioms of J and the following rules of *modus ponens* (MP) and *axiom necessitation* (AN):

$$\frac{A \quad A \to B}{B}\ , \qquad \frac{c : A \in \mathcal{CS}}{\underbrace{!!\ldots!}_{n} c : \ldots : !c : c : A}\ ,$$

where $n \geq 0$ is an integer.

For an arbitrary $\mathcal{CS}$ we write $\Delta \vdash_{\mathcal{CS}} A$ to state that A is derivable from Δ in $\mathsf{J}(\mathcal{CS})$.

Internalization is a crucial property of justification logics. It states that the logic internalizes its own notion of proof which is a key ingredient in the proof of the realization theorem [2].

Lemma 3 (Internalization)**.** *Let $\mathcal{CS}$ be an axiomatically appropriate constant specification. If*

$$B_1, \ldots, B_n \vdash_{\mathcal{CS}} A$$

then there is a term $t(x_1, \ldots, x_n) \in \mathsf{Tm}$ such that

$$x_1 : B_1, \ldots, x_n : B_n \vdash_{\mathcal{CS}} t(x_1, \ldots, x_n) : A.$$

Proof. The proof is by induction on the length of the derivation of A. We distinguish the following cases.

1. A is an axiom of J. Since $\mathcal{CS}$ is axiomatically appropriate, there exists a constant c such that $\vdash_{\mathcal{CS}} c : A$.

2. A is one of the B_is. We have $x_i : B_i \vdash_{\mathcal{CS}} x_i : B_i$.

3. A is the conclusion of B and $B \to A$ by modus ponens. By the induction hypothesis we find that there exist terms t_1 and t_2 such that

   $$x_1 : B_1, \ldots, x_n : B_n \vdash_{\mathcal{CS}} t_1(x_1, \ldots, x_n) : B$$

   and

   $$x_1 : B_1, \ldots, x_n : B_n \vdash_{\mathcal{CS}} t_2(x_1, \ldots, x_n) : B \to A.$$

   By the application axiom and modus ponens we find

   $$x_1 : B_1, \ldots, x_n : B_n \vdash_{\mathcal{CS}} t_2(x_1, \ldots, x_n) \cdot t_1(x_1, \ldots, x_n) : A.$$

4. A is the conclusion of axiom necessitation. Then there exists a ground term t such that $t : A$ also follows from axiom necessitation. □

Remark 4. It is easy to see that the term $t(x_1, \ldots, x_n)$ constructed in the proof of the internalization lemma directly corresponds to the original derivation of A from $B_1, \ldots, B_n$. In particular, we see that if a variable x_i does not occur in the constructed justification term for A, then the corresponding assumption B_i has not been used to derive A. That is we have

$$B_1, \ldots, B_{i-1}, B_{i+1}, \ldots, B_n \vdash_{\mathcal{CS}} A.$$

3 Inference Tracking

For the rest of this paper, we assume that we have a fixed axiomatically appropriate constant specification $\mathcal{CS}$ and we will write $\vdash$ for $\vdash_{\mathcal{CS}}$.

Let us now introduce our running example dealing with a set Δ of medical knowledge. Of course, this is only a toy example. For privacy issues concerning similar real world data we refer to [9]. The set Δ includes the following facts:

1. Patient 1's diagnosis is *broken leg* or *cancer*:

$$\mathsf{Patient1} \to \mathsf{brokenLeg} \vee \mathsf{cancer}. \tag{A}$$

2. Patient 1 lives in city A:

$$\mathsf{Patient1} \to \mathsf{cityA}. \tag{B}$$

3. Patient 1 receives a high cost treatment:

$$\mathsf{Patient1} \to \mathsf{highCosts}. \tag{C}$$

4. A cancer diagnosis entails a high cost treatment:

$$\mathsf{cancer} \to \mathsf{highCosts}. \tag{D}$$

5. A broken leg diagnosis entails a low cost treatment (i.e. not high cost):

$$\mathsf{brokenLeg} \to \neg\mathsf{highCosts}. \tag{E}$$

We easily find $A, B, C, D, E \vdash \mathsf{Patient1} \to \mathsf{cancer}$. Let us now look at an internalization of this fact. We first assume the following assignment of variables to facts: $\Gamma := x_1 : A, x_2 : B, x_3 : C, x_4 : D, x_5 : E$. Further we assume that our constant specification $\mathcal{CS}$ contains the following, where we let the constants justify axiom schemes.

$$\begin{aligned}
&c_1 : (A \to \neg B) \to (B \to \neg A)\\
&c_2 : (A \to B) \to ((B \to C) \to (A \to C))\\
&c_3 : (A \to (B \vee C)) \to ((A \to \neg B) \to (A \to C))
\end{aligned}$$

Thus we obtain

$$\begin{aligned}
&\Gamma \vdash c_1 \cdot x_5 : \mathsf{highCosts} \to \neg\mathsf{brokenLeg}\\
&\Gamma \vdash c_2 \cdot x_3 : (\mathsf{highCosts} \to \neg\mathsf{brokenLeg}) \to (\mathsf{Patient1} \to \neg\mathsf{brokenLeg})\\
&\Gamma \vdash (c_2 \cdot x_3) \cdot (c_1 \cdot x_5) : \mathsf{Patient1} \to \neg\mathsf{brokenLeg}\\
&\Gamma \vdash c_3 \cdot x_1 : (\mathsf{Patient1} \to \neg\mathsf{brokenLeg}) \to (\mathsf{Patient1} \to \mathsf{cancer})\\
&\Gamma \vdash (c_3 \cdot x_1) \cdot ((c_2 \cdot x_3) \cdot (c_1 \cdot x_5)) : \mathsf{Patient1} \to \mathsf{cancer}
\end{aligned}$$

We see that the last evidence term, which justifies $\mathsf{Patient1} \to \mathsf{cancer}$, does not contain the variables x_2 and x_4. That means the statements B and D have not been used in the derivation of $\mathsf{Patient1} \to \mathsf{cancer}$.

Moreover, if we assume that the constant specification is such that each constant justifies at most one axiom scheme, i.e. it is schematically injective, then we can read off from the term $(c_3 \cdot x_1) \cdot ((c_2 \cdot x_3) \cdot (c_1 \cdot x_5))$ the concrete reasoning process that led from the knowledge base to the conclusion.

4 Data Privacy

We start with defining the basic notions we need for a precise treatment of the privacy problem.

1. A knowledge base $\mathcal{KB}$ is a deductively closed set of formulas, that is

$$\mathcal{KB} \vdash A \quad \Longrightarrow \quad A \in \mathcal{KB} \quad \text{for all formulas } A.$$

2. A *query* Q is a formula of Fml.

3. An knowledge base $\mathcal{KB}$ answers yes to a query Q if and only if $Q \in \mathcal{KB}$. Otherwise it answers no.

4. A *view definition* V_D is a set of queries.

5. A *view* V of a knowledge base $\mathcal{KB}$ under a view definition V_D is a subset of V_D consisting of those queries for which $\mathcal{KB}$ answers yes. Formally we set $V := V_D \cap \mathcal{KB}$.

6. A *secret* is a formula of Fml.

In a knowledge base system, privacy is ensured by restricting the set of queries a user is allowed to issue. Usually he is granted only access to a given view definition V_D. That means he is only allowed to issue those queries that are elements of V_D. The system will then answer those queries which results in a view V. The *problem of data privacy* is to decide whether it might be possible for that user to infer from the knowledge of V_D and V whether a given secret S belongs to the underlying unknown knowledge base $\mathcal{KB}$.

Assume that $V_D = \{V_1, \ldots, V_n\}$ is a view definition and S is a secret. In the simple setting presented above, the more queries answer yes, the more a user can infer. Thus to solve the problem of data privacy, we assume that all queries of the view definition answer yes. We find that privacy is preserved if $V_D \nvdash S$ and that the secret is revealed if $V_D \vdash S$.

In the case of $V_D \vdash S$ we can apply internalization and obtain

$$x_1 : V_1, \ldots, x_n : V_n \vdash t : S$$

for some term t. Again, the variables occurring in t tell us which queries of V_D contributed to the derivation of S, i.e. are responsible for the privacy breach. This information can be used to find a more restrictive view definition, which is a subset of V_D, that preserves privacy. Of course, simply removing one of the queries that was involved in the derivation of S does not guarantee

privacy for there may be other derivations of S. Still, this approach provides valuable information for finding a privacy preserving view definition.

Instead of altering the view definition, another approach to privacy [6] suggest to alter the knowledge base (that is to make it lying) in order to preserve privacy. In that approach we could use the information provided by the justification terms to find a minimal change to the knowledge base that makes it privacy preserving.

As an example, consider the formulas A, B, C, D, E from the previous section. Assume that we are given an information system where a user is granted access to the view definition $V_D = \{A, B, C\}$ and assume that D, E are general background knowledge the user has without accessing the information system. The confidential information that we want to keep secret is

$$\mathsf{Patient1} \to \mathsf{brokenLeg} \text{ and } \mathsf{Patient1} \to \mathsf{cancer}.$$

That is want to hide the actual diagnosis of Patient 1. If either of the above statements were derivable, then we would know the diagnosis and privacy would be violated.

Since we have

$$\Gamma \vdash (c_3 \cdot x_1) \cdot ((c_2 \cdot x_3) \cdot (c_1 \cdot x_5)) : \mathsf{Patient1} \to \mathsf{cancer}, \tag{1}$$

we know that privacy does not hold. Moreover, the justification term in (1) tells us that only the queries A and C (but not B) have been used to derive the secret. Thus, to make the view definition privacy preserving, we have to remove either A or C from it (thereby restricting the user's access rights).

The definitions and techniques introduced before refer to so-called incomplete information system. Let us now turn to complete information systems. These systems work with a closed world assumption which in our setting means that we have

$$A \in \mathcal{KB} \text{ or } \neg A \in \mathcal{KB} \text{ for each formula } A.$$

Thus if the answer of $\mathcal{KB}$ to a query Q is no, then $\neg Q \in \mathcal{KB}$ holds.

Consider again the view definition $V_D = \{A, B, C\}$ and assume that the view V of $\mathcal{KB}$ under V_D consists only of A. That means in particular that the answer of $\mathcal{KB}$ to C is no. In the case where $\mathcal{KB}$ is incomplete, privacy is preserved since we cannot infer the actual diagnosis of Patient 1. However, if $\mathcal{KB}$ is complete, then we find that $\neg C \in \mathcal{KB}$. Hence we get

$$\mathcal{KB} \vdash \mathsf{Patient1} \to \mathsf{brokenLeg}.$$

Deciding whether privacy holds is more complex for complete knowledge bases than it is for incomplete ones. As an example, we assume again that

$V_D = \{V_1, \ldots, V_n\}$ is a view definition and S is a secret. As seen before, for incomplete knowledge bases we simply test $V_D \vdash S$ to decide privacy. For complete knowledge bases, however, we also have to take into account the possibility that a user may know $\neg V_i$ for a $V_i \in V_D$. Of course, we have

$$V_1, \neg V_1, \ldots, V_n, \neg V_n \vdash A \tag{2}$$

for any formula A. Thus simple logical consequence cannot be used as a test for privacy (unlike in the case of incomplete systems).

Let us now study the internalized version of (2) which is

$$x_1 : V_1, x_2 : \neg V_1, \ldots, x_{2n-1} : V_n, x_{2n} : \neg V_n \vdash t : A \tag{3}$$

for some term t. First we note that (3) does *not* hold for all terms t. Let t be such that (3) holds, then again t carries information on the derivation of A. We find that

1. if (3) holds for a term t that does not contain both x_{2i-1} and x_{2i} for all $1 \leq i \leq n$, then the view leaks the secret.

2. if (3) only holds for terms t that for some $1 \leq i \leq n$ contain both x_{2i-1} and x_{2i}, then privacy holds.

5 Conclusion

In this note we showed how to apply the internalization property of justification logics to problems of data privacy. The key property is that if a secret is derivable from a given view, then internalization allows us to reason about what part of the view is responsible for the privacy breach.

On a technical level, the reason for this is that justification logics explicitly include terms witnessing the reason why an agent knows something. In a pure modal logic approach, the formula $\Box A$ does not tell us why a secret A is known to the agent. Thus we have no information about how to restrict the agent's access rights such that privacy is preserved. In justification logic we have $t : A$ and the term t includes the information which queries of the view are responsible for leaking the secret.

For complete information systems we need justifications already to test whether privacy holds or not. In the modal logic K

$$\Box V_1, \Box\neg V_1, \ldots, \Box V_n, \Box\neg V_n \vdash \Box A$$

holds for any formula A. Thus we cannot use modal logic to test whether a secret A is revealed or not. In the justified version (3) of the above expression,

the conclusion contains more information. There it reads $t : A$, which is not derivable for all terms t, and we can use the term t in (3) to check whether the view reveals the secret.

Finally, we would like to mention that the approach presented in this paper does not only work for propositional knowledge bases. It can be extended, for example, to deal also with the privacy problem for description logic knowledge bases [10]. For this we need a justification logic that is defined over a description logic, which recently has been developed in [11].

References

[1] Sergei N. Artemov. Operational modal logic. Technical Report MSI 95–29, Cornell University, December 1995.

[2] Sergei N. Artemov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, 7(1):1–36, March 2001.

[3] Sergei [N.] Artemov. The logic of justification. *The Review of Symbolic Logic*, 1(4):477–513, December 2008.

[4] Sergei [N.] Artemov. Tracking evidence. In Andreas Blass, Nachum Dershowitz, and Wolfgang Reisig, editors, *Fields of Logic and Computation, Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*, volume 6300 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 2010.

[5] Sergei [N.] Artemov and Roman Kuznets. Logical omniscience as a computational complexity problem. In Aviad Heifetz, editor, *Theoretical Aspects of Rationality and Knowledge, Proceedings of the Twelfth Conference (TARK 2009)*, pages 14–23, Stanford University, California, July 6–8, 2009. ACM.

[6] Joachim Biskup and Lena Wiese. Preprocessing for controlled query evaluation with availability policy. *Journal of Computer Security*, 16(4):477–494, 2008.

[7] Samuel Bucheli, Roman Kuznets, Bryan Renne, Joshua Sacks, and Thomas Studer. Justified belief change. In *Proc. of LogKCA-10*, 2010.

[8] Roman Kuznets. Self-referential justifications in epistemic logic. *Theory of Computing Systems*, 46(4):636–661, May 2010.

[9] Kilian Stoffel and Thomas Studer. Provable data privacy. In K. Viborg, J. Debenham, and R. Wagner, editors, *DEXA 2005*, volume 3588 of *LNCS*, pages 324–332. Springer, 2005.

[10] Phiniki Stouppa and Thomas Studer. Data privacy for $\mathcal{ALC}$ knowledge bases. In S. Artemov and A. Nerode, editors, *LFCS 2009*, volume 5407 of *LNCS*, pages 409–421. Springer, 2009.

[11] Thomas Studer. Justified terminological reasoning. In E. Clarke, I. Virbitskaite, and A. Voronkov, editors, *PSI 11. Proceedings of the 8th Andrei Ershov Informatics Conference*, LNCS. Springer, to appear.

Address
Thomas Studer
Institut für Informatik und angewandte Mathematik, Universität Bern
Neubrückstrasse 10, CH-3012 Bern, Switzerland
`tstuder@iam.unibe.ch`