The Boolean Solution Problem from the Perspective of Predicate Logic

Christoph Wernhard(✉)

Technische Universität Dresden, Dresden, Germany
info@christophwernhard.com

Abstract. Finding solution values for unknowns in Boolean equations was a principal reasoning mode in the *Algebra of Logic* of the 19th century. Schröder investigated it as *Auflösungsproblem* (*solution problem*). It is closely related to the modern notion of Boolean unification. Today it is commonly presented in an algebraic setting, but seems potentially useful also in knowledge representation based on predicate logic. We show that it can be modeled on the basis of first-order logic extended by second-order quantification. A wealth of classical results transfers, foundations for algorithms unfold, and connections with second-order quantifier elimination and Craig interpolation show up.

1 Introduction

Finding solution values for unknowns in Boolean equations was a principal reasoning mode in the *Algebra of Logic* of the 19th century. Schröder [27] investigated it as *Auflösungsproblem* (*solution problem*). It is closely related to the modern notion of Boolean unification. For a given formula that contains unknowns formulas are sought such that after substituting the unknowns with them the given formula becomes valid or, dually, unsatisfiable. Of interest are also most general solutions, condensed representations of all solution substitutions. A central technique there is the *method of successive eliminations*, which traces back to Boole. Schröder investigated *reproductive solutions* as most general solutions, anticipating the concept of *most general unifier*. A comprehensive modern formalization based on this material, along with historic remarks, is presented by Rudeanu [23] in the framework of Boolean algebra. In automated reasoning variants of these techniques have been considered mainly in the late 80s and early 90s with the motivation to enrich Prolog and constraint processing by Boolean unification with respect to propositional formulas handled as terms [8,15,16,20–22]. An early implementation based on [23] has been also described in [29]. An implementation with BDDs of the algorithm from [8] is reported in [9]. The Π_2^P-completeness of Boolean unification with constants was proven only later in [15,16] and seemingly independently in [2]. Schröder's results were developed further by Löwenheim [18,19]. A generalization of Boole's method beyond propositional logic to relational monadic formulas has been presented by Behmann in the early 1950s [5,6]. Recently the complexity of Boolean unification in a predicate logic setting has been investigated for some formula classes, in particular for

C. Dixon and M. Finger (Eds.): FroCoS 2017, LNCS 10483, pp. 333–350, 2017.
DOI: 10.1007/978-3-319-66167-4_19

quantifier-free first-order formulas [12]. A brief discussion of Boolean reasoning in comparison with predicate logic can be found in [7].

Here we remodel the solution problem formally along with basic classical results and some new generalizations in the framework of first-order logic extended by second-order quantification. The main thesis of this work is that it is possible and useful to apply second-order quantification consequently throughout the formalization. What otherwise would require meta-level notation is then expressed just with formulas. As will be shown, classical results can be reproduced in this framework in a way such that applicability beyond propositional logic, possible algorithmic variations, as well as connections with second-order quantifier elimination and Craig interpolation become visible. As demonstrated in [30], the foundations developed here are adequate as basis for adaptions of further classical material, notably reproductive solutions, and for further studies such as the investigation of certain special cases for which constructive solution methods are available and a generalization of the solution problem where vocabulary restrictions are taken into account.

The envisaged application scenario is to let solving "solution problems", or Boolean equation solving, on the basis of predicate logic join reasoning modes like second-order quantifier elimination (or "semantic forgetting"), Craig interpolation and abduction to support the mechanized reasoning about relationships between theories and the extraction or synthesis of subtheories with given properties. On the practical side, the aim is to relate it to reasoning techniques such as Craig interpolation on the basis of first-order provers, SAT and QBF solving, and second-order quantifier elimination based on resolution [14] and the Ackermann approach [11]. Numerous applications of Boolean equation solving in various fields are summarized in [24, Chap. 14]. Applications in automated theorem proving and proof compression are mentioned in [12, Sect. 7]. The prevention of certain redundancies has been described as application of (concept) unification in description logics [4]. In [30] the synthesis of definitional equivalences is sketched as an application.

The rest of the paper is structured as follows: Notation, in particular for substitution in formulas, is introduced in Sect. 2. In Sect. 3 a formalization of the solution problem is presented and related to different points of view. Section 4 is concerned with abstract properties of and algorithmic approaches to solution problems with several unknowns. Conditions under which solutions exist are discussed in Sect. 5. Section 6 closes the paper with concluding remarks.

2 Notation and Preliminaries

2.1 Notational Conventions

We consider formulas in first-order logic extended by second-order quantification upon predicates. They are constructed from atoms, constant operators $\top$, $\bot$, the unary operator $\neg$, binary operators $\wedge, \vee$ and quantifiers $\forall, \exists$ with their usual meaning. Further binary operators $\rightarrow, \leftarrow, \leftrightarrow$, as well as n-ary versions of $\wedge$ and $\vee$ can be understood as meta-level notation. The operators $\wedge$ and $\vee$ bind stronger

than $\rightarrow$, $\leftarrow$ and $\leftrightarrow$. The scope of $\neg$, the quantifiers, and the n-ary connectives is the immediate subformula to the right. A subformula occurrence has in a given formula *positive (negative) polarity* if it is in the scope of an even (odd) number of negations.

A *vocabulary* is a set of *symbols*, that is, predicate symbols (briefly *predicates*), function symbols (briefly *functions*) and *individual symbols*. (Individual symbols are not partitioned into variables and constants. Thus, an individual symbol is – like a predicate – considered as variable if and only if it is bound by a quantifier.) The arity of a predicate or function s is denoted by $\mathsf{arity}(s)$. The set of symbols that occur *free* in a formula F is denoted by $\mathsf{free}(F)$. Symbols not present in the formulas and other items under discussion are called *fresh*. We write $F \models G$ for F *entails* G; $\models F$ for F *is valid*; and $F \equiv G$ for F *is equivalent to* G, that is, $F \models G$ *and* $G \models F$.

We write *sequences* of symbols, of terms and of formulas by juxtaposition. Their length is assumed to be finite. The empty sequence is written ϵ. A sequence with length 1 is not distinguished from its sole member. In contexts where a set is expected, a sequence stands for the set of its members. Atoms are written in the form $p(\boldsymbol{t})$, where $\boldsymbol{t}$ is a sequence of terms whose length is the arity of the predicate p. Atoms of the form $p(\epsilon)$, that is, with a nullary predicate p, are written also as p. For a sequence of *fresh* symbols we assume that its members are distinct. A sequence $p_1 \dots p_n$ of predicates is said to *match* another sequence $q_1 \dots q_m$ if and only if $n = m$ and for all $i \in \{1, \dots, n\}$ it holds that $\mathsf{arity}(p_i) = \mathsf{arity}(q_i)$. If $\boldsymbol{s} = s_1 \dots s_n$ is a sequence of symbols, then $\forall \boldsymbol{s}$ stands for $\forall s_1 \dots \forall s_n$ and $\exists \boldsymbol{s}$ for $\exists s_1 \dots \exists s_n$.

As explained below, in certain contexts the individual symbols in the set $\mathcal{X} = \{x_i \mid i \geq 1\}$ play a special role. For example in the following shorthands for a predicate p, a formula F and $\boldsymbol{x} = x_1 \dots x_{\mathsf{arity}(p)}$: $p \Leftrightarrow F$ stands for $\forall \boldsymbol{x}\, (p(\boldsymbol{x}) \leftrightarrow F)$; $p \not\Leftrightarrow F$ for $\neg(p \Leftrightarrow F)$; $p \Rightarrow F$ for $\forall \boldsymbol{x}\, (p(\boldsymbol{x}) \rightarrow F)$; and $p \Leftarrow F$ for $\forall \boldsymbol{x}\, (p(\boldsymbol{x}) \leftarrow F)$.

2.2 Substitution with Terms and Formulas

To express systematic substitution of individual symbols and predicates concisely we use the following notation:

- $F(\boldsymbol{c})$ *and* $F(\boldsymbol{t})$ – *Notational Context for Substitution of Individual Symbols.* Let $\boldsymbol{c} = c_1 \dots c_n$ be a sequence of distinct individual symbols. We write F as $F(\boldsymbol{c})$ to declare that for a sequence $\boldsymbol{t} = t_1 \dots t_n$ of terms the expression $F(\boldsymbol{t})$ denotes F with, for $i \in \{1, \dots, n\}$, all free occurrences of c_i replaced by t_i.
- $F[\boldsymbol{p}]$, $F[\boldsymbol{G}]$ *and* $F[\boldsymbol{q}]$ – *Notational Context for Substitution of Predicates.* Let $\boldsymbol{p} = p_1 \dots p_n$ be a sequence of distinct predicates and let F be a formula. We write F as $F[\boldsymbol{p}]$ to declare the following:
  - For a sequence $\boldsymbol{G} = G_1(x_1 \dots x_{\mathsf{arity}(p_1)}) \dots G_n(x_1 \dots x_{\mathsf{arity}(p_n)})$ of formulas the expression $F[\boldsymbol{G}]$ denotes F with, for $i \in \{1, \dots, n\}$, each atom occurrence $p_i(t_1 \dots t_{\mathsf{arity}(p_i)})$ where p_i is free in F replaced by $G_i(t_1 \dots t_{\mathsf{arity}(p_i)})$.
  - For a sequence $\boldsymbol{q} = q_1 \dots q_n$ of predicates that matches $\boldsymbol{p}$ the expression $F[\boldsymbol{q}]$ denotes F with, for $i \in \{1, \dots, n\}$, each free occurrence of p_i replaced by q_i.

- The above notation $F[\boldsymbol{S}]$, where $\boldsymbol{S}$ is a sequence of formulas or of predicates, is generalized to allow also p_i at the ith position of $\boldsymbol{S}$, for example $F[G_1 \ldots G_{i-1} p_i \ldots p_n]$. The formula $F[\boldsymbol{S}]$ then denotes F with only those predicates p_i with $i \in \{1, \ldots, n\}$ that are not present at the ith position in $\boldsymbol{S}$ replaced by the ith component of $\boldsymbol{S}$ as described above (in the example only $p_1, \ldots, p_{i-1}$ would be replaced).

– $\boldsymbol{F}[\boldsymbol{p}]$ – *Notational Context for Substitution in a Sequence of Formulas.* If $\boldsymbol{F} = F_1 \ldots F_n$ is a sequence of formulas, then $\boldsymbol{F}[\boldsymbol{p}]$ declares that $\boldsymbol{F}[\boldsymbol{S}]$, where $\boldsymbol{S}$ is a sequence with the same length as $\boldsymbol{p}$, is to be understood as the sequence $F_1[\boldsymbol{S}] \ldots F_n[\boldsymbol{S}]$ with the meaning of the members as described above.

In the above notation for substitution of predicates by formulas the members $x_1, \ldots, x_{\mathsf{arity}(p)}$ of $\mathcal{X}$ play a special role: $F[\boldsymbol{G}]$ can be alternatively considered as obtained by replacing predicates p_i with λ-expressions $\lambda x_1 \ldots \lambda x_{\mathsf{arity}(p_i)}.G_i$ followed by β-conversion. The shorthand $p \Leftrightarrow F$ can be correspondingly considered as $p \leftrightarrow \lambda x_1 \ldots \lambda x_{\mathsf{arity}(p)}.G$. The following property *substitutible* specifies preconditions for meaningful simultaneous substitution of formulas for predicates:

Definition 1 ($\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, F)$ – **Substitutible Sequence of Formulas).** A sequence $\boldsymbol{G} = G_1 \ldots G_m$ of formulas is called *substitutible for* a sequence $\boldsymbol{p} = p_1 \ldots p_n$ of distinct predicates *in* a formula F, written $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, F)$, if and only if $m = n$ and for all $i \in \{1, \ldots, n\}$ it holds that (1.) No free occurrence of p_i in F is in the scope of a quantifier occurrence that binds a member of $\mathsf{free}(G_i)$; (2.) $\mathsf{free}(G_i) \cap \boldsymbol{p} = \emptyset$; and (3.) $\mathsf{free}(G_i) \cap \{x_j \mid j > \mathsf{arity}(p_i)\} = \emptyset$.

The following propositions demonstrate the introduced notation for formula substitution. It is well known that terms can be "pulled out of" and "pushed in to" atoms, justified by the equivalences $p(t_1 \ldots t_n) \;\equiv\; \exists x_1 \ldots \exists x_n \, (p(x_1 \ldots x_n) \wedge \bigwedge_{i=1}^{n} x_i = t_i) \;\equiv\; \forall x_1 \ldots \forall x_n \, (p(x_1 \ldots x_n) \vee \bigvee_{i=1}^{n} x_i \neq t_i)$, which hold if no member of $\{x_1, \ldots, x_n\}$ does occur in the terms $t_1, \ldots, t_n$. Analogously, substitutible subformulas can be "pulled out of" and "pushed in to" formulas:

Proposition 2 (Pulling-Out and Pushing-In of Subformulas). *Let $\boldsymbol{G} = G_1 \ldots G_n$ be a sequence of formulas, let $\boldsymbol{p} = p_1 \ldots p_n$ be a sequence of distinct predicates and let $F = F[\boldsymbol{p}]$ be a formula such that $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, F)$. Then*

(i) $F[\boldsymbol{G}] \;\equiv\; \exists \boldsymbol{p} \, (F \wedge \bigwedge_{i=1}^{n} (p_i \Leftrightarrow G_i)) \;\equiv\; \forall \boldsymbol{p} \, (F \vee \bigvee_{i=1}^{n} (p_i \not\Leftrightarrow G_i))$.
(ii) $\forall \boldsymbol{p} \, F \;\models\; F[\boldsymbol{G}] \;\models\; \exists \boldsymbol{p} \, F$.

Ackermann's Lemma [1] can be applied in certain cases to *eliminate* second-order quantifiers, that is, to compute for a given second-order formula an equivalent first-order formula. It plays an important role in many modern methods for elimination and semantic forgetting – see, e.g., [10,11,13,17,26,31]:

Proposition 3 (Ackermann's Lemma, Positive Version). *Let F, G be formulas and let p be a predicate such that $\mathsf{SUBST}(G, p, F)$, $p \notin \mathsf{free}(G)$ and all free occurrences of p in F have negative polarity. Then $\exists p \, ((p \Leftarrow G) \wedge F[p]) \;\equiv\; F[G]$.*

3 The Solution Problem from Different Angles

3.1 Basic Formal Modeling

Our formal modeling of the Boolean solution problem is based on two concepts, *solution problem* and *particular solution*:

Definition 4 ($F[\boldsymbol{p}]$ – Solution Problem (SP), Unary Solution Problem (1-SP)). A *solution problem (SP)* $F[\boldsymbol{p}]$ is a pair of a formula F and a sequence $\boldsymbol{p}$ of distinct predicates. The members of $\boldsymbol{p}$ are called the *unknowns* of the SP. The length of $\boldsymbol{p}$ is called the *arity* of the SP. A SP with arity 1 is also called *unary solution problem (1-SP)*.

The notation $F[\boldsymbol{p}]$ for solution problems establishes as a "side effect" a context for specifying substitutions of $\boldsymbol{p}$ in F by formulas as specified in Sect. 2.2.

Definition 5 (Particular Solution). A *particular solution* (briefly *solution*) *of* a SP $F[\boldsymbol{p}]$ is defined as a sequence $\boldsymbol{G}$ of formulas such that $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, F)$ and $\models F[\boldsymbol{G}]$.

The property $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, F)$ in this definition implies that no member of $\boldsymbol{p}$ occurs free in a solution. Of course, *particular solution* can also be defined on the basis of unsatisfiability instead of validity, justified by the equivalence of $\models F[\boldsymbol{G}]$ and $\neg F[\boldsymbol{G}] \models \bot$. The variant based on validity has been chosen here because then the associated second-order quantifications are existential, matching the usual presentation of elimination techniques.

Solution problem and *solution* as defined here provide abstractions of computational problems in a technical sense that would be suitable, e.g., for complexity analysis. Problems in the latter sense can be obtained by fixing involved formula and predicate classes. The abstract notions are adequate to develop much of the material on the "Boolean solution problem" shown here and in [30]. On occasion, however, we consider restrictions, in particular to propositional and to first-order formulas, as well as to nullary predicates. As shown in [30, Sect. 6], further variants of *solution*, general representations of several particular solutions, can be introduced on the basis of the notions defined here.

Example 6 (A Solution Problem and its Particular Solutions). As an example of a solution problem consider $F[p_1 p_2]$ where

$$\begin{aligned} F = &\forall x\,(a(x) \rightarrow b(x)) \;\rightarrow \\ &(\forall x\,(p_1(x) \rightarrow p_2(x)) \;\wedge\; \forall x\,(a(x) \rightarrow p_2(x)) \;\wedge\; \forall x\,(p_2(x) \rightarrow b(x))). \end{aligned}$$

The intuition is that the antecedent $\forall x\,(a(x) \rightarrow b(x))$ specifies the "background theory", and w.r.t. that theory the unknown p_1 is "stronger" than the other unknown p_2, which is also "between" a and b. Examples of solutions are: $a(x_1)a(x_1)$; $a(x_1)b(x_1)$; $\bot a(x_1)$; $b(x_1)b(x_1)$; and $(a(x_1) \wedge b(x_1))(a(x_1) \vee b(x_1))$. No solutions are for example $b(x_1)a(x_1)$; $a(x_1)\bot$; and all members of $\{\top, \bot\} \times \{\top, \bot\}$.

Assuming a countable vocabulary, the set of valid first-order formulas is recursively enumerable. It follows that for an n-ary SP $F[\boldsymbol{p}]$ where F is first-order the set of those of its particular solutions that are sequences of first-order formulas is also recursively enumerable: An n-ary sequence $\boldsymbol{G}$ of well-formed first-order formulas that satisfies the syntactic restriction $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, F)$ is a solution of $F[\boldsymbol{p}]$ if and only if $F[\boldsymbol{G}]$ is valid.

In the following subsections further views on the solution problem will be discussed: as unification or equation solving, as a special case of second-order quantifier elimination, and as related to determining definientia and interpolants.

3.2 View as Unification

Because $\models F[\boldsymbol{G}]$ if and only if $F[\boldsymbol{G}] \equiv \top$, a particular solution of $F[\boldsymbol{p}]$ can be seen as a unifier of the two formulas $F[\boldsymbol{p}]$ and $\top$ modulo logical equivalence as equational theory. From the perspective of unification the two formulas appear as terms, the members of $\boldsymbol{p}$ play the role of variables and the other predicates play the role of constants.

Vice versa, a unifier of two formulas can be seen as a particular solution, justified by the equivalence of $L[\boldsymbol{G}] \equiv R[\boldsymbol{G}]$ and $\models (L \leftrightarrow R)[\boldsymbol{G}]$, which holds for sequences $\boldsymbol{G}$ and $\boldsymbol{p}$ of formulas and predicates, respectively, and formulas $L = L[\boldsymbol{p}], R = R[\boldsymbol{p}]$, $(L \leftrightarrow R) = (L \leftrightarrow R)[\boldsymbol{p}]$ such that $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, L)$ and $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, R)$. This view of formula unification can be generalized to sets with a finite cardinality k of equivalences, since *for all* $i \in \{1, \ldots, k\}$ *it holds that* $L_i \equiv R_i$ can be expressed as $\models \bigwedge_{i=1}^{k} (L_i \leftrightarrow R_i)$.

An exact correspondence between solving a solution problem $F[p_1 \ldots p_n]$ where F is a propositional formula with $\vee, \wedge, \neg, \bot, \top$ as logic operators and E-unification with constants in the theory of Boolean algebra (with the mentioned logic operators as signature) applied to $F =_E \top$ can be established: Unknowns $p_1, \ldots, p_n$ correspond to variables and propositional atoms in F correspond to constants. A particular solution $G_1 \ldots G_n$ corresponds to a unifier $\{p_1 \leftarrow G_1, \ldots, p_n \leftarrow G_n\}$ that is a ground substitution. The restriction to ground substitutions is due to the requirement that unknowns do not occur in solutions. General solutions [30, Sect. 6] are expressed with further special parameter atoms, different from the unknowns. These correspond to fresh variables in unifiers.

A generalization of Boolean unification to predicate logic with various specific problems characterized by the involved formula classes has been investigated in [12]. The material presented here and in [30] is largely orthogonal to that work, but a technique from [12] has been adapted to more general cases in [30, Sect. 7.3].

3.3 View as Construction of Elimination Witnesses

Another view on the solution problem is related to eliminating second-order quantifiers by replacing the quantified predicates with "witness formulas".

Definition 7 (ELIM-Witness). Let $\boldsymbol{p} = p_1 \dots p_n$ be a sequence of distinct predicates. An *ELIM-witness of* $\boldsymbol{p}$ *in* a formula $\exists \boldsymbol{p}\, F[\boldsymbol{p}]$ is defined as a sequence $\boldsymbol{G}$ of formulas such that $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, F)$ and $\exists \boldsymbol{p}\, F[\boldsymbol{p}] \equiv F[\boldsymbol{G}]$.

The condition $\exists \boldsymbol{p}\, F[\boldsymbol{p}] \equiv F[\boldsymbol{G}]$ in this definition is equivalent to $\models \neg F[\boldsymbol{p}] \vee F[\boldsymbol{G}]$. If $F[\boldsymbol{p}]$ and the considered $\boldsymbol{G}$ are first-order, then finding an ELIM-witness is second-order quantifier elimination on a first-order argument formula, restricted by the condition that the result is of the form $F[\boldsymbol{G}]$. Differently from the general case of second-order quantifier elimination on first-order arguments, the set of formulas for which elimination succeeds and, for a given formula, the set of its elimination results, are then recursively enumerable. Some well-known elimination methods yield ELIM-witnesses, for example rewriting a formula that matches the left side of Ackermann's Lemma (Proposition 3) with its right side, which becomes evident when considering that the right side $F[G]$ is equivalent to $\forall x_1 \dots \forall x_{\mathsf{arity}(p)}\, (G \leftarrow G) \wedge F[G]$. Finding particular solutions and finding ELIM-witnesses can be expressed in terms of each other:

Proposition 8 (Solutions and ELIM-Witnesses). *Let $F[\boldsymbol{p}]$ be SP and let $\boldsymbol{G}$ be a sequence of formulas. Then:*

(i) $\boldsymbol{G}$ is an ELIM-witness of $\boldsymbol{p}$ in $\exists \boldsymbol{p}\, F$ if and only if $\boldsymbol{G}$ is a solution of the SP $(\neg F[\boldsymbol{q}] \vee F)[\boldsymbol{p}]$, where $\boldsymbol{q}$ is a sequence of fresh predicates matching $\boldsymbol{p}$.
(ii) $\boldsymbol{G}$ is a solution of $F[\boldsymbol{p}]$ if and only if $\boldsymbol{G}$ is an ELIM-witness of $\boldsymbol{p}$ in $\exists \boldsymbol{p}\, F$ and it holds that $\models \exists \boldsymbol{p}\, F$.

Proof (Sketch). Assume $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, F)$. (Proposition 8i) Follows since $\exists \boldsymbol{p}\, F[\boldsymbol{p}] \equiv F[\boldsymbol{G}]$ iff $\exists \boldsymbol{p}\, F[\boldsymbol{p}] \models F[\boldsymbol{G}]$ iff $F[\boldsymbol{p}] \models F[\boldsymbol{G}]$ iff $\models \neg F[\boldsymbol{q}] \vee F[\boldsymbol{G}]$. (Proposition 8ii) Left-To-Right: Follows since $\models F[\boldsymbol{G}]$ implies $\models \exists \boldsymbol{p}\, F[\boldsymbol{p}]$ and $\models F[\boldsymbol{G}]$, which implies $\exists \boldsymbol{p}\, F[\boldsymbol{p}] \equiv \top \equiv F[\boldsymbol{G}]$. Right-to-left: Follows since $\exists \boldsymbol{p}\, F[\boldsymbol{p}] \equiv F[G]$ and $\models \exists \boldsymbol{p}\, F[\boldsymbol{p}]$ together imply $\models F[\boldsymbol{G}]$. □

3.4 View as Related to Definientia and Interpolants

The following proposition shows a further view on the solution problem that relates it to definitions of the unknown predicates:

Proposition 9 (Solution as Entailed by a Definition). *A sequence $\boldsymbol{G} = G_1 \dots G_n$ of formulas is a particular solution of a SP $F[\boldsymbol{p} = p_1 \dots p_n]$ if and only if $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, F)$ and $\bigwedge_{i=1}^{n} (p_i \Leftrightarrow G_i) \models F$.*

Proof. Follows from the definition of *particular solution* and Proposition 2i. □

In the special case where $F[p]$ is a 1-SP with a *nullary* unknown p, the characterization of a solution G according to Proposition 9 can be expressed with an entailment where a definition of the unknown p appears on the right instead of the left side: If p is nullary, then $\neg(p \Leftrightarrow G) \equiv p \Leftrightarrow \neg G$. Thus, the statement $p \Leftrightarrow G \models F$ is for nullary p equivalent to

$$\neg F \models p \Leftrightarrow \neg G. \tag{i}$$

The second condition of the characterization of *solution* according to Proposition 9, that is, $\mathsf{SUBST}(G, p, F)$, holds if it is assumed that p is not in $\mathsf{free}(G)$, that $\mathsf{free}(G) \subseteq \mathsf{free}(F)$ and that no member of $\mathsf{free}(F)$ is bound by a quantifier occurrence in F. A solution is then characterized as negated definiens of p in the negation of F. Another way to express (i) along with the condition that G is semantically independent from p is as follows:

$$\exists p\,(\neg F \wedge \neg p) \;\models\; G \;\models\; \neg\exists p\,(\neg F \wedge p). \tag{ii}$$

The second-order quantifiers upon the nullary p can be eliminated, yielding the following equivalent statement:

$$\neg F[\bot] \;\models\; G \;\models\; F[\top]. \tag{iii}$$

Solutions G then appear as the formulas in a range, between $\neg F[\bot]$ and $F[\top]$. This view is reflected in [23, Theorem 2.2], which goes back to work by Schröder. If F is first-order, then Craig interpolation can be applied to compute formulas G that also meet the requirements $\mathsf{free}(G) \subseteq \mathsf{free}(F)$ and $p \notin \mathsf{free}(F)$ to ensure $\mathsf{SUBST}(G, p, F)$. Further connections to Craig interpolation are discussed in [30, Sect. 7].

4 The Method of Successive Eliminations – Abstracted

4.1 Reducing n-ary to 1-ary Solution Problems

The *method of successive eliminations* to solve an n-ary solution problem by reducing it to unary solution problems is attributed to Boole and has been formally described in a modern algebraic setting in [23, Chap. 2, Sect. 4]. It has been rediscovered in the context of Boolean unification in the late 1980s, notably with [8]. Rudeanu notes in [23, p. 72] that variants described by several authors in the 19th century are discussed by Schröder [27, vol. 1, Sects. 26 and 27]. To research and compare all variants up to now seems to be a major undertaking on its own. Our aim is here to provide a foundation to derive and analyze related methods. The following proposition formally states the core property underlying the method in a way that, compared to the Boolean algebra version in [23, Chap. 2, Sect. 4], is more abstract in several aspects: Second-order quantification upon predicates that represent unknowns plays the role of meta-level shorthands that encode expansions; no commitment to a particular formula class is made, thus the proposition applies to second-order formulas with first-order and propositional formulas as special cases; it is not specified how solutions of the arising unary solution problems are constructed; and it is not specified how intermediate second-order formulas (that occur also for inputs without second-order quantifiers) are handled. The algorithm descriptions in the following subsections show different possibilities to instantiate these abstracted aspects.

Proposition 10 (Characterization of Solution Underlying the Method of Successive Eliminations). *Let $F[\boldsymbol{p} = p_1 \ldots p_n]$ be a SP and let $\boldsymbol{G} = G_1 \ldots G_n$ be a sequence of formulas. Then the following statements are equivalent:*

(a) $\boldsymbol{G}$ *is a solution of* $F[\boldsymbol{p}]$.
(b) For $i \in \{1, \ldots, n\}$: G_i *is a solution of the 1-SP*

$$(\exists p_{i+1} \ldots \exists p_n \, F[G_1 \ldots G_{i-1} p_i \ldots p_n])[p_i]$$

such that $\mathsf{free}(G_i) \cap \boldsymbol{p} = \emptyset$.

Proof. Left-to-right: From (a) it follows that $\models F[\boldsymbol{G}]$. Hence, for all $i \in \{1, \ldots, n\}$ by Proposition 2ii it follows that

$$\models \exists p_{i+1} \ldots \exists p_n \, F[G_1 \ldots G_i p_{i+1} \ldots p_n].$$

From (a) it also follows that $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, F)$. This implies that for all $i \in \{1, \ldots, n\}$ it holds that

$$\mathsf{SUBST}(G_i, p_i, \exists p_{i+1} \ldots \exists p_n \, F[G_1 \ldots G_{i-1} p_i \ldots p_n]) \text{ and } \mathsf{free}(G_i) \cap \boldsymbol{p} = \emptyset.$$

We thus have derived for all $i \in \{1, \ldots, n\}$ the two properties that characterize G_i as a solution of the 1-SP as stated in (b).

Right-to-left: From (b) it follows that G_n is a solution of the 1-SP

$$(F[G_1 \ldots G_{n-1} p_n])[p_n].$$

Hence, by the characteristics of *solution* it follows that $\models F[G_1 \ldots G_n]$. The property $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, F)$ can be derived from $\mathsf{free}(\boldsymbol{G}) \cap \boldsymbol{p} = \emptyset$ and the fact that for all $i \in \{1, \ldots, n\}$ it holds that $\mathsf{SUBST}(G_i, p_i, (\exists p_{i+1} \ldots \exists p_n \, F[G_1 \ldots G_{i-1} p_i \ldots p_n]))$. The properties $\models F[G_1 \ldots G_n]$ and $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, F)$ characterize $\boldsymbol{G}$ as a solution of the SP $F[\boldsymbol{p}]$. □

This proposition states an equivalence between the solutions of an n-ary SP and the solutions of n 1-SPs. These 1-SPs are on formulas with an existential second-order prefix. The following gives an example of this decomposition:

Example 11 (Reducing an n-ary Solution Problem to Unary Solution Problems). Consider the SP $F[p_1 p_2]$ of Example 6. The 1-SP with unknown p_1 according to Proposition 10 is

$$(\exists p_2 \, F[p_1 p_2])[p_1],$$

whose formula is, by second-order quantifier elimination, equivalent to $\forall x \, (a(x) \rightarrow b(x)) \rightarrow \forall x \, (p_1(x) \rightarrow b(x))$. Take $a(x_1)$ as solution G_1 of that 1-SP. The 1-SP with unknown p_2 according to Proposition 10 is

$$(F[G_1 p_2])[p_2].$$

Its formula is then, by replacing p_1 in F as specified in Example 6 with a and removing the duplicate conjunct obtained then, equivalent to

$$\forall x \, (a(x) \rightarrow b(x)) \;\rightarrow\; (\forall x \, (a(x) \rightarrow p_2(x)) \;\wedge\; \forall x \, (p_2(x) \rightarrow b(x))).$$

A solution of that second 1-SP is, for example, $b(x_1)$, yielding the pair $a(x_1)b(x_1)$ as solution of the originally considered SP $F[p_1 p_2]$.

4.2 Solving on the Basis of Second-Order Formulas

The following algorithm to compute particular solutions is an immediate transfer of Proposition 10. Actually, it is more an "algorithm template", since it is parameterized with a method to compute 1-SPs and covers a nondeterministic as well as a deterministic variant:

Algorithm 12 (SOLVE-ON-SECOND-ORDER). *Let $\mathcal{F}$ be a class of formulas and let 1-SOLVE be a nondeterministic or a deterministic algorithm that outputs for 1-SPs of the form $(\exists p_1 \ldots \exists p_n\, F[p])[p]$ with $F \in \mathcal{F}$ solutions G such that $\mathsf{free}(G) \cap \{p_1, \ldots, p_n\} = \emptyset$ and $F[G] \in \mathcal{F}$.*
INPUT: *A SP $F[p_1 \ldots p_n]$, where $F \in \mathcal{F}$, that has a solution.*
METHOD: *For $i := 1$ to n do: Assign to G_i an output of 1-SOLVE applied to the 1-SP $(\exists p_{i+1} \ldots \exists p_n\, F[G_1 \ldots G_{i-1} p_i \ldots p_n])[p_i]$.*
OUTPUT: *The sequence $G_1 \ldots G_n$ of formulas, which is a particular solution of $F[p_1 \ldots p_n]$.*

The solution components G_i are successively assigned to some solution of the 1-SP given in Proposition 10, on the basis of the previously assigned components $G_1 \ldots G_{i-1}$. Even if the formula F of the input problem does not involve second-order quantification, these 1-SPs are on second-order formulas with an existential prefix $\exists p_{i+1} \ldots \exists p_n$ upon the yet "unprocessed" unknowns.

The algorithm comes in a nondeterministic and a deterministic variant, just depending on whether *1-SOLVE* is instantiated by a nondeterministic or a deterministic algorithm. Thus, in the nondeterministic variant the nondeterminism of *1-SOLVE* is the only source of nondeterminism. With Proposition 10 it can be verified that if a nondeterministic *1-SOLVE* is "complete" in the sense that for each solution there is an execution path that leads to the output of that solution, then also SOLVE-ON-SECOND-ORDER based on it enjoys that property, with respect to the n-ary solutions $G_1 \ldots G_n$.

For the deterministic variant, from Proposition 10 it follows that if *1-SOLVE* is "complete" in the sense that it outputs some solution whenever a solution exists, then, given that $F[p_1 \ldots p_n]$ has a solution, which is ensured by the specification of the input, also SOLVE-ON-SECOND-ORDER outputs some solution $G_1 \ldots G_n$.

This method applies *1-SOLVE* to existential second-order formulas, which prompts some issues for future research: As indicated in Sect. 3.4 (and elaborated in [30, Sect. 7]) Craig interpolation can in certain cases be applied to compute solutions of 1-SPs. Can QBF solvers, perhaps those that encode QBF into predicate logic [28], be utilized to compute Craig interpolants? Can it be useful to allow second-order quantifiers in solution formulas because they make these smaller and can be passed between different calls to *1-SOLVE*?

As shown in [30, Sect. 6], if *1-SOLVE* is a method that outputs so-called *reproductive* solutions, that is, most general solutions that represent all particular solutions, then also SOLVE-ON-SECOND-ORDER outputs reproductive solutions. Thus, there are two ways to obtain representations of all particular solutions whose comparison might be potentially interesting: A deterministic

method that outputs a single reproductive solution and the nondeterministic method with an execution path to each particular solution.

4.3 Solving with the Method of Successive Eliminations

The *method of successive eliminations* in a narrower sense is applied in a Boolean algebra setting that corresponds to propositional logic and outputs reproductive solutions. The consideration of reproductive solutions belongs to the classical material on Boolean reasoning [19,23,27] and is modeled in the present framework in [30, Sect. 6]. Compared to SOLVE-ON-SECOND-ORDER, the method handles the second-order quantification by eliminating quantifiers one-by-one, inside-out, with a specific method and applies a specific method to solve 1-SPs, which actually yields reproductive solutions. These incorporated methods apply to propositional input formulas (and to first-order input formulas if the unknowns are nullary). Second-order quantifiers are eliminated by rewriting with the equivalence $\exists p\, F[p] \equiv F[\top] \vee F[\bot]$. As solution of an 1-SP $F[p]$ the formula $(\neg F[\bot] \wedge t) \vee (F[\top] \wedge \neg t)$ is taken, where t is a fresh nullary predicate that is considered specially. The intuition is that particular solutions are obtained by replacing t with arbitrary formulas in which p does not occur (see [30, Sect. 6] for a more in-depth discussion).

The following algorithm is an iterative presentation of the *method of successive eliminations*, also called *Boole's method*, in the variant due to [8]. The presentation in [22, Sect. 3.1], where apparently minor corrections compared to [8] have been made, has been taken here as technical basis. We stay in the validity-based setting, whereas [8,22,23] use the unsatisfiability-based setting. Also differently from [8,22] we do not make use of the *xor* operator.

Algorithm 13 (SOLVE-SUCC-ELIM).
INPUT: *A SP $F[p_1 \ldots p_n]$, where F is propositional, that has a solution and a sequence $t_1 \ldots t_n$ of fresh nullary predicates.*
METHOD:

1. *Initialize $F_n[p_1 \ldots p_n]$ with F.*
2. *For $i := n$ to 1 do: Assign to $F_{i-1}[p_1 \ldots, p_{i-1}]$ the formula $F_i[p_1 \ldots p_{i-1} \top] \vee F_i[p_1 \ldots p_{i-1} \bot]$.*
3. *For $i := 1$ to n do: Assign to G_i the formula $(\neg F_i[G_1 \ldots G_{i-1} \bot] \wedge t_i) \vee (F_i[G_1 \ldots G_{i-1} \top] \wedge \neg t_i)$.*

OUTPUT: *The sequence $G_1 \ldots G_n$ of formulas, which is a reproductive solution of $F[p_1 \ldots p_n]$ with respect to the special predicates $t_1 \ldots t_n$.*

The formula assigned to F_{i-1} in step (2.) is the result of eliminating $\exists p_i$ in $\exists p_i\, F_i[p_1 \ldots p_i]$ and the formula assigned to G_i in step (3.) is the reproductive solution of the 1-SP $(F_i[G_1 \ldots G_{i-1} p_i])[p_i]$, obtained with the respective incorporated methods indicated above. The recursion in the presentations of [8,22] is translated here into two iterations that proceed in opposite directions: First, existential quantifiers of $\exists p_1 \ldots \exists p_n\ F$ are eliminated inside-out and the

intermediate results, which do not involve second-order quantifiers, are stored. Solutions of 1-SPs are computed in the second phase on the basis of the stored formulas.

In this presentation it is easy to identify two "hooks" where it is possible to plug-in alternate methods that produce other outputs or apply to further formula classes: In step (2.) the elimination method and in step (3.) the method to determine solutions of 1-SPs. If the plugged-in method to compute 1-SPs outputs particular solutions, then SOLVE-SUCC-ELIM computes particular instead of reproductive solutions.

4.4 Solving by Inside-Out Witness Construction

Like SOLVE-SUCC-ELIM, the following algorithm eliminates second-order quantifiers one-by-one, inside-out, avoiding intermediate formulas with existential second-order prefixes of length greater than 1, which arise with SOLVE-ON-SECOND-ORDER. In contrast to SOLVE-SUCC-ELIM, it performs elimination by the computation of ELIM-witnesses.

Algorithm 14 (SOLVE-BY-WITNESESS). *Let $\mathcal{F}$ be a class of formulas and ELIM-WITNESS be an algorithm that computes for formulas $F \in \mathcal{F}$ and predicates p an ELIM-witness G of p in $\exists p\, F[p]$ such that $F[G] \in \mathcal{F}$.*
INPUT: *A SP $F[p_1 \ldots p_n]$, where $F \in \mathcal{F}$, that has a solution.*
METHOD: *For $i := n$ to 1 do:*

1. *Assign to $G_i[p_1 \ldots p_{i-1}]$ the output of ELIM-WITNESS applied to*

$$\exists p_i\, F[p_1 \ldots p_i G_{i+1} \ldots G_n].$$

2. *For $j := n$ to $i+1$ do: Re-assign to $G_j[p_1 \ldots p_{i-1}]$ the formula $G_j[p_1 \ldots p_{i-1} G_i]$.*

OUTPUT: *: The sequence $G_1 \ldots G_n$ of formulas, which provides a particular solution of $F[p_1 \ldots p_n]$.*

Step (2.) in the algorithm expresses that a new value is assigned to G_j and that G_j can be designated by $G_j[p_1 \ldots p_{i-1}]$, justified because the new value does not contain free occurrences of $p_i, \ldots, p_n$. In step (1.) the respective current values of $G_{i+1} \ldots G_n$ are used to instantiate F. It is not hard to see from the specification of the algorithm that for input $F[\boldsymbol{p}]$ and output $\boldsymbol{G}$ it holds that $\exists \boldsymbol{p}\, F \equiv F[\boldsymbol{G}]$ and that $\mathsf{SUBST}(\boldsymbol{G}, \boldsymbol{p}, F)$. By Proposition 8ii, $\boldsymbol{G}$ is then a solution if $\models \exists \boldsymbol{p}\, F$. This holds indeed if $F[\boldsymbol{p}]$ has a solution, as shown below with Proposition 15.

If *ELIM-WITNESS* is "complete" in the sense that it computes an elimination witness for all input formulas in $\mathcal{F}$, then SOLVE-BY-WITNESESS outputs a solution. Whether all solutions of the input SP can be obtained as outputs for different execution paths of a nondeterministic version of SOLVE-BY-WITNESESS obtained through a nondeterministic *ELIM-WITNESS*, in analogy to the nondeterministic variant of SOLVE-ON-SECOND-ORDER, appears to be an open problem.

5 Existence of Solutions

5.1 Conditions for the Existence of Solutions

We now turn to the question under which conditions there exists a solution of a given SP, or, in the terminology of [23], the SP is *consistent*. A necessary condition is easy to see:

Proposition 15 (Necessary Condition for the Existence of a Solution). *If a SP $F[\boldsymbol{p}]$ has a solution, then it holds that $\models \exists \boldsymbol{p}\, F$.*

Proof. Follows from the definition of *particular solution* and Proposition 2ii. □

Under certain presumptions that hold for propositional logic this condition is also sufficient. To express these abstractly we use the following concept:

Definition 16 (SOL-Witnessed Formula Class). A formula class $\mathcal{F}$ is called *SOL-switnessed for* a predicate class $\mathcal{P}$ if and only if for all $p \in \mathcal{P}$ and $F[p] \in \mathcal{F}$ the following statements are equivalent:

(a) $\models \exists p\, F$.
(b) There exists a solution G of the 1-SP $F[p]$ such that $F[G] \in \mathcal{F}$.

Since the right-to-left direction of that equivalence holds in general, the left-to-right direction alone would provide an alternate characterization. The class of propositional formulas is SOL-witnessed (for the class of nullary predicates). This follows since in propositional logic it holds that

$$\exists p\, F[p] \equiv F[F[\top]], \tag{iv}$$

which can be derived in the following steps: $F[F[\top]] \equiv \exists p\,(F[p] \wedge (p \leftrightarrow F[\top])) \equiv (F[\top] \wedge (\top \leftrightarrow F[\top])) \vee (F[\bot] \wedge (\bot \leftrightarrow F[\top])) \equiv F[\top] \vee F[\bot] \equiv \exists p\, F[p]$.

The following definition adds closedness under existential second-order quantification to the notion of *SOL-witnessed*, to allow the application on 1-SPs matching with item (b) in Proposition 10:

Definition 17 (MSE-SOL-Witnessed Formula Class). A formula class $\mathcal{F}$ is called *MSE-SOL-witnessed* for a predicate class $\mathcal{P}$ if and only if it is SOL-witnessed for $\mathcal{P}$ and for all sequences $\boldsymbol{p}$ of predicates in $\mathcal{P}$ and $F \in \mathcal{F}$ it holds that $\exists \boldsymbol{p}\, F \in \mathcal{F}$.

The class of existential QBFs (formulas of the form $\exists \boldsymbol{p}\, F$ where F is propositional) is MSE-SOL-witnessed (like the more general class of QBFs – second-order formulas with only nullary predicates). Another example is the class of first-order formulas extended by second-order quantification upon nullary predicates, which is MSE-SOL-witnessed for the class of nullary predicates. The following proposition can be seen as expressing an invariant of the method of successive eliminations that holds for formulas in an MSE-SOL-witnessed class:

Proposition 18 (Solution Existence Lemma). *Let $\mathcal{F}$ be a formula class that is MSE-SOL-witnessed for predicate class $\mathcal{P}$. Let $F[\boldsymbol{p} = p_1 \ldots p_n] \in \mathcal{F}$ with $\boldsymbol{p} \in \mathcal{P}^n$. If $\models \exists \boldsymbol{p}\, F[\boldsymbol{p}]$, then for all $i \in \{0, \ldots, n\}$ there exists a sequence $G_1 \ldots G_i$ of formulas such that $\mathsf{free}(G_1 \ldots G_i) \cap \boldsymbol{p} = \emptyset$, $\mathsf{SUBST}(G_1 \ldots G_i, p_1 \ldots p_i, F)$, $\models \exists p_{i+1} \ldots \exists p_n F[G_1 \ldots G_i p_{i+1} \ldots p_n]$ and $\exists p_{i+1} \ldots \exists p_n F[G_1 \ldots G_i p_{i+1} \ldots p_n] \in \mathcal{F}$.*

Proof. By induction on the length i of the sequence $G_1 \ldots G_i$. The conclusion of the proposition holds for the base case $i = 0$: The statement $\mathsf{SUBST}(\epsilon, \epsilon, F)$ holds trivially, $\models \exists \boldsymbol{p}\, F$ is given as precondition, and $\exists \boldsymbol{p}\, F \in \mathcal{F}$ follows from $F \in \mathcal{F}$. For the induction step, assume that the conclusion of the proposition holds for some $i \in \{0, \ldots n-1\}$. That is, $\mathsf{SUBST}(G_1 \ldots G_i, p_1 \ldots p_i, F)$, $\models \exists \boldsymbol{p}\, F[G_1 \ldots G_i p_{i+1} \ldots p_n]$ and $\exists \boldsymbol{p}\, F[G_1 \ldots G_i p_{i+1} \ldots p_n] \in \mathcal{F}$. Since $\mathcal{F}$ is witnessed for $\mathcal{P}$ and $p_{i+1} \in \mathcal{P}$ it follows that there exists a solution G_{i+1} of the 1-SP $(\exists \boldsymbol{p}\, F[G_1 \ldots G_i p_{i+1} \ldots p_n])[p_{i+1}]$ such that $(\exists \boldsymbol{p}\, F[G_1 \ldots G_{i+1} p_{i+2} \ldots p_n]) \in \mathcal{F}$. From the characteristics of *solution* it follows that $\models \exists \boldsymbol{p}\, F[G_1 \ldots G_{i+1} p_{i+2} \ldots p_n])$ and $\mathsf{SUBST}(G_{i+1}, p_{i+1}, \exists \boldsymbol{p}\, F[G_1 \ldots G_{i+1} p_{i+2} \ldots p_n])$. In the latter statement the quantifier $\exists \boldsymbol{p}$ ensures that $\mathsf{free}(G_{i+1}) \cap \boldsymbol{p} = \emptyset$. With the induction hypothesis $\mathsf{SUBST}(G_1 \ldots G_i, p_1 \ldots p_i, F)$ it follows that $\mathsf{SUBST}(G_1 \ldots G_{i+1}, p_1 \ldots p_{i+1}, F)$, which completes the proof of the induction step. (The existential quantification is here upon $\boldsymbol{p}$, not just $p_{i+1} \ldots p_n$, to ensure that no members of $\boldsymbol{p}$ at all occur as free symbols in the solutions.) □

A sufficient and necessary condition for the existence of a solution of formulas in MSE-SOL-witnessed classes now follows from Propositions 15 and 18:

Proposition 19 (Existence of a Solution). *Let $\mathcal{F}$ be a formula class that is MSE-SOL-witnessed on predicate class $\mathcal{P}$. Then for all $F[\boldsymbol{p}] \in \mathcal{F}$ where the members of $\boldsymbol{p}$ are in $\mathcal{P}$ the following statements are equivalent:*

(a) $\models \exists \boldsymbol{p}\, F$.
(b) There exists a solution $\boldsymbol{G}$ of the SP $F[\boldsymbol{p}]$ such that $F[\boldsymbol{G}] \in \mathcal{F}$.

Proof. Follows from Propositions 15 and 18. □

From that proposition it is easy to see that for SPs with propositional formulas the complexity of determining the existence of a solution is the same as the complexity of deciding validity of existential QBFs, as proven in [2,15,16], that is, Π_2^P-completeness: By Proposition 19, an SP $F[\boldsymbol{p}]$ where F is propositional has a solution if and only if the existential QBF $\exists \boldsymbol{p}\, F[\boldsymbol{p}]$ is valid and, vice versa, an arbitrary existential QBF $\exists \boldsymbol{p}\, F[\boldsymbol{p}]$ (where F is quantifier-free) is valid if and only if the SP $F[\boldsymbol{p}]$ has a solution.

5.2 Characterization of SOL-Witnessed in Terms of ELIM-Witness

The following proposition shows that under a minor syntactic precondition on formula classes, *SOL-witnessed* can also be characterized in terms of *ELIM-witness* instead of *solution* as in Definition 16:

Proposition 20 (SOL-Witnessed in Terms of ELIM-Witness). *Let $\mathcal{F}$ be a class of formulas that satisfies the following properties: For all $F[p] \in \mathcal{F}$ and predicates q with the same arity of p it holds that $F[p] \vee \neg F[q] \in \mathcal{F}$, and for all $F \vee G \in \mathcal{F}$ it holds that $F \in \mathcal{F}$. The class $\mathcal{F}$ is SOL-witnessed for a predicate class $\mathcal{P}$ if and only if for all $p \in \mathcal{P}$ and $F[p] \in \mathcal{F}$ there exists an ELIM-witness G of p in $F[p]$ such that $F[G] \in \mathcal{F}$.*

Proof. Left-to-right: Assume that $\mathcal{F}$ is meets the specified closedness conditions and is SOL-witnessed for $\mathcal{P}$, $p \in \mathcal{P}$ and $F[p] \in \mathcal{F}$. Let q be a fresh predicate with the arity of p. The obviously true statement $\models \exists p\, F[p] \vee \neg \exists p\, F[p]$ is equivalent to $\models \exists p\, F[p] \vee \neg F[q]$ and thus to $\models \exists p\, (F[p] \vee \neg F[q])$. By the closedness properties of $\mathcal{F}$ it holds that $F[p] \vee \neg F[q] \in \mathcal{F}$. Since $\mathcal{F}$ is SOL-witnessed for $\mathcal{P}$ it thus follows from Definition 16 that there exists a solution G of the SP $(F[p] \vee \neg F[q])[p]$ such that $(F[G] \vee \neg F[q]) \in \mathcal{F}$, and, by the closedness properties, also $F[G] \in \mathcal{F}$. From the definition of *solution* it follows that $\models F[G] \vee \neg F[q]$, which is equivalent to $\exists p\, F[p] \equiv F[G]$, and also that $\mathsf{SUBST}(G, p, F[G] \vee \neg F[q])$, which implies $\mathsf{SUBST}(G, p, F[G])$. Thus G is an SO-witness of p in $F[p]$ such that $F[G] \in \mathcal{F}$. Right-to-left: Easy to see from Proposition 8ii. □

5.3 The Elimination Result as Precondition of Solution Existence

Proposition 19 makes an interesting relationship between the existence of a solution and second-order quantifier elimination apparent that has been pointed out by Schröder [27, vol. 1, Sect. 21] and Behmann [5], and is briefly reflected in [23, p. 62]: The formula $\exists \boldsymbol{p}\, F$ is valid if and only if the result of eliminating the existential second-order prefix (called *Resultante* by Schröder [27, vol. 1, Sect. 21]) is valid. If it is not valid, then, by Proposition 19, the SP $F[\boldsymbol{p}]$ has no solution, however, in that case the elimination result represents the *unique (modulo equivalence) weakest precondition under which the SP would have a solution.* The following proposition shows a way to make this precise:

Proposition 21 (The Elimination Result is the Unique Weakest Precondition of Solution Existence). *Let $\mathcal{F}$ be a formula class and let $\mathcal{P}$ be a predicate class such that $\mathcal{F}$ is MSE-SOL-witnessed on $\mathcal{P}$. Let $F[\boldsymbol{p}]$ be a solution problem where $F \in \mathcal{F}$ and all members of $\boldsymbol{p}$ are in $\mathcal{P}$. Let A be a formula such that $(A \to F) \in \mathcal{F}$, $A \equiv \exists \boldsymbol{p}\, F$, and no member of $\boldsymbol{p}$ does occur in A. Then*

(i) The SP $(A \to F)[\boldsymbol{p}]$ has a solution.
(ii) If B is a formula such that $(B \to F) \in \mathcal{F}$, no member of $\boldsymbol{p}$ occurs in B, and the SP $(B \to F)[\boldsymbol{p}]$ has a solution, then $B \models A$.

Proof. (Proposition 19i) From the specification of A it follows that $\models A \to \exists \boldsymbol{p} F$ and thus $\models \exists \boldsymbol{p}\, (A \to F)$. Hence, by Proposition 19, the SP $(A \to F)[\boldsymbol{p}]$ has a solution. (Proposition 19ii) Let B be a formula such that the left side of holds. With Proposition 19 it follows that $\models B \to \exists \boldsymbol{p} F$. Hence $B \models \exists \boldsymbol{p} F$. Hence $B \models A$. □

The following example illustrates Proposition 21:

Example 22 (Elimination Result as Precondition for Solvability). Consider the SP $F[p_1 p_2]$ where

$$F = \forall x\,(p_1(x) \rightarrow p_2(x)) \;\wedge\; \forall x\,(a(x) \rightarrow p_2(x)) \;\wedge\; \forall x\,(p_2(x) \rightarrow b(x)).$$

Its formula is the consequent of the SP considered in Example 6. Since $\exists p_1 \exists p_2\, F \equiv \forall x\,(a(x) \rightarrow b(x)) \not\equiv \top$, from Proposition 19 it follows that $F[p_1 p_2]$ has no solution. If, however, the elimination result $\forall x\,(a(x) \rightarrow b(x))$ is added as an antecedent to F, then the resulting SP, which is the SP of Example 6, has a solution.

6 Conclusion

The *solution problem* and second-order quantifier *elimination* were interrelated tools in the early mathematical logic. Today elimination has entered automatization with applications in the computation of circumscription, in modal logics, and for semantic forgetting and modularizing knowledge bases, in particular for description logics. Since the solution problem on the basis of first-order logic is, like first-order validity, recursively enumerable there seems some hope to adapt techniques from first-order theorem proving.

The paper makes the relevant scenario accessible from the perspective of predicate logic and theorem proving. Together with the consideration of most general solutions in [30] it shows that a wealth of classical material on Boolean equation solving can be transferred to predicate logic. Some essential diverging points crystallize, like the constructability of witness formulas for quantified predicates. An abstracted version of the core property underlying the classical method of successive eliminations provides a foundation for systematizing and generalizing algorithms that reduce n-ary solution problems to unary solution problems.

Beyond the presented core framework there seem to be many results from different communities that are potentially relevant for further investigation. This includes the vast amount of techniques for equation solving on the basis of Boolean algebra and its variants, developed over the last 150 years. For description logics there are several results on concept unification, e.g., [3,4]. Variants of Craig interpolation such as disjunctive interpolation [25] share with the solution problem at least the objective to find substitution formulas such that the overall formula becomes valid (or, dually, unsatisfiable).

First steps towards constructive methods for the solution problem that apply to special cases of first-order inputs are described in [30], including methods based on Craig interpolation and on an elimination technique from [12]. The possible characterization of *solution* by an entailment also brings up the question whether Skolemization and Herbrand's theorem justify some "instance-based" technique for computing solutions that succeeds on large enough quantifier expansions.

Acknowledgments. The author thanks anonymous reviewers for their helpful comments. This work was supported by DFG grant WE 5641/1-1.

References

1. Ackermann, W.: Untersuchungen über das Eliminationsproblem der mathematischen Logik. Math. Ann. **110**, 390–413 (1935)
2. Baader, F.: On the complexity of Boolean unification. Inf. Process. Lett. **67**(4), 215–220 (1998)
3. Baader, F., Morawska, B.: Unification in the description logic $\mathcal{EL}$. Log. Methods Comput. Sci. **6**(3), 1–31 (2010)
4. Baader, F., Narendran, P.: Unification of concept terms in description logics. J. Symb. Comput. **31**, 277–305 (2001)
5. Behmann, H.: Das Auflösungsproblem in der Klassenlogik. Archiv für Philosophie **4**(1), 97–109 (1950). (First of two parts, also published in Archiv für mathematische Logik und Grundlagenforschung, 1.1 (1950), pp. 17–29)
6. Behmann, H.: Das Auflösungsproblem in der Klassenlogik. Archiv für Philosophie **4**(2), 193–211 (1951). (Second of two parts, also published in Archiv für mathematische Logik und Grundlagenforschung, 1.2 (1951), pp. 33–51)
7. Brown, F.M.: Boolean Reasoning, 2nd edn. Dover Publications, Mineola (2003)
8. Büttner, W., Simonis, H.: Embedding Boolean expressions into logic programming. J. Symb. Comput. **4**(2), 191–205 (1987)
9. Carlsson, M.: Boolean constraints in SICStus Prolog. Technical report SICS T91:09, Swedish Institute of Computer Science, Kista (1991)
10. Conradie, W., Goranko, V., Vakarelov, D.: Algorithmic correspondence and completeness in modal logic. I. The core algorithm SQEMA. LMCS **2**(1:5), 1–26 (2006)
11. Doherty, P., Łukaszewicz, W., Szałas, A.: Computing circumscription revisited: a reduction algorithm. J. Autom. Reason. **18**(3), 297–338 (1997)
12. Eberhard, S., Hetzl, S., Weller, D.: Boolean unification with predicates. J. Log. Comput. **27**(1), 109–128 (2017)
13. Gabbay, D.M., Schmidt, R.A., Szałas, A.: Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications. College Publications, London (2008)
14. Gabbay, D., Ohlbach, H.J.: Quantifier elimination in second-order predicate logic. In: KR 1992, pp. 425–435. Morgan Kaufmann (1992)
15. Kanellakis, P.C., Kuper, G.M., Revesz, P.Z.: Constraint query languages. In: PODS 1990, pp. 299–313. ACM Press (1990)
16. Kanellakis, P.C., Kuper, G.M., Revesz, P.Z.: Constraint query languages. J. Comput. Syst. Sci. **51**(1), 26–52 (1995)
17. Koopmann, P., Schmidt, R.A.: Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In: Fontaine, P., Ringeissen, C., Schmidt, R.A. (eds.) FroCoS 2013. LNCS, vol. 8152, pp. 87–102. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40885-4_7
18. Löwenheim, L.: Über das Auflösungsproblem im logischen Klassenkalkül. In: Sitzungsberichte der Berliner Mathematischen Gesellschaft, vol. 7, pp. 89–94. Teubner (1908)
19. Löwenheim, L.: Über die Auflösung von Gleichungen im logischen Gebietekalkül. Math. Ann. **68**, 169–207 (1910)
20. Martin, U., Nipkow, T.: Unification in Boolean rings. In: Siekmann, J.H. (ed.) CADE 1986. LNCS, vol. 230, pp. 506–513. Springer, Heidelberg (1986). doi:10.1007/3-540-16780-3_115
21. Martin, U., Nipkow, T.: Unification in Boolean rings. J. Autom. Reason. **4**(4), 381–396 (1988)

22. Martin, U., Nipkow, T.: Boolean unification - the story so far. J. Symb. Comput. **7**, 275–293 (1989)
23. Rudeanu, S.: Boolean Functions and Equations. Elsevier, Amsterdam (1974)
24. Rudeanu, S.: Lattice Functions and Equations. Springer, London (2001). doi:10.1007/978-1-4471-0241-0
25. Rümmer, P., Hojjat, H., Kuncak, V.: Disjunctive interpolants for Horn-clause verification. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 347–363. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39799-8_24
26. Schmidt, R.A.: The Ackermann approach for modal logic, correspondence theory and second-order reduction. J. Appl. Log. **10**(1), 52–74 (2012)
27. Schröder, E.: Vorlesungen über die Algebra der Logik. Teubner (vol. 1, 1890; vol. 2, pt. 1, 1891; vol. 2, pt. 2, 1905; vol. 3, 1895)
28. Seidl, M., Lonsing, F., Biere, A.: bf2epr: a tool for generating EPR formulas from QBF. In: PAAR-2012. EPiC, vol. 21, pp. 139–148 (2012)
29. Sofronie, V.: Formula-handling computer solution of Boolean equations. I. Ring equations. Bull. EATCS **37**, 181–186 (1989)
30. Wernhard, C.: The Boolean solution problem from the perspective of predicate logic - extended version. Technical report KRR 17-01, TU Dresden (2017)
31. Zhao, Y., Schmidt, R.A.: Concept forgetting in $\mathcal{ALCOI}$-ontologies using an Ackermann approach. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 587–602. Springer, Cham (2015). doi:10.1007/978-3-319-25007-6_34