Semantic Knowledge Partitioning

Ph.D. Project, Extended Abstract

Christoph Wernhard

Institut für Informatik, Universität Koblenz-Landau, D-56070 Koblenz, Germany,
`wernhard@uni-koblenz.de`

1 Introduction

We investigate some possibilities of decomposing a knowledge base in a way such that only its semantics determines the results. We regard a knowledge base as represented by a logic formula and confine ourselves here to classical propositional logic and classical first order clausal logic. A decomposition operation takes a formula as input and returns one or several other formulas that are in some sense "parts" of the original one. We plan to work out such operations, develop algorithms for them, show application scenarios and provide a prototypical implementation. "Semantics" is understood in the standard sense for these logics, for our purposes it is however sufficient to consider just the framework of Herbrand interpretations.

From an application point of view, a knowledge base — possibly large and possibly created by someone else than its user — should be accessed through semantic query functions, i.e. an interface consisting of functions that are characterized solely by the semantics of the knowledge base and query expression.[1] Some types of such queries can be straightforwardly expressed as theorem proving tasks, others require different modes of reasoning such as model generation, decision procedures, abduction, partial deduction or knowledge compilation. Semantic knowledge partitioning can be understood as such a reasoning mode. Outputs can be again inputs to reasoning in this or some other mode.

Application relevant operations of the following kinds can be expressed using semantic knowledge partitioning:

- Extraction of an application relevant portion from a large knowledge base.
- Analysis of a knowledge base: Computing definitional dependency of predicates and functions. Computing possibilities to decompose it into mutually independent sentences.
- Expressing a theorem in a certain signature, matching the level of abstraction required by an application.
- Removing information from an answer for security reasons.

It is also a goal of this work to provide a general notion for semantic based partitioning which can be applied to other languages by embedding them into first order logic — for example a description logic.

[1] Such a "semantic" view of a knowledge base has been already suggested in the context of the KRYPTON system [6].

2 Notation and Preliminaries

Syntactic Framework for First Order Logic. We consider first order formulas in Skolemized clausal form: A *formula* (or *matrix*) is a set of clauses; a *clause* is a set of literals; a *literal* is a pair of an atom and a sign. If A is an atom, we write $+A$ ($-A$) for the positive (negative) literal with atom A. The *dual* of a literal L is written $\overline{L}$.

Semantic Framework. We use a semantic framework based on Herbrand interpretations. An *interpretation* is a set of ground literals that contains for each ground atom A exactly one of $+A$ or $-A$ and no other elements. If $+A$ $(-A) \in I$, we say that I *assigns* A the value *true* (*false*). For interpretation I and ground literal L $I[L]$ denotes $(I - \{\overline{L}\}) \cup \{L\}$. A clause C *contradicts* an interpretation if and only if there is a ground instance C' of C such that the dual of each literal in C' is in I. An interpretation I is a *model* for a formula F, in symbols $I \models F$, if and only if F contains no clause that contradicts I. A formula F *implies* a formula G, in symbols $F \Rightarrow G$, if and only if for all interpretations I holds that if $I \models F$ then $I \models G$. Two formulas F and G are *equivalent*, in symbols $F \Leftrightarrow G$, if and only if $F \Rightarrow G$ and $G \Rightarrow F$.

Base, Base of a Formula, Essential Base of a Formula. We call a set of ground atoms a *base*. The *base of a formula* F, in symbols $\mathcal{B}(F)$, is the set of all ground instances of atoms in the F formed by instantiation with all constants and function symbols in the formula, and, if the formula contains no constant, the constant *sk*, which is reserved for this purpose. The *essential base of a formula* F, in symbols $\mathcal{E}(F)$, is the set of all ground atoms A such that there exists an interpretation I for which exactly one of $I[+A] \models F$ or $I[-A] \models F$ does hold. For example $\mathcal{E}(p \wedge (q \vee \neg q)) = \{p\}$.

3 Model Relative to a Base

The notion of *model relative to a base* is the central concept in this work. It is used to specify the partitioning operations semantically. It also provides semantics for certain formula transformations which do not preserve equivalence. Some of these transformations are well known in the context of theorem proving, where the fact that they preserve satisfiability is used. Since we are interested in knowledge base preprocessing and model generation, we make use of the stronger properties that also models relative to certain bases are preserved.

Definition 1 (Model Relative to a Base). For all interpretations I, formulas F and bases B, *I is a model of F relative to B*, in symbols $I \models_B F$, if and only if there exists an interpretation I' such that $I' \models F$ holds and I and I' assign the same values to all atoms in B.

Definition 2 (Equivalence Relative to a Base). For all formulas F, G and bases B, *F and G are equivalent relative to B*, in symbols $F \Leftrightarrow_B G$, if and only if for all interpretations I holds that $I \models_B F$ if and only if $I \models_B G$.

4 Projection and Forgetting

Projection and *forgetting* are the core operations for semantic knowledge partitioning. They are defined as ternary relationships $\mathsf{project}(F, B, F')$ and $\mathsf{forget}(F, B, F')$, where F and F' are formulas and B is a base. F and B are considered as input and F' as result.

The intuition of projection is that F' is equivalent to F with respect to the atoms in B and provides no information about other atoms (like a databases relation can be "projected" onto a given set of attributes). The intuition of forgetting is that F' is equivalent to F, except that it does not provide any information about the atoms in B (B is "forgotten"). Projection and forgetting can be defined in terms of each other: $\mathsf{project}(F, B, F')$ is equivalent to $\mathsf{forget}(F, \mathcal{E}(F) - B, F')$. $\mathsf{forget}(F, B, F')$ is equivalent to $\mathsf{project}(F, \mathcal{E}(F) - B, F')$. In these definitions, instead of the essential base operator $\mathcal{E}$, equivalently the base operator $\mathcal{B}$, which is easier to handle computationally, can be used.

Definition 3 (Projection). For all formulas F, F' and bases B, F' is a *projection of F onto B*, in symbols $\mathsf{project}(F, B, F')$, if and only if for all interpretations I holds that $I \models F'$ if and only if $I \models_B F$.

An alternate but equivalent definition of projection is: $\mathsf{project}(F, B, F')$ holds if and only if $\mathcal{E}(F') \subseteq B$ and $F \Leftrightarrow_B F'$. Based on this definition forgetting can be defined as follows:

Definition 4 (Forgetting). For all formulas F, F' and bases B, F' is a *forgetting of B on F*, in symbols $\mathsf{forget}(F, B, F')$, if and only if the following three conditions hold: 1. $F \Leftrightarrow_{\mathcal{E}(F)-B} F'$, 2. $\mathcal{E}(F') \cap B = \emptyset$, and 3. $\mathcal{E}(F') \subseteq \mathcal{E}(F)$.

For propositional formulas F and atoms A a function $\mathsf{forget}(F, A)$ such that $\mathsf{forget}(F, \{A\}, \mathsf{forget}(F, A))$ holds can be defined: $\mathsf{forget}(F, A) = F[A \backslash \mathsf{true}] \vee F[A \backslash \mathsf{false}]$ where $F[A \backslash G]$ denotes F with all occurrences of A substituted by G. $\mathsf{forget}(F, A)$ can also be computed by the following method: Convert F into disjunctive normal form, remove all clauses that contain complementary literals (so far equivalence to F is preserved) and then remove all literals with atom A. A function $\mathsf{project}(F, B)$ that satisfies $\mathsf{project}(F, B, \mathsf{project}(F, B))$ can be defined as $\mathsf{forget}(...\mathsf{forget}(\mathsf{forget}(F, A_1), A_2), ..., A_n)$ where $\{A_1, A_2, ...A_n\} = \mathcal{B}(F) - B$.

5 Constructions with Projection

We describe some constructions with projection that can be used in the applications outlined in the introduction. We restrict ourselves here to propositional logic.

Justification of Knowledge Base Preprocessing. For all formulas F and G holds that $F \Rightarrow G$ if and only if $\mathsf{project}(F, \mathcal{B}(G)) \Rightarrow G$. So the result of computing $\mathsf{project}(F, \mathcal{B}(G))$ can be regarded as a preprocessed knowledge base that can be used to answer queries whose base is a subset of $\mathcal{B}(G)$.

Determining Definability and Constructing Definitions. If F is formula, A an atom and B is a set of ground atoms not containing A, then A is called *definable in terms of* B *within* F if and only if F implies a formula $A \leftrightarrow D$ where $\mathcal{B}(D) \subseteq B$. Definability can be expressed using projection: A is definable in terms of B within F if and only if $\mathsf{project}(F \land A,\ B) \land \mathsf{project}(F \land \neg A,\ B)$ is unsatisfiable. If definability holds for F,A and B, then definitions can be constructed in two ways: 1. $A \leftrightarrow \mathsf{project}(F \land A,\ B)$ and 2. $A \leftrightarrow \neg\mathsf{project}(F \land \neg A,\ B)$.

Lossless Decomposition. Two bases B_1 and B_2 are a *lossless decomposition* of a formula F if and only if $\mathsf{project}(F, B_1) \land \mathsf{project}(F, B_2) \Rightarrow F$.

6 Algorithm: Replace-by-Resolvents

We show an algorithmic way for constructing the forgetting of all ground instances of a given linear atom pattern P on a given first order matrix M.

Theorem 1. *For all matrices M, clauses C and literals L with $L \in C$ and $C \in M$ holds that if each resolvent*[2] *over literal L and clause C with a clause in M is tautological or subsumed by a clause in $M - \{C\}$ and B is a base not containing an instance of the atom of L, then $M \Leftrightarrow_B M - \{C\}$.*

Definition 5 (Replace-by-Resolvents). $\mathsf{replace\text{-}by\text{-}resolvents}(M, L, C)$ is a procedure that takes a matrix M, clause C and literal L with $L \in C$ and $C \in M$ as input. Let M' be the saturation of M with respect to resolvent generation over L and C.[3] Return $M' - \{C\}$.

The input matrix M can contain a literal with an atom that is unifiable with pattern P but not an instance of it. Direct application of replace-by-resolvents to such a matrix will yield unintended results. An equivalence preserving preprocessing transformation corrects this: Each clause C in M containing such a literal is replaced by the set of its most general instances that has the same Herbrand expansion as $\{C\}$ for the signature of M and does not contain such literals.[4]

So far we can compute the forgetting of M and P by first applying the instantiation preprocessing transformation and then successively apply replace-by-resolvents to each literal with an atom that is an instance of P. If this procedure terminates, the result is indeed the supposed forgetting: Theorem 1 ensures condition (1.) of proposition 4, condition (2.) is obviously satisfied and condition (3.) follows from conditions (1.) and (2.) and the fact that $\mathcal{B}(F') \subseteq \mathcal{B}(F)$.

---

[2] A *resolvent* is understood here as a binary resolvent from factors of the parents [7].

[3] I.e. compute a resolvent D over L and C with some other clause from M. If D is not tautological and not subsumed by $M - \{C\}$ then add it to M. Repeat this until no more resolvent D that is not tautological and not subsumed by $M - \{C\}$ can be computed. Return M.

[4] An alternate method would be the use of disequality constraints. But then symbols to be "forgotten" are preserved "negatively" in the constraints and the straightforward framework of first order logic is left.

If we allow a relaxation of condition (3.) we can transform M in a preprocessing step into a form where each clause contains only a single literal whose atom is unifiable with P by introducing auxiliary definitional literals. Replace-by-resolvents terminates for all M in this form. Auxiliary literals can afterwards be removed as far as possible by equivalence preserving reductions and by reductions which preserve models relative to bases not containing atoms of the auxiliary literals, for example the *PURE*, *ISOL* and *ISOL** [4], which are special cases of replace-by-resolvents.

7 Related Work

Replace-by-resolvents may be regarded as a variant of the SCAN algorithm for second order quantifier elimination [9, 1]. Lin and Reiter [11] describe the forgetting operation semantically. In the context of knowledge compilation applications of projection are discussed [8]. Aside from DNF there are other normal forms for propositional logic such as DNNF and OBDD in which forgetting is an operation of low complexity. Forgetting can be used to simulate the original Davis-Putnam procedure. A refinement of this procedure has been lifted to first order logic [3]. Projection can be considered as a special case of marginalization, one of the fundamental operations in the "local computation" framework [10, 12]. Partition-based logical reasoning [2] is a framework for distributed theorem proving based on consequence finding in signature restricted knowledge base partitions. The proof of the Craig interpolation theorem by Dreben and Putnam [5] involves propositional forgetting.

References

1. W. Ackermann. Untersuchungen über das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110:390–413, 1935.
2. E. Amir and S. A. McIlraith. Partition-based logical reasoning. In *Principles of Knowledge Representation and Reasoning*, pages 389–400, 2000.
3. P. Baumgartner. FDPLL – A First-Order Davis-Putnam-Logeman-Loveland Procedure. In *CADE-17*, pages 200–219, 2000.
4. W. Bibel. *Deduktion.* R. Oldenbourg, München, 1992.
5. G. S. Boolos and R. C. Jeffrey. *Computability and Logic.* Cambridge University Press, Cambridge, third edition, 1989.
6. R. J. Brachman, R. E. Fikes, and H. J. Levesque. KRYPTON: A functional approach to knowledge representation. *IEEE COMPUTER*, 16(10):67–73, Oct. 1983.
7. C.-L. Chang and R. C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, Boston, 1973.
8. A. Darwiche. Decomposable negation normal form. *JACM*, 48(4):608–647, 2001.
9. D. Gabbay and H. J. Ohlbach. Quantifier elimination in second-order predicate logic. In B. Nebel et al., editors, *KR'92*, pages 425–435, San Mateo, Calif., 1992.
10. J. Kohlas, R. Haenni, and S. Moral. Propositional information systems. *Journal of Logic and Computation*, 9(5):651–681, 1999.
11. F. Lin and R. Reiter. Forget It! In R. Greiner et al., editors, *Working Notes, AAAI Fall Symposium on Relevance*, pages 154–159, Menlo Park, Calif., 1994.
12. N. Wilson and J. Mengin. Logical deduction using the local computation framework. In *FTP'97*, RISC-Linz Report No. 97–50, pages 135–139. Linz, 1997.