Datalog for Inconsistency-tolerant Knowledge Engineering

Hendrik Decker
Instituto Tecnológico de Informática, Universidad Politécnica de Valencia, Valencia, Spain

Keywords: Inconsistency Tolerance, Datalog, Integrity.

Abstract: Inconsistency tolerance is widely discussed and accepted in the scientific community of knowledge engineering. From a principled, theoretical point of view, however, the fundamental conflict of sound reasoning with unsound data has remained largely unresolved. The vast majority of applications that need inconsistency tolerance either does not care about a firm theoretical underpinning, or recurs on non-standard logics, or superficially refers to well-established classical foundations. We argue that hardly any of these paradigms will survive in the long run. We defend the position that datalog (Abiteboul et al., 1995), including integrity constraints, is a viable candidate for a sound and robust foundation of inconsistency-tolerant knowledge engineering. We line our argument by a propaedeutic glance at the history of issues related to inconsistency.

1 INTRODUCTION

In computing, the term "inconsistency tolerance" denominates the capacity of software systems to process data correctly even if the data are inconsistent. In knowledge engineering (abbr. *KE*), which comprises various kinds of automated reasoning with data, inconsistency tolerance also means to produce valid conclusions from inconsistent data, and the capability to make guarantees about the correctness of results in the presence of inconsistency.

KEsubsumes subfields of database management such as query answering, updating, integrity management, the evolution and integration of schemas and ontologies, etc. These are the fields of interest in this paper. Due to limitations of space and time, we do not deal here with KE subfields such as requirements engineering, knowledge acquisition and conceptual modeling.

In the KE literature, inconsistency tolerance is widely discussed and accepted as a desirable feature (Chopra and Parikh, 1999; Nuseibeh et al., 2000; Koogan Breitman et al., 2003; Bertossi et al., 2005; Calvanese et al., 2008; Imam and MacCaull, 2009; Qi et al., 2009; Hinrichs et al., 2009; Dunnei et al., 2009; Calvanese et al., 2012).

Unfortunately, however, solid formal foundations are largely missing. That is deplorable, since a lack of firm foundations of any technical approach always tends to abet doubts in its validity and universality.

Occasionally, such foundations are considered an unnecessary luxury or a practically irrelevant playground for egghead theoreticians. Such an attitude is frequently encountered with "application-oriented" people (who might very well have an admirable talent of producing amazing special-purpose solutions for intricate problems). However, solutions that are not grounded on theoretical foundations that have withstood the test of time tend to suffer the fate of most ad-hoc solutions: they lack generalizability, are hard to maintain, difficult to evolve and become old-fashioned soon after their novelty appeal has worn off.

Less lamentable, perhaps, are those proposals that favour some non-standard logic for capturing the principles that underly automated reasoning in the presence of inconsistency. Calculi that are paraconsistent, multi-valued, annotated, probabilistic or possibilistic are among the most frequently used technical means to provide a formal framework for consistent reasoning with inconsistency. Yet, those logics are largely divergent, and none of them has ever attained a status of acceptance that could be called a standard.

Several other approaches simply are content with relying more or less explicitly on conventional first-order predicate logic as a theoretical underpinning for their ways to cope in a reasonable way with inconsistency. However, they usually ignore or pass by the devastating effects that a deployment of full-fledged classical logic can have, due to the principle known as *ex contradictione sequitur quodlibet* (*ECQ*), i.e., that everything, and thus nothing reasonable at all, can be inferred from inconsistency.

In fact, the problem is not just to find workarounds for avoiding the explosive effects of inconsistency. The deep problem is that a foundation of inconsistency tolerance must provide a meta-level for rea-

Decker H..
Datalog for Inconsistency-tolerant Knowledge Engineering.
DOI: 10.5220/0004172202960301
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2012), pages 296-301
ISBN: 978-989-8565-30-3

soning about reasoning in the presence of inconsistency on the object-level. Such meta-level reasoning is bound to recur on irrefutable principles of rational reflection and argumentation that do not tolerate inconsistency, as long as it argues in its own defense. In particular, meta-level reasoning may insist on fundamental principles of logic such as the law of non-contradiction (*LNC*) (i.e., no statement can be both true and false), and *reductio ad absurdum* (*RaA*) (i.e., to infer the negation of a hypothesis that would lead to a contradiction), which, at the same time, are abdicated on the object-level. In other words, reasoning consistently with inconsistent data risks to be self-contradictory.

In Section 2, we take a propaedeutic look on the historical roots of LNC, RaA and ECQ, and their relationship to inconsistency. In Section 3, we argue why datalog, augmented with integrity constraints, is not just a lucky compromise or a pragmatically convenient tool, but an appropriate candidate for a sound and robust foundation of inconsistency-tolerant KE. It respects and deploys LNC and RaA while tolerating inconsistency and avoiding the application of ECQ. In Section 4, we conclude.

2 LNC, RaA, ECQ, LEM AND INCONSISTENCY

In logic, consistency is understood as the absence of contradiction, and inconsistency as the presence of contradiction. LNC denunciates inconsistency as illogical. Consistency and LNC have played constitutional roles in western philosophy, logic and computation. Aristotle and Kant took LNC to be a *conditio sine qua non* of all reasoning. In mathematical logic, LNC has had a solid standing since Leibniz formalized it as the fundamental principle of human comprehension. Frege attempted to base mathematics on LNC, defining the elementary number 0 as the cardinality of the set of true contradictions. Hilbert and Gödel required proofs of consistency as the most desirable property of any mathematical theory. The semantic consistency of stored data, a.k.a. integrity, is a key requirement in most database systems. The vast majority of established proof procedures, including abductive ones, implicitly or explicitly use LNC for making valid inferences.

Also inconsistency is foundational in many philosophical, logical and computational systems. Heraklit, Zenon, Plato, Hegel, C. G. Jung and others have taken inconsistency (as embodied by contradictions, paradoxes, dialectic aporias, thesis/antithesis or opposed polarities) as a constitutive element of human cognition. Popper recalled the ancient wisdom that universal sentences can never be proved by experience, but only be falsified by contradiction. Moreover, as already indicated, inconsistency is constitutional in the principle of RaA, which has been a venerable inference rule since the ancient Greeks. Many automated theorem provers are based on RaA, i.e., proving a sentence by showing its negation to be inconsistent. Nowadays, data mining may infer useful information from detected contradictions, e.g., for information integration (Müller et al., 200), decision support (Padmanabhan and Tuzhilin, 2002), or identifying fraudulent tax declarations (Bonchi et al., 1999; Yu et al., 2003).

On the other hand, LNC has not always enjoyed unanimous approval. Aristotle discussed his own doubts about LNC. Medieval and modern-day philosophers became aware of the potentially devastating logical effects of any violation of LNC by the ECQ principle, by which anything (thus, nothing useful at all) can be derived from contradiction. Similar to abandoning the axiom of parallels in non-Euklidian geometry, Peirce, Lukasiewicz and Post contemplated to abandon LNC, and Vasiliev effectively did so, in systems of non-Aristotelian logic.

Russell shattered Frege's LNC-based attempt by showing naïve set theory to be inconsistent. That, and the inflationary effects of ECQ ignited Orlov and Lewis to tame ECQ by introducing less powerful forms of implication. The more radical approaches proposed by Jakowski, da Costa and others rejected the universality of LNC or, in some cases, the law of excluded middle (*LEM*), a.k.a. *tertium non datur*, in order to avoid ECQ.

According to (Diamond, 1976), Wittgenstein denied the malignance of LNC and ECQ, by (1) qualifying known and unknown inconsistencies as innocuous ("When an inconsistency comes out into the open it can do no harm" and "As long as its hidden an inconsistency is as good as gold"), (2) suggesting a kind of exception handling for known inconsistency ("If an inconsistency were to arise (...), all we have to do is to make a new stipulation to cover the case where the rules conflict and the matters resolved""), and (3) proposing to confine inconsistency by not drawing any conclusions from it ("You might get p.$\sim$p by means of Freges system. If you can draw any conclusion you like from it, then (...) I would say, 'Well, then, just dont draw any conclusions from a contradiction' ").

By common standards, Wittgenstein's attitude toward inconsistency seems frivolous. In fact, (1) does not consider that inconsistent data which are not known to be inconsistent may be used bona fide to

derive possibly fatal consequences; (2) does not take into account that exception handling easily gets out of hand as the number of exceptions grows; finally, (3) can only be qualified as a much too careless statement, since wrong conclusions can be drawn from data involved in contradiction "without actually going through the contradiction", as remarked by Turing (Diamond, 1976).

Nevertheless, a philosophical debate has been going on about the untenability or, resp., the justifiability of Wittgenstein's enunciations on contradictions and inconsistency. For instance, see (Chihara, 1977; Wright, 1980; Wrigley, 1980; Caruana, 2004; Bagni, 2008; Decker, 2010). Basically, the positions are that, either Wittgenstein's arguments are not cogent, unconvincing, vague, besides the point, if not plainly wrong, or that his utterances should be appreciated in a pragmatic sense, or in the context of his own mindset and certain tendencies of his times. Anyway, Wittgenstein's suggestion to refrain from inferring arbitrary conclusions from contradictions remain incomplete, since he did not hint at any systematic approach of how to achieve that.

3 DATALOG

In this section, we are going to see that Wittgenstein's recommendation of how to confine inconsistency is complied with easily, in the framework of datalog.

In datalog, each theory T is usually represented by a set of Horn clauses of the form $H \leftarrow B$, where H either is a positive literal called *head* or is empty, and B is a possibly empty conjunction of positive literals called body. H can be inferred in T if B can be inferred in T. Atoms that do not match any literal in the head of any clause in T cannot be inferred in T. Thus, as opposed to ECQ, there is no way to derive any arbitrary conclusion from T.

Each such T in datalog is partitioned into a database D and an integrity theory IC. For each clause in D, its head is not empty. Each clause in IC, called *integrity constraint* (or, in short, *constraint*), is represented as a *denial*, i.e., a clause with empty head, from which nothing can be inferred. The body of each constraint expresses a condition that should not hold. If it does, then the database is inconsistent. Thus, inconsistency is syntactically hedged in datalog: inconsistency of $T = D \cup IC$ means that some I in IC is violated, i.e., the body of I can be inferred in D. And nothing more can be derived from that.

This way of representing inconsistency in datalog is sometimes emphasized by rewriting each denial $\leftarrow B$ in IC as $violated \leftarrow B$, where *violated* is a distinguished 0-ary predicate that does not occur in the body of any clause in IC nor in any clause of D. Thus, in each datalog theory T, inferences are immune against any inconvenience associated with inconsistency, since nothing (or, at most, *violated*) can be derived from a contradiction such as $A \in D$ and $\leftarrow A \in IC$.

Early on, Kowalski has pointed out that logic programming (*LP*), and hence datalog (which is a somewhat restricted form of LP), has the potential of paraconsistency (Kowalski, 1979). In LP, no use is ever made of the law of disjunctive weakening (*LDW*) by which conclusions $p \vee q$ can be inferred from any premise p for arbitrary q, so that ECQ cannot become effective. That is, q cannot be inferred from contradictory premises p, $\sim p$ by inferring $p \vee q$ and resolving that with $\sim q$.

Datalog has been characterized as a form of resource-constrained first-order predicate logic. In practice, the available computer memory and time may always constrain the power of computation. Apart from that, however, the resource-constrained approach of inferencing in datalog is indeed beneficial in terms of inconsistency tolerance, as observed in (Kowalski, 1988), and also in terms of the oxymoron of reasoning consistently with inconsistent knowledge, as exposed in the introduction.

More precisely, datalog renounces on several resources of inference mechanisms that are available in classical logic, without sacrificing the computational power and deductive capacities that are needed for knowledge engineering. While Datalog involves RaA as an essential inference principle, it never applies LDW, and thereby avoids the effects of ECQ.

Moreover, the goal-orientedness of datalog can be interpreted as another form of constraining resources, since inference steps that evidently are not conductive to reach the goal (which either is to deduce an answer to a query or to test if a constraint is violated) simply are not taken.

Yet, apart from the possibly explosive effect of LDW, there is another possible cause of ECQ becoming effective, as identified in (Hewitt, 2012). That possible cause uses RaA, which, similar to LDW, is an inference rule, i.e., a principle on the meta-level of reasoning, and goes as follows. Let $T = \{p, \sim p\}$ and $\sim q$ a hypothesis, where q is an arbitrary sentence. Then, $(T \cup \{\sim q\}) \vdash p \wedge \sim p$ holds trivially. From that, RaA infers q.

However, the only time RaA is applied in datalog is the moment in which a *goal clause* (i.e., a query in denial form or a constraint) is refuted by input clauses from the database. The refutation of a goal of the form $\leftarrow B$ in a database D means that $D \cup \{\leftarrow B\}$ is incon-

sistent, and hence the existential closure $\exists B$ is inferred from D. (Moreover, the refutation of $\leftarrow B$ in D also computes a set of *answer substitutions* $\{\theta_1,\ldots,\theta_n\}$ $(n>0)$ of the variables that are free in B such that $D\vdash\forall B\theta_i$ $(1\leq i\leq n)$, but that is not of importance in this context).

Now, if it happens that, e.g., A is a unit clause in a database D and $\leftarrow A$ is a constraint in an integrity theory IC, then the refutation of the goal $\leftarrow A$ in D correctly states that A is *true* in D (or, proof-theoretically speaking, A can be inferred from D), and that $\leftarrow A$ is violated in D. But no other consequence is inferred from that inconsistency, and in particular not in the way arbitrary sentences can be inferred from RaA in general, as indicated above.

Now, let us wrap up what we have seen up to this point. By amalgamating object- and meta-level reasoning (Bowen and Kowalski, 1982), datalog is a self-consistent problem solving paradigm that may consistently reason about its own reasoning, even if the latter is done with inconsistent knowledge. Thus, datalog appears to be an ideal candidate for inconsistency-tolerant knowledge engineering.

In particular, datalog can be seen as a realization of Wittgenstein's advice to simply not draw any conclusion from inconsistent sentences. Similarly, each paraconsistent logic can be seen that way. Yet, the essential difference is that datalog is much closer to classical logic than any other paraconsistent form of reasoning.

However, as soon as other reasoning principles are used in datalog applications, i.e., on top of datalog, more caution has to be taken. For example, for checking if updates would violate integrity, most integrity checking methods assume the *total integrity premise*, i.e., that the theory before the update is consistent.

For instance, let D be a database containing $r(a,a)$ and IC an integrity theory containing $\leftarrow r(x,x)$ and $\leftarrow r(a,y)\wedge s(y)$, where r, s are predicates, x, y are variables and a is a constant. Clearly, $D\cup IC$ is inconsistent. For checking if the update $U = \mathit{insert}\, s(a)$ in D violates integrity, the instance $\leftarrow r(a,a)\wedge s(a)$ of the constraint $\leftarrow r(a,y)\wedge s(y)$ is considered relevant by most methods, since U may violate it, while $\leftarrow r(x,x)$ is considered irrelevant, since it cannot be violated by U. By the total integrity premise, $\leftarrow r(x,x)$ is not violated before the update. Hence, some methods (e.g., the one in (Gupta et al., 1994)) wrongly infer that $\leftarrow r(x,x)\wedge s(a)$ also cannot be violated by U, since $\leftarrow r(a,a)\wedge s(a)$ is subsumed by $\leftarrow r(x,x)$. Thus, such methods do not confine inconsistency, since they risk to miss an increase of inconsistency across updates.

As opposed to that, it is shown in (Decker and Martinenghi, 2011) how to confine inconsistency in databases across updates, by methods for inconsistency-tolerant integrity checking (*ITIC*) that do not recur on the total integrity premise. A more general approach to ITIC in database theories of the form (D,IC) is discussed in (Decker, 2012). It is based on a definition of *inconsistency measures*, which size the amount of inconsistency in a database, and allow to determine if an update increases or decreases the current amount of inconsistency. Hence, an integrity checking method is re-defined to be inconsistency-tolerant if it only accepts updates that do not increase the amount of integrity violation in the updated database.

Inconsistency measures also may serve for computing consistency-preserving updates and partial repairs of inconsistent databases that decrease the amount of constraint violations, as shown in (Decker, 2012). Another application of inconsistency measures is an inconsistency-tolerant approach to the evolution of database schemas, as described for some specific measures in (Decker, 2011b). Also, the consistency preservation of concurrent transactions can be controlled in an inconsistency-tolerant manner by inconsistency measures, as shown in (Decker and Muñoz-Escoí, 2010) and (Cuzzocrea et al., 2012), where inconsistency measures are also used for uncertainty management. Moreover, specific inconsistency measures allow to determine if an answer to a given query "has integrity" or not, by checking if the data involved in computing answer substitutions are disjoint or not from the data involved in any constraint violation, as shown in (Decker, 2011a).

Even though a lot of datalog-based KE methods had not been conceived for working in the presence of inconsistency, many of them have turned out to be inconsistency-tolerant, in the sense of confining extant integrity violations. Thus, the capacity of being inconsistency-tolerant comes for free in most conventional methods. This observation confirms the main point of this paper, which is that the inconsistency tolerance of datalog, as well as of many important datalog-based KE applications, provides a reliable reasoning that guarantees consistency in the presence of inconsistency without further ado.

4 CONCLUSIONS

In this paper, we have argued that datalog is a viable solution to the problem of pragmatic but theoretically well-founded reasoning with data that are possibly inconsistent.

To facilitate our arguments, we have confined attention to the *definite* case of datalog, i.e., we have

not considered its extension by non-monotonic negation, as obtained by an abductive or argumentation-theoretic interpretation of negative literals in bodies of clauses. Also, we have not considered extension to a more general syntax and semantics of integrity constraints that allow disjunctions of atoms in the head of clauses, as proposed, e.g., in a variety of papers by Robert Kowalski his co-authors.

Future work of ours is concerned with defending the claim that the so-extended datalog continues to go out of the way of any inadvertent application of ECQ, and thus is an even more powerful paradigm for inconsistency-tolerant KE. Here, we already remark that the abductive interpretation of negation involves an active use of LNC. As opposed to that, abductive datalog is careful with applying LEM, an unbridled use of which may lead to inconsistent conclusions, as shown in (Dung, 1995).

A more radical approach to embrace inconsistency as an ubiquitous feature in computing and KE on a foundational level has been proposed by (Hewitt, 2012). As opposed to datalog, which, by its avoidance of LDW and its controlled, goal-oriented use of RaA, is consistent on the meta-level, Hewitt's Direct Logic (which does not support RaA) is inherently inconsistent, on purpose, and arguably is even more in line with Wittgenstein's thoughts on inconsistency. Perhaps, time will tell if the conservative stance of datalog (by which inconsistency on the object-level can be kept at bay by a consistent, resource-constrained way of reasoning on the meta-level) could prevail over an approach that fully embraces inconsistency.

ACKNOWLEDGEMENTS

The work of the author for this publication has been partially supported by FEDER (European Fund for Regional Development) and the grants TIN2009-14460-C03 and TIN2010-17139 from the Spanish Ministry of Economy and Competitiveness.

REFERENCES

Abiteboul, S., Hull, R., and Vianu, V. (1995). *Foundations of Databases*. Addison-Wesley.

Bagni, G. (2008). Obeying a rule: Ludwig Wittgenstein and the foundations of set theory. *The Montana Mathematics Enthusiast*, 5(2,3):215–222.

Bertossi, L., Hunter, A., and Schaub, T. (2005). *Inconsistency Tolerance*, volume 3300 of *LNCS*. Springer.

Bonchi, F., Giannotti, F., Mainetto, G., and Pedreschi, D. (1999). Using data mining techniques in fiscal fraud detection. In *Proc. 1st DaWaK*, volume 1676 of *LNCS*, pages 369–376. Springer.

Bowen, K. and Kowalski, R. A. (1982). Amalgamating language and metalanguage. In Clark, K. and Tärnlund, S.-A., editors, *Logic Programming*, pages 153–172. Academic Press.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R. (2008). Inconsistency tolerance in P2P data integration: an epistemic logic approach. *Information Systems*, 33(4-5):360–384.

Calvanese, D., Kharlamov, E., Montali, M., and Zheleznyakov, D. (2012). Inconsistency tolerance in OWL 2 QL knowledge and action bases. In Klinov, P. and Horridge, M., editors, *Proc. OWL Experiences and Directions Workshop*, volume 849. CEUR Electronic Workshop Proceedings. Available at "http://ceur-ws.org/Vol-849/.

Caruana, L. (2004). Wittgenstein and the status of contradictions. In A. Coliva, E. P., editor, *Wittgenstein Today*, pages 223–232. Il Poligrafo, Padova.

Chihara, C. (1977). Wittgenstein's analysis of the paradoxes in his lectures on the foundations of mathematics. *Philosophical Review*, 86(3):365–381.

Chopra, S. and Parikh, R. (1999). An inconsistency tolerant model for belief representation and belief revision. In *Proc. IJCAI*, pages 192–197. Morgan Kaufmann.

Cuzzocrea, A., de Juan-Marín, R., Decker, H., and Muñoz-Escoí, F. D. (2012). Managing uncertainty in databases and scaling it up to concurrent transactions. To appear in Springer LNCS.

Decker, H. (2010). How to contain inconsistency or, why Wittgenstein only scratched the surface. In *Proc. 7th European Conf. on Computing and Philosophy*, pages 70–75. Dr. Hut.

Decker, H. (2011a). Answers that have integrity. In Schewe, K.-D. and Thalheim, B., editors, *Semantics in Data and Knowledge Bases - 4th International Workshop SDKB*, volume 6834 of *LNCS*, pages 54–72. Springer.

Decker, H. (2011b). Causes for inconsistency-tolerant schema update management. In *Proc. 27th IDCE Workshops*, pages 157–161. IEEE CSP.

Decker, H. (2012). Measure-based inconsistency-tolerant maintenance of database integrity. To appear in Springer LNCS.

Decker, H. and Martinenghi, D. (2011). Inconsistency-tolerant integrity checking. *Transactions on Knowledge and Data Engineering*, 23(2):218–234.

Decker, H. and Muñoz-Escoí, F. D. (2010). Revisiting and improving a result on integrity preservation by concurrent transactions. In *Proc. OTM Workshops*, volume 6428 of *LNCS*, pages 297–306. Springer.

Diamond, C. (1976). *Wittgenstein's Lectures on the Foundations of Mathematics, Cambridge, 1939*. Harvester, Hassocks.

Dung, P. M. (1995). An argumentation-theoretic foundations for logic programming. *J. Logic Programming*, 22(2):151–171.

Dunnei, P., Hunter, A., McBurney, P., Parsons, S., and Wooldridge, M. (2009). Inconsistency tolerance in weighted argument systems. In *Proc. 8th Int. Conf.*

on Autonomous Agents and Multiagent Systems (AAMAS 2009), volume 2, pages 851–858. International Foundation for Autonomous Agents and Multiagent Systems.

Gupta, A., Sagiv, Y., Ullman, J. D., and Widom, J. (1994). Constraint checking with partial information. In *Proceedings of PODS 1994*, pages 45–55. ACM Press.

Hewitt, C. (2012). Formalizing common sense for scalable inconsistency-robust information integration using direct logic reasoning and the actor model. http://arxiv.org/pdf/0812.4852v85.

Hinrichs, T., Kao, J., and Genesereth, M. (2009). Inconsistency-tolerant reasoning with classical logic and large databases. In *Proc. 8th SARA*, pages 105–112. AAAI Press.

Imam, F. and MacCaull, W. (2009). Integrating healthcare ontologies: Inconsistency tolerance and case study. In D. Ardagna, M. M. and Yang, J., editors, *Business Process Management Workshops (BPM 2008)*, volume 17 of *Lecture Notes in Business Information Processing*, pages 373–384. Springer.

Koogan Breitman, K., Felicssimo, C. H., and Cysneiros, L. M. (2003). Semantic interoperability by aligning ontologies. In Galvo Martins, L. E. and Franch, X., editors, *Workshop em Engenharia de Requisitos (WER03)*, pages 213–222.

Kowalski, R. (1988). Logic-based open systems. In Hoepelman, J., editor, *Representation and Reasoning*, pages 125–134, Tübingen. Max Niemeyer Verlag.

Kowalski, R. A. (1979). *Logic for Problem Solving*. Elsevier.

Müller, H., Leser, U., and Freytag, J.-C. (200). Mining for patterns in contradictory data. In *Proc. 1st IQIS*, pages 51–58. ACM SIGMOD.

Nuseibeh, B., Easterbrook, S., and Russo, A. (2000). Leverage inconsistency in software development. *Computer*, 33(4):24–29.

Padmanabhan, B. and Tuzhilin, A. (2002). Knowledge refinement based on the discovery of unexpected patterns in data mining. *Decision Support Systems*, 33(3):309–321.

Qi, G., Haase, P., Schenk, S., Stadtmüller, S., and Hitzler, P. (2009). Inconsistency-tolerant reasoning with networked ontologies. Technical report, NeOn Deliverable D1.2.4.

Wright, C. (1980). *Wittgenstein on the Foundations of Mathematics*. Duckworth, London.

Wrigley, M. (1980). Wittgenstein on inconsistency. *Philosophy*, 55(214):471–484.

Yu, F., Qin, Z., and Jia, X.-L. (2003). Data mining application issues in fraudulent tax declaration detection. In *Proc. 2nd Conf. Machine Learning and Cybernetics*, pages 2202–2206. IEEE.