Literal Projection for First-Order Logic

Christoph Wernhard

Universität Koblenz-Landau
wernhard@uni-koblenz.de

Abstract. The computation of literal projection generalizes predicate quantifier elimination by permitting, so to speak, quantifying upon an arbitrary sets of ground literals, instead of just (all ground literals with) a given predicate symbol. Literal projection allows, for example, to express predicate quantification upon a predicate just in positive or negative polarity. Occurrences of the predicate in literals with the complementary polarity are then considered as unquantified predicate symbols. We present a formalization of literal projection and related concepts, such as literal forgetting, for first-order logic with a Herbrand semantics, which makes these notions easy to access, since they are expressed there by means of straightforward relationships between sets of literals. With this formalization, we show properties of literal projection which hold for formulas that are free of certain links, pairs of literals with complementary instances, each in a different conjunct of a conjunction, both in the scope of a universal first-order quantifier, or one in a subformula and the other in its context formula. These properties can justify the application of methods that construct formulas without such links to the computation of literal projection. Some tableau methods and direct methods for second-order quantifier elimination can be understood in this way.

1 Introduction

Predicate quantifier elimination has a large variety of applications in knowledge processing, which continue to become apparent since the early nineties until very recently [1–8]. This is sometimes not obvious, because operations such as the computation of uniform interpolants, forgetting and projection are in fact variants of predicate quantifier elimination. In parallel to discovering applications, the development of methods that perform predicate quantifier elimination in first-order and related logics has been of continued interest since the early nineties [1, 3, 9, 8]. In recent years also predicate quantifier elimination in propositional logic became a subject of research, driven largely by advances in SAT-solving [4, 5, 10–12].

In this paper we focus on *literal projection*, which generalizes predicate quantification by permitting, so to speak, quantifying upon an *arbitrary set of ground literals*, instead of just (all ground literals with) a given predicate symbol. Literal projection allows, for example, to express predicate quantification upon a predicate just in positive or negative polarity. Eliminating such a quantifier from a formula in negation normal form results in a formula that might still contain

the quantified predicate, but only in literals whose polarity is complementary to the quantified one. The result of wrapping a formula into such an existential quantifier, followed by eliminating it, has – among the formulas that do not contain the quantified predicate in the quantified polarity – exactly the same theorems as the original formula. The result can be considered as an extract of the original formula, where knowledge about the quantified predicate in the quantified polarity is "forgotten", but all other knowledge is retained. A look over the 40000 theorems in the Mizar mathematical library, the largest collection of formalized mathematical knowledge, indicates the order of magnitude of the role of asymmetric polarity in actual knowledge bases: In 98.2 percent of the theorems, at least one predicate symbol occurs only in a single polarity. In 66.9 percent, no predicate symbol occurs in both polarities. On average, 88.9 percent of the predicate symbols in the signature of a theorem occur in the theorem only in a single polarity.[1]

Literal forgetting, a variant of literal projection, has been formalized for propositional logic in [11]. We generalize this formalization to first-order logic with Herbrand interpretations. A new formulation of the characterization in [11] facilitates the formal access to literal projection and related notions, which then can be expressed by means of straightforward relationships between sets of literals. With this formalization, we show some properties of literal projection which hold for formulas that are free of certain *links*, pairs of literals with complementary instances, each in a different conjunct of a conjunction, both in the scope of a universal first-order quantifier, or one in a subformula and the other in its context formula. These properties can justify the application of methods that construct formulas without such links to predicate quantifier elimination, or, more generally, to the computation of literal projection. Some tableau construction procedures and direct methods [8] for second-order quantifier elimination can be understood in this way.

The structure of the paper is as follows: In Sect. 2 the semantic framework with the characterization of literal projection is defined and illustrated by some examples. Further related concepts are defined in Sect. 3, in particular a formal account of the set of literals "about which a formula expresses something." In Sect. 4 basic properties of literal projection are summarized, properties that relate to linklessness and conjunction are developed, and their application to computation methods is outlined. A further property that relates to linklessness between a subformula and its context is discussed in Sect. 5. In the conclusion, applications of literal projection in knowledge-based systems are suggested.

A short version of this paper has been published as [14].

[1] These statistics have been obtained with the MPTP 0.2 translation of the Mizar library [13]. About 1000 theorems which have just equality as predicate or translate to *true* have not been considered. Predicates that are only implicit in the original Mizar syntax (*modes*, *attributes* and *aggregates*), as well as predicate occurrences in set abstractions and type specifiers, have been taken into account.

2 Semantic Framework, Projection and Forgetting

Notation. We write the positive (negative) literal with atom A as $+A$ ($-A$). We understand a *first-order formula* as in negation normal form, constructed from literals, truth value constants $\top, \bot$, binary connectives $\wedge, \vee$, and quantifiers $\forall, \exists$. Implication $\rightarrow$, negation $\neg$ and dropping the sign of positive literals are understood as meta-level notation with respect to this syntax. We assume a fixed first-order signature with at least one constant symbol. The sets of all ground terms and all ground atoms, with respect to this signature, are denoted by GTERMS and GATOMS. Variables are x, y, z, also with subscripts. To avoid clumsy handling of quantifier scopes, we assume that in a formula all occurrences of the same variable are either free or are bound by an occurrence of a quantifier, and that no two quantifier occurrences bind occurrences of the same variable.

The Projection Operator and Literal Scopes. A *formula* in general is like a first-order formula, but in its construction a further operator, $\mathsf{project}(F, S)$, is permitted, where F is a formula and S specifies a set of ground literals. We call a set of ground literals in the role as argument to $\mathsf{project}$ a *literal scope*. The formula $\mathsf{project}(F, S)$ is called the *literal projection* of F onto S. Literal projection generalizes existential second-order quantification. It is further discussed below.

Interpretations. We characterize the semantics with a notational variant of the framework of Herbrand interpretations: An *interpretation* is a pair $\langle I, \beta \rangle$, where I is a *structure*, that is, a set of ground literals that contains for all ground atoms A exactly one of $+A$ or $-A$, and β is a *variable assignment*, that is, a mapping of the set of variables into GTERMS.

Satisfaction Relation. The satisfaction relation between interpretations $\langle I, \beta \rangle$ and formulas is defined by the clauses in Tab. 1, where L matches a literal, F, F_1, F_2 match a formula, and S matches a literal scope specifier. Two operations are defined on variable assignments β : If F is a formula, then $F\beta$ denotes F with all variables replaced by their image in β. If x is a variable and t a ground term, then $\beta\frac{t}{x}$ is the variable assignment that maps x to t and all other variables to the same values as β.

The semantic definition of literal projection in Tab. 1 gives a formal account of the following more intuitive characterization: An interpretation $\langle I, \beta \rangle$ satisfies $\mathsf{project}(F, S)$ if and only if there is a structure J such that $\langle J, \beta \rangle$ satisfies F and I can be obtained from J by replacing literals that are not in S with their complements. This includes the special case $I = J$, where no literals are replaced.

Entailment and equivalence can be straightforwardly defined in terms of the satisfaction relation: A formula F_1 *entails* a formula F_2, in symbols $F_1 \models F_2$, if and only if for all interpretations $\langle I, \beta \rangle$ it holds that if $\langle I, \beta \rangle \models F_1$ then $\langle I, \beta \rangle \models F_2$. A formula F_1 is *equivalent* to a formula F_2, in symbols $F_1 \equiv F_2$, if and only if $F_1 \models F_2$ and $F_2 \models F_1$.

Table 1. The Satisfaction Relation with the Semantic Definition of Literal Projection

$\langle I,\beta\rangle \models \top$		
$\langle I,\beta\rangle \not\models \bot$		
$\langle I,\beta\rangle \models L$	iff_{def}	$L\beta \in I$
$\langle I,\beta\rangle \models F_1 \wedge F_2$	iff_{def}	$\langle I,\beta\rangle \models F_1$ and $\langle I,\beta\rangle \models F_2$
$\langle I,\beta\rangle \models F_1 \vee F_2$	iff_{def}	$\langle I,\beta\rangle \models F_1$ or $\langle I,\beta\rangle \models F_2$
$\langle I,\beta\rangle \models \forall x\ F$	iff_{def}	for all $t \in \mathsf{GTERMS}$ it holds that $\langle I,\beta\frac{t}{x}\rangle \models F$
$\langle I,\beta\rangle \models \exists x\ F$	iff_{def}	there exists a $t \in \mathsf{GTERMS}$ such that $\langle I,\beta\frac{t}{x}\rangle \models F$
$\langle I,\beta\rangle \models \mathsf{project}(F,S)$	iff_{def}	there exists a structure J such that $\langle J,\beta\rangle \models F$ and $J \cap S \subseteq I$

Relation to Conventional Model Theory. Literal sets as components of interpretations permit the straightforward definition of the semantics of literal projection given in the last clause in Tab. 1. The set of literals I of an interpretation $\langle I,\beta\rangle$ is called *"structure"*, since it can be considered as representation of a structure in the conventional sense used in model theory: The domain is the set of ground terms. Function symbols f with arity $n \geq 0$ are mapped to functions f' such that for all ground terms $t_1,...,t_n$ it holds that $f'(t_1,...,t_n) = f(t_1,...,t_n)$. Predicate symbols p with arity $n \geq 0$ are mapped to $\{\langle t_1,...,t_n\rangle \mid +p(t_1,...,t_n) \in I\}$. Moreover, an interpretation $\langle I,\beta\rangle$ represents a conventional second-order interpretation [15] (if predicate variables are considered as distinguished predicate symbols): The structure in the conventional sense corresponds to I, as described above, except that mappings of predicate variables are omitted. The assignment is β, extended such that all predicate variables p are mapped to $\{\langle t_1,...,t_n\rangle \mid +p(t_1,...,t_n) \in I\}$.

Some More Notation. If L is a literal, S is a literal scope, I is an interpretation, F is a formula, x is a variable, and t is a term, then: $\widetilde{L}$ denotes the *complement* of L; $\widetilde{S} \stackrel{\text{def}}{=} \{\widetilde{L} \mid L \in S\}$; $\overline{S} \stackrel{\text{def}}{=} \mathsf{GLITS} - S$; S is called *consistent* if it does not contain a literal and its complement; $I[L] \stackrel{\text{def}}{=} (I - \{\widetilde{L}\}) \cup \{L\}$; $I[S] \stackrel{\text{def}}{=} (I - \widetilde{S}) \cup S$; $F\{x \mapsto t\}$ is F with all occurrences of x replaced by t.

Literal Forgetting. In some applications it is natural to consider projection onto all literals *with exception* of those in a given set. The concept of *forgetting* allows to express this conveniently: *The literal forgetting in F about S*, in symbols $\mathsf{forget}(F,S)$, is defined by $\mathsf{forget}(F,S) \stackrel{\text{def}}{=} \mathsf{project}(F,\overline{S})$.

Formulation Variants of Literal Projection and Forgetting. The phrase $J \cap S \subseteq I$ in the semantic definition of literal projection (Tab. 1) can be expressed in a variety of equivalent formulations, shown as (P1)–(P5) in Tab. 2. Adaptions of these formulations to literal forgetting are shown as (F1)–(F5). The two characterizations of literal forgetting in [11, Prop. 14] correspond to (F4) and (F5).[2]

[2] Seemingly there is a bug in [11, Prop. 14]: the second characterization should probably be $Mod(\Sigma) \cup \{\omega \mid Force(\omega,l) \models \Sigma\}$ instead of just $\{\omega \mid Force(\omega,l) \models \Sigma\}$.

Table 2. Formulation Variants of Literal Projection and Forgetting

(P1) $J \cap S \subseteq I$.	(F1) $J \cap \overline{S} \subseteq I$.
(P2) $J \subseteq I \cup \overline{S}$.	(F2) $J \subseteq I \cup S$.
(P3) $I \cap \widetilde{S} \subseteq J$.	(F3) $I \subseteq J \cup \widetilde{S}$.
(P4) There exists a literal scope S' such that $S' \cap S = \emptyset$ and $J = I[S']$.	(F4) There exists a literal scope S' such that $S' \subseteq S$ and $J = I[S']$.
(P5) There exists a literal scope S' such that $S' \cap \widetilde{S} = \emptyset$ and $I = J[S']$.	(F5) There exists a literal scope S' such that $S' \subseteq \widetilde{S}$ and $I = J[S']$

Atom Projection and Forgetting. In the special case where the literal scope S is equal to $\widetilde{S}$, we speak of *atom projection* and *atom forgetting*. The condition $J \cap S \subseteq I$ in the semantic definition of $\mathsf{project}$ is then equivalent to $I \cap S = J \cap S$. Existential second-order quantification can be expressed in terms of atom forgetting: $\exists p\, F$ corresponds to $\mathsf{forget}(F, \{L \mid L \textit{ is a ground literal with predicate } p\})$.

By the way, at least for propositional logic, it is also possible to define literal forgetting in terms of atom forgetting, since for propositional formulas F and ground literals L it holds that $\mathsf{forget}(F, \{L\}) \equiv (\mathsf{forget}(L \wedge F, \{L, \widetilde{L}\}) \vee (\widetilde{L} \wedge F))$.

Example 1 (Forgetting a Negative Literal). Let $F \stackrel{\text{def}}{=} ((p \rightarrow q) \wedge (q \rightarrow r))$ and $S \stackrel{\text{def}}{=} \{+p, -p, +q, +r, -r\}$. We now illustrate that $\mathsf{project}(F, S) \equiv ((p \rightarrow q) \wedge (p \rightarrow r))$. It is not hard to see that the models of F are exactly those interpretations whose structures are a superset of at least one of $M_1, ..., M_4$, defined as shown in Tab. 3.(i). Thus the equations in Tab. 3.(ii) hold. By the semantic definition of literal projection, $\mathsf{project}(F, S)$ is a formula whose models are exactly the interpretations which are a superset of at least one of the $M_i \cap S$, for $i \in \{1, ..., 4\}$. It is easy to see that this condition on interpretations, being a superset of at least one of the $M_i \cap S$, is equivalent to being a superset of M_1' or M_2', defined as in Tab. 3.(iii). It is not hard to see that the models of $((p \rightarrow q) \wedge (p \rightarrow r))$ are exactly the interpretations that satisfy this condition.

Example 2 (Forgetting a Positive Literal). Let F be defined as in Examp. 1 and $S \stackrel{\text{def}}{=} \{+p, -p, -q, +r, -r\}$. Thus, S is as in Examp. 1, except that it contains q negatively instead of positively. Analogously to Examp. 1, it can be shown that $\mathsf{project}(F, S) \equiv ((p \rightarrow r) \wedge (q \rightarrow r))$, where, if $M_1, ..., M_4$ are defined as in Examp. 1, the equations in Tab. 3.(iv) hold.

Table 3. Examples of Literal Projection

(i) $M_1 \stackrel{\text{def}}{=} \{+p, +q, +r\}$, $M_2 \stackrel{\text{def}}{=} \{-p, +q, +r\}$, $M_3 \stackrel{\text{def}}{=} \{-p, -q, +r\}$, $M_4 \stackrel{\text{def}}{=} \{-p, -q, -r\}$.	(ii) $M_1 \cap S = \{+p, +q, +r\}$, $M_2 \cap S = \{-p, +q, +r\}$, $M_3 \cap S = \{-p, +r\}$, $M_4 \cap S = \{-p, -r\}$.	(iii) $M_1' \stackrel{\text{def}}{=} \{+p, +q, +r\}$, $M_2' \stackrel{\text{def}}{=} \{-p\}$.

(iv) $M_1 \cap S = \{+p, +r\}$, $M_2 \cap S = \{-p, +r\}$, $M_3 \cap S = \{-p, -q, +r\}$, $M_4 \cap S = \{-p, -q, -r\}$.	(v) $M_1 \cap S = \{+p, +r\}$, $M_2 \cap S = \{-p, +r\}$, $M_3 \cap S = \{-p, +r\}$, $M_4 \cap S = \{-p, -r\}$.

Example 3 (Forgetting an Atom). Let F be defined as in Examp. 1 and $S \stackrel{\text{def}}{=} \{+p, -p, +r, -r\}$. Thus, S is as in Examp. 1 and 2, except that it does not contain a literal with atom q. Analogously to Examp. 1, it can be shown that $\mathsf{project}(F, S) \equiv (p \rightarrow r)$, where, if $M_1, ..., M_4$ are defined as in Examp. 1, the equations in Tab. 3.(v) hold.

3 Essential Literal Base and Related Concepts

Literal Base and Essential Literal Base. The signature, that is, the set of predicate and function symbols, of a knowledge base hints the objects, concepts and relations about which it expresses "knowledge". But the signature might be too large: For example the formula $KB = (p \vee (q \wedge \neg q))$ is clearly equivalent to p. Thus KB does express something about p but not about q, although q is in its signature. One might argue that in practice such redundancies might be avoided by carefully engineering knowledge bases. But such redundancies may also arise by combining knowledge bases that are free of them. For example conjoining $(q \rightarrow p)$ with $(\neg q \rightarrow p)$ results in an equivalent to KB, which, as we have seen, does not express anything about q, although each of the conjuncts expresses something about q. We call the set of ground literals "about which a formula expresses something" its *essential literal base*, made precise in Def. 2. The essential literal base of KB, for example, is $\{p\}$.

Definition 1 (Literal Base). The *literal base* of a formula F, in symbols $\mathcal{L}(F)$, is the set of ground instances of literals in F.

Definition 2 (Essential Literal Base). The *essential literal base* of a formula F, in symbols $\mathcal{L}_{\mathcal{E}}(F)$, is defined as $\mathcal{L}_{\mathcal{E}}(F) \stackrel{\text{def}}{=} \{L \mid L \text{ is a ground literal and there exist an interpretation } \langle I, \beta\rangle \text{ such that } \langle I, \beta\rangle \models F \text{ and } \langle I[\widetilde{L}], \beta\rangle \not\models F\}$.

The essential literal base of a formula is a subset of its literal base. The essential literal base is independent of syntactic properties: equivalent formulas have the same essential literal base.[3]

Switching Values Outside the [Essential] Literal Base. Proposition 1 below states a property of literal bases that is useful to prove further properties: From a given model of a formula another model can be obtained by switching only literals not in the literal base of the formula. The precondition $S \cap \mathcal{L}(F) = \emptyset$ is equivalent to $\widetilde{S} \cap \widetilde{\mathcal{L}(F)} = \emptyset$. This suggests a second way to read the proposition: another model can be obtained by switching literals in a way such that none of the new values is complementary to an element of the literal base.

[3] For propositional formulas, *literal base* and *essential literal base* are defined in [11], called the sets of literals on which a formula is *syntactically* (*semantically*, resp.) *Lit-dependent*.

Proposition 1. *If $\langle I, \beta\rangle$ is an interpretation, F is a formula and S is a consistent set of ground literals such that $\langle I, \beta\rangle \models F$ and $S \cap \mathcal{L}(F) = \emptyset$, then $\langle I[\widetilde{S}], \beta\rangle \models F$.*

The analog to Prop. 1 for the *essential* literal base can be shown for first-order formulas which do not contain existential quantifiers. For such formulas F, interpretations $\langle I, \beta\rangle$, and consistent sets of ground literals S, it can be proven that if $\langle I, \beta\rangle \models F$ and $\langle I[S], \beta\rangle \not\models F$, then there exists a *finite* set $S' \subseteq S$ such that $\langle I[S'], \beta\rangle \not\models F$. From this property, the analog to Prop. 1 for the essential literal base can be derived. Since this analog is useful to prove properties of projection, we give formulas that satisfy it a name, $\mathcal{E}$-formulas:

Definition 3 ($\mathcal{E}$-Formula). A formula F is called *$\mathcal{E}$-formula* if and only if for all interpretations $\langle I, \beta\rangle$ and consistent sets of ground literals S such that $\langle I, \beta\rangle \models F$ and $S \cap \mathcal{L}_{\mathcal{E}}(F) = \emptyset$ it holds that $\langle I[\widetilde{S}], \beta\rangle \models F$.

As indicated above, first-order formulas without existential quantifier – including propositional formulas and first-order clausal formulas – are $\mathcal{E}$-formulas. Being an $\mathcal{E}$-formula is a property that just depends on the semantics of a formula, that is, an equivalent to an $\mathcal{E}$-formula is also an $\mathcal{E}$-formula.

4 Properties of Projection

Basic Properties. Based on the semantic definition of project, it is not hard to prove properties of projection as displayed in Tab. 4 and 5. They hold for all formulas F, F_1, F_2, $\mathcal{E}$-formulas E, literals L, and literal scopes S, S_1, S_2. The properties in Tab. 5 strengthen properties in Tab. 4, but apply only to $\mathcal{E}$-formulas.

Projection is "semantically determined", in the sense that projections of equivalent formulas onto the same scope are equivalent (Tab. 4.iii). The projection of a formula onto its literal base is equivalent to the original formula (Tab. 4.vii). The essential literal base of a projection is a subset of the projection scope and also a subset of the essential literal base of the projection formula (Tab. 4.xii,xiii). Only the intersection of the given scope with the literal base of the argument formula is relevant for projection and forgetting (Tab. 4.xv, xvi). Property Tab. 4.xvii is behind many applications of predicate quantifier elimination and its variants: A formula *Query* is entailed by a formula *KB* if and only if it is entailed by the projection of *KB* onto the literal base of *Query*. Properties Tab. 4.xviii–xxiii show relationships of projection with other logic operators.

Conjunction and Linklessness. While projection distributes straightforwardly over disjunction and existential first-order quantification (Tab. 4.xx, xxii), only one direction of the corresponding equivalences holds for conjunction and universal first-order quantification (Tab. 4.xxi, xxiii). The precondition of the following theorem is sufficient for the converse of property Tab. 4.xxi. The essential part of its proof is deferred to Lemma 1 below. The theorem is based

Table 4. Properties of Projection

(i)	$F \models \mathsf{project}(F, S)$
(ii)	*if* $F_1 \models F_2$, *then* $\mathsf{project}(F_1, S) \models \mathsf{project}(F_2, S)$
(iii)	*if* $F_1 \equiv F_2$, *then* $\mathsf{project}(F_1, S) \equiv \mathsf{project}(F_2, S)$
(iv)	*if* $S_1 \supseteq S_2$, *then* $\mathsf{project}(F, S_1) \models \mathsf{project}(F, S_2)$
(v)	$\mathsf{project}(\mathsf{project}(F, S_1), S_2) \equiv \mathsf{project}(F, S_1 \cap S_2)$
(vi)	$F_1 \models \mathsf{project}(F_2, S)$ *if and only if* $\mathsf{project}(F_1, S) \models \mathsf{project}(F_2, S)$
(vii)	$\mathsf{project}(F, \mathcal{L}(F)) \equiv F$
(viii)	$\mathsf{project}(F, \mathsf{GLITS}) \equiv F$
(ix)	$\mathsf{project}(\top, S) \equiv \top$
(x)	$\mathsf{project}(\bot, S) \equiv \bot$
(xi)	F *is satisfiable if and only if* $\mathsf{project}(F, S)$ *is satisfiable*
(xii)	$\mathcal{L}_\mathcal{E}(\mathsf{project}(F, S)) \subseteq S$
(xiii)	$\mathcal{L}_\mathcal{E}(\mathsf{project}(F, S)) \subseteq \mathcal{L}_\mathcal{E}(F)$
(xiv)	*if* $\mathsf{project}(F, S) \models F$, *then* $\mathcal{L}_\mathcal{E}(F) \subseteq S$
(xv)	$\mathsf{project}(F, S) \equiv \mathsf{project}(F, \mathcal{L}(F) \cap S)$
(xvi)	$\mathsf{forget}(F, S) \equiv \mathsf{forget}(F, \mathcal{L}(F) \cap S)$
(xvii)	$F_1 \models F_2$ *if and only if* $\mathsf{project}(F_1, \mathcal{L}(F_2)) \models F_2$
(xviii)	*if no instance of* L *is in* S, *then* $\mathsf{project}(L, S) \equiv \top$
(xix)	*if all instances of* L *are in* S, *then* $\mathsf{project}(L, S) \equiv L$
(xx)	$\mathsf{project}(F_1 \vee F_2, S) \equiv \mathsf{project}(F_1, S) \vee \mathsf{project}(F_2, S)$
(xxi)	$\mathsf{project}(F_1 \wedge F_2, S) \models \mathsf{project}(F_1, S) \wedge \mathsf{project}(F_2, S)$
(xxii)	$\mathsf{project}(\exists x F, S) \equiv \exists x\, \mathsf{project}(F, S)$
(xxiii)	$\mathsf{project}(\forall x F, S) \models \forall x\, \mathsf{project}(F, S)$

Table 5. Properties of Projection for $\mathcal{E}$-Formulas

(i)	$\mathsf{project}(E, \mathcal{L}_\mathcal{E}(E)) \equiv E$	(strengthens Tab. 4.vii)
(ii)	$\mathcal{L}_\mathcal{E}(E) \subseteq S$ *if and only if* $\mathsf{project}(E, S) \equiv E$	(strengthens Tab. 4.xiv)
(iii)	$\mathsf{project}(E, S) \equiv \mathsf{project}(E, \mathcal{L}_\mathcal{E}(E) \cap S)$	(strengthens Tab. 4.xv)
(iv)	$\mathsf{forget}(E, S) \equiv \mathsf{forget}(E, \mathcal{L}_\mathcal{E}(E) \cap S)$	(strengthens Tab. 4.xvi)
(v)	$F \models E$ *if and only if* $\mathsf{project}(F, \mathcal{L}_\mathcal{E}(E)) \models E$	(strengthens Tab. 4.xvii)

on the relation *linkless outside*, which applies to a pair of formulas and a literal scope if all ground atoms "involved in links" between the formulas (that is, are the atoms of complementary ground instances of two literals, one in each component of the pair) are contained in the literal scope, positively as well as negatively. For example, the pair of formulas $\langle p \vee q,\ \neg p \vee q \rangle$ is linkless outside the literal scope $\{+p, -p\}$. The relation is symmetric with respect to the pair components. Its formal definition is:

Definition 4 (Linkless Pair of Formulas). A pair of formulas $\langle F_1, F_2 \rangle$ is called *linkless outside* a literal scope S if and only if $\mathcal{L}(F_1) \cap \widetilde{\mathcal{L}(F_2)} \subseteq S \cap \widetilde{S}$. In the case where $S = \emptyset$, we also just say that $\langle F_1, F_2 \rangle$ is *linkless*.

Theorem 1 (Projection and Linkless Conjunction). *If* F_1, F_2 *are formulas and* S *is a literal scope such that* $\langle F_1, F_2 \rangle$ *is linkless outside* S, *then* $\mathsf{project}(F_1 \wedge F_2, S) \equiv \mathsf{project}(F_1, S) \wedge \mathsf{project}(F_2, S)$.

Proof. The case where $F_1 = F_2$ is trivial. Assume $F_1 \neq F_2$. Left-to-right is stated as Tab. 4.xxi. Right to left follows from Lemma 1 below, with $\Phi = \{F_1, F_2\}$, along with the semantic definitions of projection and conjunction (Tab. 1). □

Applications of Theorem 1. Boolean quantifier elimination is generalized by propositional projection computation, that is, computing for a propositional formula with the projection operator an equivalent formula without the projection operator. Theorem 1 can be applied to justify methods for this task. They take as input a formula $\mathsf{project}(F, S)$, where F is a propositional formula in negation normal form, and proceed as follows:

1. Compute a formula F' which is equivalent to F and has the property that for all conjunctive subformulas $(F_1 \wedge F_2)$ it holds that $\langle F_1, F_2 \rangle$ is linkless outside S.
2. Replace all literals in F' that are not in S by $\top$. The obtained formula is the result of projection computation.

Propositional formulas that satisfy the condition of step (1.) for the empty set as S (and thus also for any other set of literals as S) are called *linkless* [10]. Subclasses of linkless formulas are DNNF [5] and DNF, if complementary literals are not permitted in the same clause. Step (2.) is justified, since by property 4.xx and Theorem 1 the $\mathsf{project}$ operator in $\mathsf{project}(F', S)$ can be distributed inwards, immediately in front of literal subformulas, where its value is determined by Tab. 4.xviii and xix. Regular tableaux, including semantic trees, whose nodes are labeled with literals, either propositional, or with just non-rigid variables, can be considered as representations of formulas – which are linkless. This is utilized by propositional tableau- and DPLL-based knowledge compilation methods that also perform Boolean quantifier elimination [10, 16, 12].

Another application of Theorem 1 is the justification of methods for propositional *Lit-simplifying* [11]. That is, computing for a given propositional formula an equivalent formula containing only literals in the essential literal base, which is assumed to be known. By Tab. 4.i, this task can be computed as described above for projection computation: transforming to an equivalent formula where all pairs of conjuncts are linkless outside the essential literal base, followed by replacing the literals not in the essential base by $\top$. There is also a dual alternative: From Tab. 4.i follows $\mathsf{project}(F, \mathcal{L}_{\mathcal{E}}(F)) \equiv \neg\mathsf{project}(\neg F, \mathcal{L}_{\mathcal{E}}(\neg F))$. Thus, *Lit-simplifying* can also be performed by operating on $\neg F$ instead of F, or, considered dually, by computing a formula that is equivalent to F and has the property that for all *disjunctive* subformulas $(F_1 \vee F_2)$ it holds that $\langle F_1, F_2 \rangle$ is linkless outside the essential literal base of F (CNF formulas without tautological clauses, for example, have this property). The literals not in the essential base are then replaced by $\bot$.[4]

[4] In [11, Sect. 3.2] it is erroneously stated that Lit-simplifying of a propositional formula in *negation normal form (NNF)* can be performed by (in our terminology) substituting the literals which are not in the essential literal base with $\bot$, and thus

Conjunction and Essential Linklessness. Theorem 2, which follows, strengthens Theorem 1 for $\mathcal{E}$-formulas. In its precondition the property *essentially linkless outside* (Def. 5) takes the place of the stronger *linkless outside*. That *essentially linkless outside* is weaker follows from the fact that the essential literal base of a formula is a subset of its literal base. *Essentially linkless outside* is in a further respect different from *linkless outside*: it is independent of syntactic properties – if $\langle F_1, F_2\rangle$ is essentially linkless outside S, and F_1', F_2' are formulas such that $F_1' \equiv F_1$ and $F_2' \equiv F_2$, then also $\langle F_1', F_2'\rangle$ is essentially linkless outside S. This follows, since equivalent formulas have the same essential literal base.

Definition 5 (Essentially Linkless Pair of Formulas). A pair of formulas $\langle F_1, F_2\rangle$ is called *essentially linkless outside* a literal scope S if and only if $\mathcal{L}_{\mathcal{E}}(F_1) \cap \widetilde{\mathcal{L}_{\mathcal{E}}(F_2)} \subseteq S \cap \widetilde{S}$.

Example 4 (Essentially Linkless Pair of Formulas). Let S be the set of literals $\{+p, -p\}$. Then $\langle p \vee (q \wedge \neg q),\ \neg p \vee q\rangle$ is not linkless outside S, but essentially linkless outside S.

Theorem 2 (Projection and Essentially Linkless Conjunction). *If F_1, F_2 are $\mathcal{E}$-formulas and S is a literal scope such that $\langle F_1, F_2\rangle$ is essentially linkless outside S, then* $\mathsf{project}(F_1 \wedge F_2, S) \equiv \mathsf{project}(F_1, S) \wedge \mathsf{project}(F_2, S)$.

Proof. Can be shown in the same way as Theorem 1, but based on a variant of Lemma 1, where Φ is a set of $\mathcal{E}$-formulas, and *linkless outside* in (A1) is replaced by *essentially linkless outside*. The varied lemma can be proven like the original one, with $\mathcal{L}_{\mathcal{E}}$ in place of $\mathcal{L}$ and referring to Def. 3 instead of Prop. 1. □

Universal Quantification and Linklessness. The same principles that permit to push the projection operator inside conjunctions can be applied to universal quantification. We state it for *linkless outside* as precondition in the following theorem.

Theorem 3 (Projection and Linkless Universal Quantification). *If F is a formula, x is a variable that occurs free or not at all in F, and S is a literal scope such that for all $t, u \in \mathsf{GTERMS}$ where $t \neq u$ it holds that $\langle F\{x \mapsto t\}, F\{x \mapsto u\}\rangle$ is linkless outside S, then* $\mathsf{project}(\forall x F, S) \equiv \forall x\ \mathsf{project}(F, S)$.

Proof. The case where x does not occur in F is trivial. Assume x occurs free in F. Left-to-right is stated as Tab. 4.xxiii. Right-to-left follows from Lemma 1 below, with $\Phi = \{F\{x \mapsto t\} \mid t \in \mathsf{GTERMS}\}$, along with the semantic definitions of projection and universal quantification (Tab. 1). □

---

would be a polynomial operation. The statement is false: Let $F \stackrel{\text{def}}{=} (p \vee \neg p)$. Then F is in NNF and $\mathcal{L}_{\mathcal{E}}(F) = \emptyset$. Substituting in F the literals not in $\mathcal{L}_{\mathcal{E}}(F)$ with $\bot$ yields $(\bot \vee \bot)$, which is not equivalent to F.

The Lemma Underlying Theorems 1–3. We conclude this section by stating the lemma underlying Theorems 1–3 and giving a proof sketch for it.

Lemma 1. *If $\langle I, \beta\rangle$ is an interpretation, S is a literal scope, and Φ a set of formulas such that*

(A1) for all formulas $F, G \in \Phi$ such that $F \neq G$ it holds that $\langle F\beta, G\beta\rangle$ is linkless outside S, and
(A2) for all formulas $F \in \Phi$ it holds that $\langle I, \beta\rangle \models \mathsf{project}(F, S)$,

then there exists an interpretation $\langle J, \beta\rangle$ such that

(C1) for all formulas $F \in \Phi$ it holds that $\langle J, \beta\rangle \models F$, and
(C2) $J \cap S \subseteq I$.

Proof (Sketch – see [12, Theorems 2–4] for detailed proofs of similar properties). Let $\langle I, \beta\rangle, S$, and Φ be as specified for the lemma, and assume that they satisfy preconditions (A1) and (A2). For all $F \in \Phi$ let J_F be a structure such that $\langle J_F, \beta\rangle \models F$ and $J_F \cap S \subseteq I$. The existence of such J_F follows from (A2) and the semantic definition of $\mathsf{project}$. We prove the lemma by showing the construction of a structure J such that consequences (C1) and (C2) are satisfied. The construction of J is based on two auxiliary sets of literals, $\mathcal{L}^{+A}$ and $\mathcal{L}^{-A}$, which are associated with each ground atom A:

$$\mathcal{L}^{+A} \stackrel{\text{def}}{=} \bigcup_{+A \,\in\, J_F,\ F \in \Phi} \mathcal{L}(F\beta), \qquad \mathcal{L}^{-A} \stackrel{\text{def}}{=} \bigcup_{-A \,\in\, J_F,\ F \in \Phi} \mathcal{L}(F\beta).$$

The structure J is then defined by:

$$\text{If } +A \notin \mathcal{L}^{+A} \text{ and } (-A \in \mathcal{L}^{-A} \text{ or } -A \in I), \text{ then } -A \stackrel{\text{def}}{\in} J, \text{ else } +A \stackrel{\text{def}}{\in} J.$$

For all elements F of Φ let $M_F \stackrel{\text{def}}{=} J_F \cap \widetilde{J}$. Then $J = J_F[\widetilde{M_F}]$. It can be verified that $M_F \cap \mathcal{L}(F\beta) = \emptyset$. Since $\langle J_F, \beta\rangle \models F$, consequence (C1) then follows from Prop. 1. It can be verified that $J \subseteq \bigcup_{F \in \Phi} J_F \cup I$. Consequence (C2) then follows since $J_F \cap S \subseteq I$. □

5 Unlinked Literal Occurrences

The following theorem uses the *linkless outside* property related to specific occurrences of literals in formulas. If a literal is "not linked" to its context within a formula, then replacing the literal by $\top$ yields a formula whose projection is equivalent to the projection of the original one. The idea is to use the equivalence stated in the theorem as building block for methods to eliminate the projection operator, although this remains largely future work. As a first approach, we show below, with Prop. 2, that a restricted variant of the *Ackermann lemma* [17, 18], the basis of several known methods for second-order quantifier elimination [3, 9, 8], can be modeled with the theorem.

Following the terminology of [19], if F is a formula with a subformula occurrence replaced by a *hole*, and G is a formula, then $F[G]$ is F with the hole replaced by G. At the same time $F[G]$ indicates that the formula F contains an occurrence of the subformula G.

Theorem 4 (Eliminating an Unlinked Literal Occurrence). *If S is a literal scope, L is a literal of which no instance is in S, and $F[L]$ is a first-order formula such that*

(A1) *for all subformulas $(F_1[L] \wedge F_2)$ and $(F_2 \wedge F_1[L])$ of $F[L]$ it holds that $\langle L,\ F_2\rangle$ is linkless,*[5] *and*

(A2) *for all subformulas $(\forall x F'[L])$ of $F[L]$ and $t_1, t_2 \in \mathsf{GTERMS}$ such that $t_1 \neq t_2$ it holds that $\langle L\{x \mapsto t_1\},\ F'[L]\{x \mapsto t_2\}\rangle$ is linkless,*

then

$$\mathsf{project}(F[L], S) \equiv \mathsf{project}(F[\top], S).$$

Proof (Sketch). The left-to-right direction follows from Tab. 4.ii, since $F[L] \models F[\top]$. The right-to-left direction can be shown as follows: Let $\langle I, \beta\rangle$ be an interpretation such that $\langle I, \beta\rangle \models \mathsf{project}(F[\top], S)$. By the definition of $\mathsf{project}$, there exists an interpretation $\langle J, \beta\rangle$ such that $\langle J, \beta\rangle \models F[\top]$ and $J \cap S \subseteq I$. We need to show that $\langle I, \beta\rangle \models \mathsf{project}(F[L], S)$, that is, that there exists an interpretation $\langle K, \beta\rangle$ such that $\langle K, \beta\rangle \models F[L]$ and $K \cap S \subseteq I$. In case $\langle J, \beta\rangle \models F[L]$, we have found in J a suitable K. The other case, where $\langle J, \beta\rangle \not\models F[L]$, can be shown by induction on first-order formulas $F[L]$, where L is a literal, and $F[L]$ satisfies (A1) and (A2). Before stating the induction property, we need some more notation: If F, G are formulas and β is a variable assignment, then $F(\beta \downarrow G)$ denotes F with those variables that are free in G replaced by their images in β. The induction property is: *If S is a literal scope such that no instance of L is in S and $\langle J, \beta\rangle$ is an interpretation such that $\langle J, \beta\rangle \models F[\top]$ and $\langle J, \beta\rangle \not\models F[L]$, then there exists a set M of ground instances of $L(\beta \downarrow F[L])$ such that $\langle J[M], \beta\rangle \models F[L]$.* The set of literals M is defined such that $S \cap M = \emptyset$, which allows to conclude $J[M] \cap S \subseteq J \cap S$. If $K = J[M]$, then from $J \cap S \subseteq I$ follows $K \cap S \subseteq I$.

We sketch the base case for literals, and the induction steps for conjunction and universal quantification. The remaining induction steps for disjunction and existential quantification are straightforward to show. In the base case where $F[L] = L$ it holds that $\langle J, \beta\rangle \not\models L$. A suitable M is $\{L\beta\}$. The induction step for $F[L] = (F_1[L] \wedge F_2)$ can be shown as follows: From $\langle J, \beta\rangle \models (F_1[\top] \wedge F_2)$ and $\langle J, \beta\rangle \not\models (F_1[L] \wedge F_2)$ follows $\langle J, \beta\rangle \not\models F_1[L]$, hence by the induction assumption, there exists a set M of instances of $L(\beta \downarrow F_1[L])$ such that $\langle J[M], \beta\rangle \models F_1[L]$. By condition (A1) it holds that $\mathcal{L}(F_2) \cap \widetilde{M} = \emptyset$. From $\langle J, \beta\rangle \models F_2$, then by Prop. 1 follows $\langle J[M], \beta\rangle \models F_2$.

The induction step for $F[L] = (\forall x F')$ can be shown as follows: From $\langle J, \beta\rangle \models \forall x\ F'[\top]$ follows that for all ground terms t it holds that $\langle J, \beta\frac{t}{x}\rangle \models F'[\top]$. Let T be the set of ground terms t such that $\langle J, \beta\frac{t}{x}\rangle \not\models F'[L]$. From $\langle J, \beta\rangle \not\models \forall x\ F'[L]$ follows that T is not empty. For all $t \in T$ let M_t be a set of literals, ground instances of $L(\beta\frac{t}{x} \downarrow F'[L])$ such that $\langle J[M_t], \beta\frac{t}{x}\rangle \models F'[L]$. The existence of these M_t follows from the induction assumption. Let $M \stackrel{\text{def}}{=} \bigcup_{t \in T} M_t$. The induction conclusion $\langle J[M], \beta\rangle \models \forall x F'[L]$ is implied by the fact that for all

[5] By the precondition that no instance of L is in S, the *linkless* condition here and in precondition (A2) is equivalent to *linkless outside* S.

ground terms t it holds that $\langle J[M], \beta\frac{t}{x}\rangle \models F'[L]$, which can be shown as follows: Let t be an element of $\widetilde{T}$ and s be an arbitrary ground term, different from t. By (A2) it holds that $\mathcal{L}(\widetilde{L\{x \mapsto t\}}) \cap \mathcal{L}(F'[L]\{x \mapsto s\}) = \emptyset$. Since $M_t \subseteq \mathcal{L}(F'[L]\{x \mapsto t\})$, it follows that

$$\widetilde{M_t} \cap \mathcal{L}(F'[L]\{x \mapsto s\}) = \emptyset. \tag{i}$$

If $t \in T$, then $\langle J[M_t], \beta\frac{t}{x}\rangle \models F'[L]$, hence $\langle J[M_t], \beta\frac{t}{x}\rangle \models F'[L]\{x \mapsto t\}$. From Eq. (i) follows $(M - M_t) \cap \mathcal{L}(F'[L]\{x \mapsto t\}) = \emptyset$. By Prop. 1 follows $\langle J[M], \beta\frac{t}{x}\rangle \models F'[L]\{x \mapsto t\}$, hence $\langle J[M], \beta\frac{t}{x}\rangle \models F'[L]$. Else, if $t \notin T$, from Eq. (i) follows $M \cap \mathcal{L}(F'[L]\{x \mapsto t\}) = \emptyset$. From $\langle J, \beta\frac{t}{x}\rangle \models F'[L]$ it can be concluded, similarly as in the case where $t \in T$, with Prop. 1 that $\langle J[M], \beta\frac{t}{x}\rangle \models F'[L]$. □

Application of Theorem 4. The Ackermann lemma states an equivalence of formulas of a certain form and with a second-order quantifier to first-order formulas. Proposition 2 can be used to prove a restricted variant of the Ackermann lemma (the restriction is that the subformula which may contain multiple instantiated occurrences of the quantified predicate is not permitted to contain the existential first-order quantifier). The proposition statement shows that equivalence with respect to the projection (expressed conveniently as forgetting) is preserved by eliminating a single occurrence of a positive literal with predicate p. The way the literal occurrence is eliminated is identical to the way in which according to the Ackermann lemma all occurrences of literals with positive p are eliminated. By iterated rewriting with the equivalence of Prop. 2, all positive occurrences of p can be eliminated, permitting the negative occurrence finally to be eliminated directly according to Theorem 4, followed by dropping the forget operator. The result is then the same first-order formula that would be obtained by a single rewriting step with the Ackermann lemma. Of course, as for the Ackermann lemma, there is also a dual variant of Prop. 2 with polarities of the occurrences of p switched, but we do not state this explicitly here.

We use additional notation: A sequence of terms $t_1, ..., t_n$ is abbreviated by $\overline{t}$. If F is a formula, then $F(\overline{t})$ denotes F with all occurrences of variables x_i replaced by term t_i, for $i \in \{1, ..., n\}$.

Proposition 2 (Ackermann Lemma Step for Universal Formulas). *Let p be a n-ary predicate symbol, and F, G be first-order formulas such that p does not occur in F and does occur in G only in positive literals, G has the form $G[p(\overline{t})]$, and G does not contain an existential quantifier. Then*

$$\begin{array}{lll} \textit{(F1)} & \mathsf{forget}((\forall \overline{x}\ \neg p(\overline{x}) \vee F(\overline{x})) \wedge G[p(\overline{t})],\ P_+) & \equiv \\ \textit{(F2)} & \mathsf{forget}((\forall \overline{x}\ \neg p(\overline{x}) \vee F(\overline{x})) \wedge G[F(\overline{t})],\ P_+), & \end{array}$$

where P_+ is the set of all positive ground literals with predicate p.

Proof. We first show the proof in broad lines, filling in details subsequently: Assume $\approx$ is a "built-in" predicate symbol representing equality, that is, interpretations satisfy the usual equality axioms with respect to $\approx$. Let $\overline{x} \not\approx \overline{t}$

abbreviate the formula $(\neg(x_1 \approx t_1) \vee ... \vee \neg(x_n \approx t_n))$. Formula (F1) is equivalent to the following formula (F1′), since the formula arguments in both forget expressions are equivalent.

$$\text{(F1}') \quad \mathsf{forget}(\forall \overline{x}\ (p(\overline{x}) \wedge F(\overline{x}) \wedge G[\top]) \vee (\neg p(\overline{x}) \wedge G[\overline{x} \not\approx \overline{t}]), P_+)$$

The leftmost literal $p(\overline{x})$ in (F1′) meets the requirements on L in Theorem 4. By that theorem (F1′) is equivalent to (F2′), which, again by equivalence of the formula arguments in both forget expressions, is equivalent to (F2).

$$\text{(F2}') \quad \mathsf{forget}(\forall \overline{x}\ (\top \wedge F(\overline{x}) \wedge G[\top]) \vee (\neg p(\overline{x}) \wedge G[\overline{x} \not\approx \overline{t}]), P_+)$$

We now fill in the details of the proof by deriving the equivalences of the formula arguments in the forget expressions. The following auxiliary equivalence holds for all first-order formulas $G[H(\overline{t})]$ that do not contain a variable from $\overline{x}$:

$$\text{(Aux)} \quad G[H(\overline{t})] \equiv (\forall \overline{x}\ H(\overline{x}) \vee G[\overline{x} \not\approx \overline{t}]) \wedge G[\top].$$

A presupposition of equivalence (Aux) is that the subformula H is within G not in the scope of an odd number of negation operators – which is trivially ensured by our definition of *first-order formula* as constructed from literals. Equivalence (Aux) can then be derived from well-known formula equivalences: $G[H(\overline{t})] \equiv G[\forall \overline{x}\ \overline{x} \not\approx \overline{t} \vee H(\overline{x})] \equiv (\forall \overline{x}\ G[\overline{x} \not\approx \overline{t} \vee H(\overline{x})]) \equiv (\forall \overline{x}\ (H(\overline{x}) \vee G[\overline{x} \not\approx \overline{t} \vee \bot]) \wedge G[\overline{x} \not\approx \overline{t} \vee \top]) \equiv ((\forall \overline{x}\ H(\overline{x}) \vee G[\overline{x} \not\approx \overline{t}]) \wedge G[\top])$.

By expanding $G[p(\overline{t})]$ according to (Aux), the formula argument of (F1) is equivalent to $(\forall \overline{x}\ \ (\neg p(\overline{x}) \vee F(\overline{x})) \wedge (p(\overline{x}) \vee G[\overline{x}\ \not\approx\ \overline{t}]) \wedge G[\top])$. The formula argument of (F1′) can be obtained from this formula, which has the form $\forall \overline{x}\ F'$, by expanding according to the equivalence $\forall \overline{x} F' \equiv (\forall \overline{x}(p(\overline{x}) \wedge F') \vee (\neg p(\overline{x}) \wedge F'))$, followed by simplifying with well-known equivalences and the equivalence $(G[\overline{x} \not\approx \overline{t}] \wedge G[\top]) \equiv G[\overline{x} \not\approx \overline{t}]$. Equivalence of the formula argument of (F2) to that of (F2′) can be shown similarly: By expanding $G[F(\overline{t})]$ according to (Aux), the formula argument of (F2) is equivalent to $(\forall \overline{x}\ (\neg p(\overline{x}) \vee F(\overline{x})) \wedge (F(\overline{x}) \vee G[\overline{x} \not\approx \overline{t}]) \wedge G[\top])$. The formula argument of (F2′) can be obtained from this formula by simplifying with well-known equivalences and $(G[\overline{x} \not\approx \overline{t}] \wedge G[\top]) \equiv G[\overline{x} \not\approx \overline{t}]$, followed by inserting the truth value constant $\top$ into the leftmost conjunction. □

6 Conclusion

We expect that the consideration of polarity will play an important role in some applications of predicate quantifier elimination – for example to compute knowledge base extracts that just keep information about a predicate in one polarity and thus suffice to answer queries containing the predicate just in this polarity, or for knowledge base modularization, where it should be ensured that additions to a knowledge base affect some predicate only in a certain polarity. We presented a formalization that expresses polarity sensitive predicate quantification in an easily accessible way by means of literal sets. It applies to first-order logic, and thus should provide a basis also for other logics used in knowledge representation that are more expressive than just propositional logic. We applied the formalization to show some properties of literal projection which relate to methods

for predicate quantifier elimination. These properties already suffice as building blocks to justify some methods that can in practice be used for predicate quantifier elimination, and provide a basis to extend the successful applications of projection computation in the context of propositional knowledge compilation to more expressive logics. The investigation of further methods, especially for non-propositional formulas, remains future work.

Acknowledgments. I am grateful to Renate A. Schmidt for valuable comments and discussions on earlier versions of some of the material in the paper, and to anonymous referees for helpful suggestions to improve the presentation.

References

1. Gabbay, D., Ohlbach, H.J.: Quantifier elimination in second-order predicate logic. In: KR'92. (1992) 425–435
2. Lin, F., Reiter, R.: Forget It! In: Working Notes, AAAI Fall Symposium on Relevance. (1994) 154–159
3. Doherty, P., Łukaszewicz, W., Szałas, A.: Computing circumscription revisited: A reduction algorithm. J. Autom. Reason. **18**(3) (1997) 297–338
4. McMillan, K.L.: Applying SAT methods in unbounded symbolic model checking. In: CAV 2002. (2002) 250–264
5. Darwiche, A.: Decomposable negation normal form. JACM **48**(4) (2001) 608–647
6. Wernhard, C.: Semantic knowledge partitioning. In: JELIA 04. (2004) 552–564
7. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In: KR'06. (2006) 187–197
8. Gabbay, D.M., Schmidt, R.A., Szałas, A.: Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications. CollegePublications (2008)
9. Conradie, W., Goranko, V., Vakarelov, D.: Algorithmic correspondence and completeness in modal logic. I. the core algorithm SQEMA. Log. Meth. in Comp. Science **2**(1:5) (2006) 1–26
10. Murray, N.V., Rosenthal, E.: Tableaux, path dissolution and decomposable negation normal form for knowledge compilation. In: TABLEAUX 2003. (2003) 165–180
11. Lang, J., Liberatore, P., Marquis, P.: Propositional independence — formula-variable independence and forgetting. JAIR **18** (2003) 391–443
12. Wernhard, C.: Automated Deduction for Projection Elimination. PhD thesis, Universität Koblenz-Landau, Germany (2008). To appear.
13. Urban, J.: MPTP 0.2: Design, implementation, and initial experiments. J. Autom. Reason. **37**(1-2) (2006) 21–43
14. Wernhard, C.: Literal projection for first-order logic. In: JELIA 08. (2008) 389–402
15. Ebbinghaus, H.D., Flum, J., Thomas, W.: Einführung in die mathematische Logik. 4th edn. Spektrum Akademischer Verlag, Heidelberg (1996)
16. Huang, J., Darwiche, A.: DPLL with a trace: From SAT to knowledge compilation. In: IJCAI-05. (2005) 156–162
17. Ackermann, W.: Untersuchungen über das Eliminationsproblem der mathematischen Logik. Math. Annalen **110** (1935) 390–413
18. Szalas, A.: On the correspondence between modal and classical logic: An automated approach. J. Log. Comput. **3**(6) (1993) 605–620
19. Dershowitz, N., Plaisted, D.A.: Rewriting. In: Handbook of Automated Reasoning. Volume I. Elsevier Science (2001) 537–610