Can Owls Play Poker? (An OWL 2 DL Benchmark)

Guillermo Garda[1], Aidan Hogan[1,2]

[1]DCC, Universidad de Chile
[2]Instituto Milenio Fundamentos de los Datos

Abstract

We present ongoing work towards a challenging, parameterised and comprehensible OWL 2 DL benchmark based on poker, which we call the *OWL 2 Poker Benchmark* (*OWL2PB*). Current OWL benchmarks either involve a static and lightweight T-Box combined with parameterisable A-Box instances, or very large T-Boxes that are difficult to understand. We argue that poker offers the possibility of a challenging classification benchmark that involves a diversity of features of OWL 2 DL, allows for parameterising the complexity of the T-Box, and is understandable. Specifically, we propose a benchmark to classify the hands of poker with a classic 52-card deck (13 ranks, 4 suits), presenting the axioms to define them. This benchmark permits varying the T-Box complexity by considering the number of ranks and suits as parameters. We discuss performance results for three OWL 2 DL reasoners: HermiT, JFact and Openllet. The results demonstrate that OWL2PB offers a challenging stress-test of such reasoners.

1. Introduction

Description Logics (DLs) are decidable fragments of first-order logic. One of the most expressive DLs is $\mathcal{SROIQ}$ [1], built on top of the core DL ($\mathcal{ALC}$) features of concepts (or classes) like Human, and definitions of concepts based on their complement (not Human, denoted $\neg$Human), conjunction (Human and Alive, denoted Human $\sqcap$ Alive), disjunction (Human or Neanderthal, denoted Human $\sqcup$ Neanderthal), universal restriction (those whose friends are all Human, denoted $\forall$friend.Human), existential restriction (those with a friend who is Human, denoted $\exists$friend.Human), and the closure of these features (those with a friend who is Alive and Human or a Neanderthal whose friends are all not Alive, denoted $\exists\mathsf{friend}.((\mathsf{Alive}\sqcap\mathsf{Human})\sqcup(\mathsf{Neanderthal}\sqcap\forall\mathsf{friends}.\neg\mathsf{Alive}))$). In this core of DLs, we can also state axioms in the form of concept inclusions, such as to say that any instance of Neanderthal is also an instance of not Alive (Neanderthal $\sqsubseteq$ $\neg$Alive). We can also use the top concept that includes everything, and the bottom concept with no instance, denoted $\top$ and $\bot$ respectively.

$\mathcal{SROIQ}$ further supports: $\mathcal{S}$: *transitive relations*, like part of, denoted Trans(partOf); $\mathcal{R}$: *relation hierarchies*, like how mother is a sub-relation of parent, denoted mother $\sqsubseteq$ parent and *relation chains*, like how uncle is the composition / chain of parent and brother, denoted parent $\circ$ brother $\sqsubseteq$ uncle; $\mathcal{O}$: *nominals*, or classes comprising a fixed set of instances, like how the Benelux class of countries is equivalent ($\equiv$) to $\{\mathsf{Belgium}, \mathsf{Netherlands}, \mathsf{Luxembourg}\}$; $\mathcal{I}$: *inverse roles*, like how parent is the inverse of child, denoted parent $\equiv$ child$^-$; $\mathcal{Q}$: *qualified*

AMW 2026: 17th Alberto Mendelzon International Workshop on Foundations of Data Management, November 9th–November 13th, 2026, Arequipa, Peru
guillermo.garda@ug.uchile.cl (G. Garda); ahogan@dcc.uchile.cl (A. Hogan)
0009-0001-6840-1965 (G. Garda); 0000-0001-9482-1982 (A. Hogan)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

cardinality restrictions, like those who have at least two siblings who are Triplets, denoted ≥ 2 sibling.Triplet. $\mathcal{SROIQ}$ is then the DL underlying OWL 2 DL.

To see an example of $\mathcal{SROIQ}$ in action, consider that a Zebroid is a hybrid of a Zebra parent and a non-Zebra Equine parent. We could define this as the following *T-Box* (that is, the *terminological box*, containing class axioms and definitions):

$$\mathsf{Zebroid} \equiv \exists \mathsf{parent}.\mathsf{Zebra} \sqcap \exists \mathsf{parent}.(\neg \mathsf{Zebra} \sqcap \mathsf{Equine})$$

Now assume that we have the following unary and binary relations in our *A-Box* (that is, the *assertion box*, containing unary and binary relations over individuals), where dam is an equine mother, and sire an equine father:

$$\mathsf{dam}(\mathsf{Zia}, \mathsf{Lia}), \mathsf{sire}(\mathsf{Zia}, \mathsf{Frank}), \mathsf{dam}(\mathsf{Zach}, \mathsf{Lia}), \mathsf{sire}(\mathsf{Zach}, \mathsf{Frank}), \mathsf{Zebroid}(\mathsf{Zach})$$

We might guess that herein, Zebroid(Zia) holds, and in the Web of Data course at the University of Chile, students are asked to complete the axioms required to derive this conclusion, and run them in a shared OWL 2 DL reasoner. If their answer coincides closely with the intended solution, and is correct, then the reasoner will comply with the expected inference. However, other approaches that are potentially correct often cause the reasoner to not terminate.

The complexity of most reasoning tasks on $\mathcal{SROIQ}$ or OWL 2 DL ontologies is 2NExpTime-complete (for query answering, it's unknown if the problem is decidable or not). Thus it is not surprising that some ontologies might kill a reasoner. However, it is difficult to predict what features of an ontology are particularly costly for reasoning.

2. Related Work

In order to better understand the performance of OWL 2 DL systems, benchmarks are a potentially very useful tool. However, current benchmarks provide limited insights. Benchmarks such as the Lehigh University Benchmark (LUBM) [2] have long been used to understand the performance of OWL reasoners, but predate OWL 2 by several years, and feature a fixed, simplistic T-Box. An updated version – the University Ontology Benchmark (UOBM) [3] – later modified and extended LUBM to include more expressive OWL (1) ontologies for OWL Lite and OWL DL. OntoBench [4] enables generation of OWL 2 T-Boxes, but focuses on compliance testing rather than evaluating scalability or performance. OWL2Bench [5] uses universities as its core domain, provides a fixed T-Box in different OWL 2 profiles, and allows for generating A-Boxes of parameterised size, but has limited complexity. Another approach is to benchmark over real-world ontologies, such as from BioPortal [6], but such examples are not parameterisable, and are difficult to conceptualise without background knowledge.

3. Can Owls Play Poker?

We propose that poker can offer a stress-test for OWL 2 DL reasoners, and describe our ongoing work on the OWL 2 Poker Benchmark (OWL2PB), available at https://aidhog.github.io/owl2pb/.

Modelling hands

We now describe our model of poker, and the different types of hands that result.

Poker There are many variants of poker, but we are initially interested in the simplest case of trying to classify a hand of five cards drawn from a deck of fifty-two cards, comprising 13 ranks $(2, \ldots, K, A)$ and 4 suits $(\heartsuit, \clubsuit, \diamondsuit, \spadesuit)$.

Deck As a first step, we create an ontology with these 52 cards, 13 ranks and 4 suits, all declared pairwise distinct. This involves defining, for example, that $\mathsf{hasSuit}(A\spadesuit, \spadesuit)$ and $\mathsf{hasValue}(A\spadesuit, A)$. For convenience, and with a little punning, we also define, for example, that $A(A\spadesuit)$ and $\spadesuit(A\spadesuit)$, i.e., that ranks and suits are classes containing the cards with them.

We define the class Card as the pairwise distinct enumeration of the 52 cards:

$$\{2\heartsuit, \ldots, A\heartsuit, 2\diamondsuit, \ldots, A\diamondsuit, 2\spadesuit, \ldots, A\spadesuit, 2\clubsuit, \ldots, A\clubsuit\}\,.$$

Hands Next we define a Hand as a class with exactly five cards:

$$\mathsf{Hand} \equiv = 5\ \mathsf{hasCard.Card}$$

The hand types that follow are defined to be subclasses of Hand (omitted for brevity).

High card Next we define a HighCard as equivalent to Hand. This might seem unusual since in poker we consider high card to be the absence of any other hand, i.e., having five cards with distinct ranks that do not contain a straight (defined as having five consecutive ranks) nor a flush (defined as having five cards all of the same suit). However, in designing the ontology, making such hands disjoint would greatly increase complexity (something perhaps worthy of future work). Rather, when we have a hand that has, e.g., three-of-a-kind, we may consider it to also be a pair and a high card since it also "contains" such hands.

Pair This is defined as a hand that contains two cards with the same rank, as follows:

$$\mathsf{Pair} \equiv (\geq 2\ \mathsf{hasCard}.2) \sqcup \ldots \sqcup (\geq 2\ \mathsf{hasCard}.A)$$

Two Pair As one of the most complex definitions, this hand contains two pairs for two distinct ranks; to the best of our knowledge, the only way to capture this hand is as follows:

$$\mathsf{TwoPair} \equiv ((\geq 2\,\mathsf{hasCard}.2) \sqcap (\geq 2\,\mathsf{hasCard}.3)) \sqcup \ldots \sqcup ((\geq 2\,\mathsf{hasCard}.K) \sqcap (\geq 2\,\mathsf{hasCard}.A))$$

which is to say, the definition covers $\binom{13}{2} = 78$ possible disjuncts covering the two pairs.

Three-of-a-kind This is defined as a hand that contains three cards with the same rank:

$$\mathsf{ThreeOfAKind} \equiv (\geq 3\ \mathsf{hasCard}.2) \sqcup \ldots \sqcup (\geq 3\ \mathsf{hasCard}.A)$$

Straight A straight involves having five cards with five sequential ranks, such as 4, 5, 6, 7, 8. We define this as follows:

$$\mathsf{Straight} \equiv (\exists \mathsf{hasCard}.A \sqcap \ldots \sqcap \exists \mathsf{hasCard}.5) \sqcup \ldots \sqcup (\exists \mathsf{hasCard}.10 \sqcap \ldots \sqcap \exists \mathsf{hasCard}.A)$$

Flush A flush involves having five cards all of the same suit. We define this as follows:

$$\mathsf{Flush} \equiv (= 5\ \mathsf{hasCard}.\heartsuit) \sqcup (= 5\ \mathsf{hasCard}.\clubsuit) \sqcup (= 5\ \mathsf{hasCard}.\diamondsuit) \sqcup (= 5\ \mathsf{hasCard}.\spadesuit)$$

Full house As one of the more complex and paradigmatic definitions, this hand follows a similar pattern to that of the two-pair hand:

$$\begin{aligned}\mathsf{FullHouse} \equiv\ & ((\geq 2\ \mathsf{hasCard}.2) \sqcap (\geq 3\ \mathsf{hasCard}.3)) \sqcup ((\geq 2\ \mathsf{hasCard}.3) \\ & \sqcap (\geq 3\ \mathsf{hasCard}.2)) \sqcup \ldots \sqcup ((\geq 2\ \mathsf{hasCard}.K) \sqcap (\geq 3\ \mathsf{hasCard}.A))\end{aligned}$$

which is to say, the definition covers $2\binom{13}{2} = 156$ possible disjuncts covering all possible combinations of pair with three-of-a-kind.

Four-of-a-kind This is defined as a hand that contains four cards with the same rank:

$$\mathsf{FourOfAKind} \equiv (= 4\ \mathsf{hasCard}.2) \sqcup \ldots \sqcup (= 4\ \mathsf{hasCard}.A)$$

Straight flush This is a hand that combines a straight and a flush.

$$\begin{aligned}\mathsf{StraightFlush} \equiv\ & ((\exists \mathsf{hasCard}.A \sqcap \ldots \sqcap \exists \mathsf{hasCard}.5) \sqcup \\ & \ldots \sqcup (\exists \mathsf{hasCard}.10 \sqcap \ldots \sqcap \exists \mathsf{hasCard}.A)) \sqcap \\ & ((= 5\ \mathsf{hasCard}.\heartsuit) \sqcup (= 5\ \mathsf{hasCard}.\clubsuit) \sqcup (= 5\ \mathsf{hasCard}.\diamondsuit) \sqcup (= 5\ \mathsf{hasCard}.\spadesuit))\end{aligned}$$

Royal flush This is a straight flush with an ace and face cards.

$$\begin{aligned}\mathsf{RoyalFlush} \equiv\ & (\exists \mathsf{hasCard}.10 \sqcap \ldots \sqcap \exists \mathsf{hasCard}.A) \sqcap \\ & ((= 5\ \mathsf{hasCard}.\heartsuit) \sqcup (= 5\ \mathsf{hasCard}.\clubsuit) \sqcup (= 5\ \mathsf{hasCard}.\diamondsuit) \sqcup (= 5\ \mathsf{hasCard}.\spadesuit))\end{aligned}$$

OWL 2 DL Limitations It might be tempting to count the number of ranks or suits that a hand has, for example, to define that a flush has one suit:

$$\mathsf{Flush} \equiv =\ 1\ \mathsf{hasCard} \circ \mathsf{hasSuit}.\top$$

While this would simplify many of the definitions herein, OWL 2 DL does not permit counting non-simple roles, such as role chains.

4. Preliminary Results

We have tested three reasoners using the OWL API [7], specifically HermiT [8], JFact[1] (based on FaCT++ [9]) and Openllet[2] (based on Pellet [10]). We generate four instances of all ten hand types and then measure the time taken to classify these instances. These results show that the reasoners struggle to deal with such tasks within a timeout of 30 minutes. For this reason, we looked for opportunities to simplify the definitions, as follows:

Full house Rather than listing 156 ($O(n^2)$) disjuncts, there are two options. The first is to observe that every rank has 0, 2 or 3 cards in a full house, giving the following definition:

$$\begin{aligned}\mathsf{FullHouse} \equiv ((\leq 3\ \mathsf{hasCard}.2) \sqcap (\neq 1\ \mathsf{hasCard}.2)) \sqcap \ldots \sqcap \\ ((\leq 3\ \mathsf{hasCard}.A) \sqcap (\neq 1\ \mathsf{hasCard}.A))\end{aligned}$$

Since a hand contains exactly five cards, restricting each rank to occur zero, two or three times forces exactly one pair and one three-of-a-kind. The second is to use an intersection:

$$\mathsf{FullHouse} \equiv \mathsf{TwoPair} \sqcap \mathsf{ThreeOfAKind}$$

Straight flush Likewise, the fabled straight flush is the intersection of two hands:

$$\mathsf{StraightFlush} \equiv \mathsf{Straight} \sqcap \mathsf{Flush}$$

With these simplifications, we manage to classify hands using HermiT, and some hands using JFact, but not Openllet.

Pair shortcut Given that reasoners find the given ontologies very challenging to classify, we apply a simplification in order to reduce the difficulty of the benchmark. In particular, we add a pair shortcut with the property hasPair, which we add as input assertions to hands that contain two or more cards of the same rank, taking the rank as a value. This pragmatic shortcut then allows us to simplify the definition of two pair to:

$$\mathsf{TwoPair} \equiv \geq 2\ \mathsf{hasPair}.\top$$

Simplified benchmark Applying the definition of full house as the intersection of two pair and three-of-a-kind, straight flush as the intersection of straight and flush, and the pair shortcut along with the simplified two-pair definition, we run the experiments again providing 24GB of RAM for the Java heap. We further vary the number of ranks in $\{6, 13, 19, 25, 31, 37\}$, and the number of suits in $\{4, 8, 12, 16\}$. Again, JFact failed on most classification instances while Openllet failed on all instances. Hence we present the results for HermiT in Figure 1, which succeeded in all cases; the results show the total time to classify four instances of each of the ten hand types (40 hands total). The most costly hand to classify was four-of-a-kind, which took HermiT 12–463 seconds to classify four such instances, depending on the deck size.

[1] https://sourceforge.net/projects/jfact/; retr. 2026/07/31
[2] https://github.com/Galigator/openllet; retr. 2026/07/31

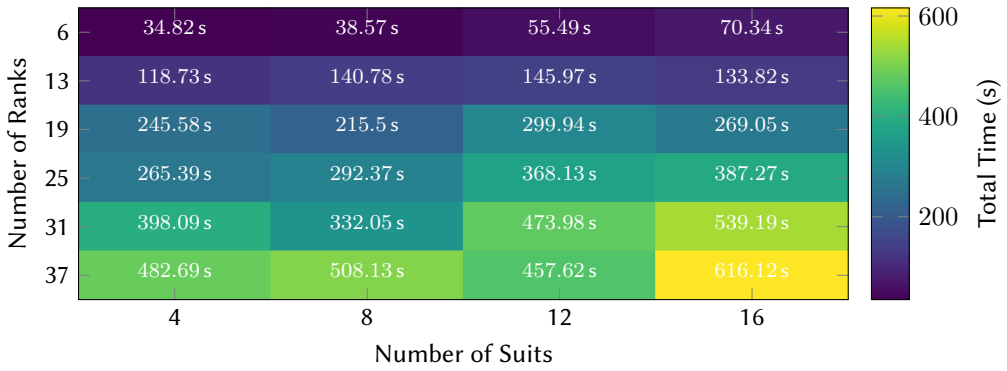


Figure 1: Total classification time for HermiT over 40 hand instances (4 of each of the 10 hand types).

5. Discussion & Next Steps

The preliminary results presented herein suggest that poker presents a challenging scenario for OWL 2 DL / $\mathcal{SROIQ}$ reasoners, further presenting opportunities for testing equivalent types of definitions, and for testing performance on a parameterisable T-Box. We thus believe that the benchmark can lead to a better understanding of the performance of such reasoners. Indeed, the classification of poker hands is not inherently a difficult task – custom code could classify such hands in milliseconds – which raises the question of where the underlying complexity arises, and what could be done to optimise such cases.

We are working on advancing the benchmark. In future work, we plan to consider parameterising the hand size to further adjust the difficulty. However, this may require also updating the kinds of hands in a non-trivial way, for example, to permit five-of-a-kind, or to ensure that the cardinality restrictions for flush are equal to the hand size. It might also be of interest to try to define disjoint hands, such that full house is not, for example, a subclass of pair. A more general direction for future work is to try to understand why certain classification tasks are so challenging for particular reasoners, for example, why HermiT performs worst in the case of four-of-a-kind. This may be difficult to ascertain given the complexity of the underlying reasoning algorithms, but would help to gain key insights into performance bottlenecks and potential optimisations.

Acknowledgements Funded by ANID – Millennium Science Initiative Program – Code ICN17 002, Chile.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT (GPT-6) in order to check grammar and spelling, to assist in translating benchmark materials from Spanish to English, and to assist in documenting the benchmark. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

[1] I. Horrocks, O. Kutz, U. Sattler, The even more irresistible SROIQ, in: P. Doherty, J. Mylopoulos, C. A. Welty (Eds.), Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006, AAAI Press, 2006, pp. 57–67. URL: https://aaai.org/papers/KR06-009-the-even-more-irresistible-sroiq/.

[2] Y. Guo, Z. Pan, J. Heflin, LUBM: A benchmark for OWL knowledge base systems, J. Web Semant. 3 (2005) 158–182. URL: https://doi.org/10.1016/j.websem.2005.06.005. doi:10.1016/J.WEBSEM.2005.06.005.

[3] L. Ma, Y. Yang, Z. Qiu, G. T. Xie, Y. Pan, S. Liu, Towards a Complete OWL Ontology Benchmark, in: Y. Sure, J. Domingue (Eds.), The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings, volume 4011 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 125–139. URL: https://doi.org/10.1007/11762256_12. doi:10.1007/11762256_12.

[4] V. Link, S. Lohmann, F. Haag, OntoBench: Generating Custom OWL 2 Benchmark Ontologies, in: P. Groth, E. Simperl, A. J. G. Gray, M. Sabou, M. Krötzsch, F. Lécué, F. Flöck, Y. Gil (Eds.), The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II, volume 9982 of *Lecture Notes in Computer Science*, 2016, pp. 122–130. URL: https://doi.org/10.1007/978-3-319-46547-0_13. doi:10.1007/978-3-319-46547-0_13.

[5] G. Singh, S. Bhatia, R. Mutharaju, OWL2Bench: A Benchmark for OWL 2 Reasoners, in: J. Z. Pan, V. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, L. Kagal (Eds.), The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II, volume 12507 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 81–96. URL: https://doi.org/10.1007/978-3-030-62466-8_6. doi:10.1007/978-3-030-62466-8_6.

[6] N. F. Noy, N. H. Shah, P. L. Whetzel, B. Dai, M. Dorf, N. Griffith, C. Jonquet, D. L. Rubin, M. D. Storey, C. G. Chute, M. A. Musen, Bioportal: ontologies and integrated data resources at the click of a mouse, Nucleic Acids Res. 37 (2009) 170–173. URL: https://doi.org/10.1093/nar/gkp440. doi:10.1093/NAR/GKP440.

[7] M. Horridge, S. Bechhofer, The OWL API: A Java API for OWL Ontologies, Semantic Web 2 (2011) 11–21.

[8] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, Z. Wang, HermiT: An OWL 2 Reasoner, J. Autom. Reason. 53 (2014) 245–269. URL: https://doi.org/10.1007/s10817-014-9305-1. doi:10.1007/S10817-014-9305-1.

[9] D. Tsarkov, I. Horrocks, FaCT++ Description Logic Reasoner: System Description, in: U. Furbach, N. Shankar (Eds.), Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings, volume 4130 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 292–297. URL: https://doi.org/10.1007/11814771_26. doi:10.1007/11814771_26.

[10] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, J. Web Semant. 5 (2007) 51–53. URL: https://doi.org/10.1016/j.websem.2007.03.004. doi:10.1016/J.WEBSEM.2007.03.004.