Range-Restricted and Horn Interpolation through Clausal Tableaux

Christoph Wernhard(✉)

University of Potsdam, Potsdam, Germany
info@christophwernhard.com

Abstract. We show how variations of range-restriction and also the Horn property can be passed from inputs to outputs of Craig interpolation in first-order logic. The proof system is clausal tableaux, which stems from first-order ATP. Our results are induced by a restriction of the clausal tableau structure, which can be achieved in general by a proof transformation, also if the source proof is by resolution/paramodulation. Primarily addressed applications are query synthesis and reformulation with interpolation. Our methodical approach combines operations on proof structures with the immediate perspective of feasible implementation through incorporating highly optimized first-order provers.

1 Introduction

We show how variations of range-restriction and also the Horn property can be passed from inputs to outputs of Craig interpolation in first-order logic. The primarily envisaged application field is synthesis and reformulation of queries with interpolation [5,39,56]. Basically, the sought target query R is understood there as the right side of a definition of a given query Q within a given background knowledge base K, i.e., it holds that $K \models (Q \leftrightarrow R)$, where the vocabulary of R is in a given set of permitted target symbols. In first-order logic, the formulas R can be characterized as the Craig interpolants of $K \wedge Q$ and $\neg K' \vee Q'$, where K, Q are copies of K', Q' with the symbols not allowed in R replaced by fresh symbols [14]. Formulas R exist if and only if the entailment $K \wedge Q \models \neg K' \vee Q'$ holds. They can be constructed as Craig interpolants from given proofs of the entailment in a suitable calculus.

In databases and knowledge representation, syntactic fragments of first-order logic ensure desirable properties, for example domain independence. Typically, for given K and Q in some such fragment, also R must be in some specific fragment to be usable as a query or as a knowledge base component. Our work addresses this by showing for certain such fragments how membership is passed on to interpolants and thus to the constructed right sides of definitions. The

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 457292495. The work was supported by the North-German Supercomputing Alliance (HLRN).

R. Ramanayake and J. Urban (Eds.): TABLEAUX 2023, LNAI 14278, pp. 3–23, 2023.
https://doi.org/10.1007/978-3-031-43513-3_1

fragment in focus here is a variant of range-restriction from [59], known as a rather general syntactic condition to ensure domain independence [1, p. 97]. It permits conversion into a shape suitable for "evaluation" by binding free and quantified variables successively to the members of given predicate extensions. Correspondingly, if the vocabulary is relational, a range-restricted formula can be translated into a relational algebra expression. First-order representations of widely-used classes of integrity constraints, such as tuple-generating dependencies, are sentences that are range-restricted in the considered sense.

As proof system we use *clausal tableaux* [26,29–31,33], devised in the 1990s to take account of automated first-order provers that may be viewed as enumerating tree-shaped proof structures, labeled with instances of input clauses.[1] Such systems include the Prolog Technology Theorem Prover [53], SETHEO [32], leanCoP [42,43] and CMProver [16,45,60,61]. As shown in [62], a *given* closed clausal tableau is quite well-suited as a proof structure to extract a Craig interpolant. Via the translation of a resolution deduction tree [12] to a clausal tableau in cut normal form [31,62] this transfers also to interpolation from a given resolution/paramodulation proof.

Since the considered notion of range-restriction is based on prenexing and properties of both a CNF and a DNF representation of the formula, it fits well with the common first-order ATP setting involving Skolemization and clausification and the ATP-oriented interpolation on the basis of clausal tableaux, where in a first stage the propositional structure of the interpolant is constructed and in a second stage the quantifier prefix.

Our strengthenings of Craig interpolation are induced by a specific restriction of the clausal tableau structure, which we call *hyper*, since it relates to the proof structure restrictions of hyperresolution [46] and hypertableaux [2]. However, it is considered here for tree structures with rigid variables. A proof transformation that converts an arbitrary closed clausal tableau to one with the hyper property shows that the restriction is w.l.o.g. and, moreover, allows the prover unhampered search for the closed clausal tableaux or resolution/paramodulation proof underlying interpolation.

Structure of the Paper. Section 2 summarizes preliminaries, in particular interpolation with clausal tableaux [62]. Our main result on strengthenings of Craig interpolation for range-restricted formulas is developed in Sect. 3. Section 4 discusses Craig interpolation from a Horn formula, also combined with range-restriction. The proof transformation underlying these results is introduced in Sect. 5. We conclude in Sect. 6 with discussing related work, open issues and perspectives.

[1] Alternate accounts and views are provided by model elimination [34] and the connection method [7,8].

Proofs of nontrivial claims that are not proven in the body of the paper are supplemented in the preprint version [63]. An implementation with the PIE environment [60,61][2] is in progress.

2 Notation and Preliminaries

2.1 Notation

We consider *formulas* of first-order logic. An *NNF formula* is a quantifier-free formula built up from *literals* (atoms or negated atoms), truth-value constants $\top, \bot$, conjunction and disjunction. A *CNF formula*, also called *clausal formula*, is an NNF formula that is a conjunction of disjunctions (*clauses*) of literals. A *DNF formula* is an NNF formula that is a disjunction of conjunctions (*conjunctive clauses*) of literals. The complement of a literal L is denoted by $\overline{L}$. An occurrence of a subformula in a formula has positive (negative) *polarity*, depending on whether it is in the scope of an even (odd) number of possibly implicit occurrences of negation. Let F be a formula. $\mathcal{V}ar(F)$ is set of its free variables. $\mathcal{V}ar^+(F)$ ($\mathcal{V}ar^-(F)$) is the set of its free variables with an occurrence in an atom with positive (negative) polarity. $\mathcal{F}un(F)$ is the set of functions occurring in it, including constants, regarded here throughout as 0-ary functions. $\mathcal{P}red^{\pm}(F)$ is the set of pairs $\langle p, pol\rangle$, where p is a predicate and $pol \in \{+,-\}$, such that an atom with predicate p occurs in F with the polarity indicated by pol. $\mathcal{V}oc^{\pm}(F)$ is $\mathcal{F}un(F) \cup \mathcal{P}red^{\pm}(F)$. A *sentence* is a formula without free variables. An NNF is *ground* if it has no variables. If S is a set of terms, we call its members S-terms. The $\models$ symbol expresses semantic entailment.

2.2 Clausal First-Order Tableaux

A *clausal tableau* (briefly *tableau*) *for* a clausal formula F is a finite ordered tree whose nodes N with exception of the root are labeled with a literal $\mathsf{lit}(N)$, such that for each node N the disjunction of the literals of all its children in their left-to-right order, $\mathsf{clause}(N)$, is an instance of a clause in F. A branch of a tableau is *closed* iff it contains nodes with complementary literals. A node is *closed* iff all branches through it are closed. A tableau is *closed* iff its root is closed. A node is *closing* iff it has an ancestor with complementary literal. With a closing node N, a particular such ancestor is associated as *target of* N, written $\mathsf{tgt}(N)$. A tableau is *regular* iff no node has an ancestor with the same literal and is *leaf-closing* iff all closing nodes are leaves. A closed tableau that is leaf-closing is called *leaf-closed*. Tableau simplification can convert any tableau to a regular and leaf-closing tableau for the same clausal formula, closed iff the original tableau is so. Regularity is achieved by repeating the following operation [31, Sect. 2.1.3]: Select a node N with an ancestor that has the same literal, remove the edges originating in the parent of N and replace them with the edges originating in N. The leaf-closing property is achieved by repeatedly selecting an inner node

[2] http://cs.christophwernhard.com/pie.

N that is closing and removing the edges originating in N. All occurrences of variables in (the literal labels of) a tableau are free and their scope spans the whole tableau. That is, we consider *free-variable tableaux* [30, p. 158ff] with *rigid* variables [26, p. 114]. A tableau without variables is called *ground*. The universal closure of a clausal formula F is unsatisfiable iff there exists a closed clausal tableau for F. This holds also if *clausal tableau* is restricted by the properties *ground*, *regular* and *leaf-closing* in arbitrary combinations.

2.3 Interpolation with Clausal Tableaux

Craig's interpolation theorem [13,15] along with Lyndon's observation on the preservation of predicate polarities [35] ensures for first-order logic the existence of *Craig-Lyndon interpolants*, defined as follows. Let F, G be formulas such that $F \models G$. A *Craig-Lyndon interpolant* of F and G is a formula H such that (1) $F \models H$ and $H \models G$. (2) $\mathcal{V}oc^{\pm}(H) \subseteq \mathcal{V}oc^{\pm}(F) \cap \mathcal{V}oc^{\pm}(G)$. (3) $\mathcal{V}ar(H) \subseteq \mathcal{V}ar(F) \cap \mathcal{V}ar(G)$. The perspective of validating an entailment $F \models G$ by showing unsatisfiability of $F \wedge \neg G$ is reflected in the notion of *reverse Craig-Lyndon interpolant* of F and G, defined as Craig-Lyndon interpolant of F and $\neg G$.

Following [62], our interpolant construction is based on a generalization of clausal tableaux where nodes have an additional *side* label that is shared by siblings and indicates whether the tableau clause is an instance of an input clause derived from the formula F or of the formula G of the statement $F \wedge G \models \bot$ underlying the reverse interpolant. Thus, a *two-sided clausal tableau for* clausal formulas F *and* G is a tableau for $F \wedge G$ whose nodes N with exception of the root are labeled additionally with a *side* $\mathsf{side}(N) \in \{\mathsf{F}, \mathsf{G}\}$, such that (1) if N and N' are siblings, then $\mathsf{side}(N) = \mathsf{side}(N')$; (2) if N has a child N' with $\mathsf{side}(N') = \mathsf{F}$, then $\mathsf{clause}(N)$ is an instance of a clause in F, and if N has a child N' with $\mathsf{side}(N') = \mathsf{G}$, then $\mathsf{clause}(N)$ is an instance of a clause in G. We also refer to the side of the children of a node N as *side of* $\mathsf{clause}(N)$. For *side* $\in \{\mathsf{F}, \mathsf{G}\}$ define $\mathsf{path}_{side}(N) \stackrel{\text{def}}{=} \bigwedge_{N' \in Path \text{ and } \mathsf{side}(N') = side} \mathsf{lit}(N')$, where *Path* is the union of the set of the ancestors of N and $\{N\}$.

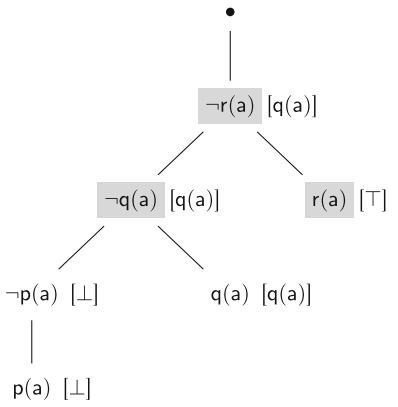

Fig. 1. A two-sided clausal tableau.

Let N be a node of a leaf-closed two-sided clausal tableau. The value of $\mathsf{ipol}(N)$ is an NNF formula, defined inductively as specified with the tables below, the left for the base case where N is a leaf, the right for the case where N is an inner node with children $N_1, \ldots, N_n$.

$\mathsf{side}(N)$	$\mathsf{side}(\mathsf{tgt}(N))$	$\mathsf{ipol}(N)$
F	F	$\bot$
F	G	$\mathsf{lit}(N)$
G	F	$\overline{\mathsf{lit}(N)}$
G	G	$\top$

$\mathsf{side}(N_1)$	$\mathsf{ipol}(N)$
F	$\bigvee_{i=1}^{n} \mathsf{ipol}(N_i)$
G	$\bigwedge_{i=1}^{n} \mathsf{ipol}(N_i)$

Example 1. Figure 1 shows a two-sided tableau for $F = \mathsf{p}(\mathsf{a}) \wedge (\neg\mathsf{p}(\mathsf{a}) \vee \mathsf{q}(\mathsf{a}))$ and $G = (\neg\mathsf{q}(\mathsf{a}) \vee \mathsf{r}(\mathsf{a})) \wedge \neg\mathsf{r}(\mathsf{a})$. Side G is indicated by gray background. For each node the value of ipol, after truth-value simplification, is annotated in brackets. The clauses of the tableau are $\neg\mathsf{r}(\mathsf{a})$ and $\neg\mathsf{q}(\mathsf{a}) \vee \mathsf{r}(\mathsf{a})$, which have side G, and $\neg\mathsf{p}(\mathsf{a}) \vee \mathsf{q}(\mathsf{a})$ and $\mathsf{p}(\mathsf{a})$, which have side F. If N is the node shown bottom left, labeled with $\mathsf{p}(\mathsf{a})$, then $\mathsf{path}_{\mathsf{F}}(N) = \neg\mathsf{p}(\mathsf{a}) \wedge \mathsf{p}(\mathsf{a})$ and $\mathsf{path}_{\mathsf{G}}(N) = \neg\mathsf{r}(\mathsf{a}) \wedge \neg\mathsf{q}(\mathsf{a})$.

If N_0 is the root of a two-sided tableaux for clausal *ground* formulas F and G, then $\mathsf{ipol}(N_0)$ is a Craig-Lyndon interpolant of F and $\neg G$.[3] The CTIF (*Clausal Tableau Interpolation for First-Order Formulas*) procedure (Fig. 2) [62] extends this to a two-stage [9,24] (inductive construction and lifting) interpolation method for full first-order logic. It is complete (yields a Craig-Lyndon interpolant for all first order formulas F and G such that $F \models G$) under the assumption that the method for tableau computation in Step 3 is complete (yields a closed tableau for all unsatisfiable clausal formulas). Some steps leave room for interpolation-specific heuristics: In step 4 the choice of the terms used for grounding; in step 5 the choice of the side assigned to clauses that are an instance of both a clause in F' and a clause in G'; and in step 7 the quantifier prefix, which is constrained just by a partial order.

Example 2. Let $F \stackrel{\text{def}}{=} \forall x\, \mathsf{p}(x) \wedge \forall x\, (\neg\mathsf{p}(x) \vee \mathsf{q}(x))$ and let $G \stackrel{\text{def}}{=} \forall x\, (\neg\mathsf{q}(x) \vee \mathsf{r}(x)) \rightarrow \mathsf{r}(\mathsf{a})$. Clausifying F and $\neg G$ then yields $F' = \mathsf{p}(x) \wedge \neg\mathsf{p}(x) \vee \mathsf{q}(x)$ and $G' = \neg\mathsf{q}(x) \vee \mathsf{r}(x) \wedge \neg\mathsf{r}(\mathsf{a})$. The tableau from Fig. 1 is a leaf-closed ground tableau for F' and G' and we obtain $\mathsf{q}(\mathsf{a})$ as H_{GRD}. Lifting for $\mathcal{F} = \{\}$ and $\mathcal{G} = \{\mathsf{a}\}$ yields the interpolant $H = \forall v_1\, \mathsf{q}(v_1)$.

Example 3. Let $F \stackrel{\text{def}}{=} \forall x \forall y\, \mathsf{p}(x, \mathsf{f}(x), y)$ and let $G \stackrel{\text{def}}{=} \exists x \mathsf{p}(\mathsf{a}, x, \mathsf{g}(x))$. Clausifying yields $F' = \mathsf{p}(x, \mathsf{f}(x), y)$ and $G' = \neg\mathsf{p}(\mathsf{a}, z, \mathsf{g}(z))$. We obtain $\mathsf{p}(\mathsf{a}, \mathsf{f}(\mathsf{a}), \mathsf{g}(\mathsf{f}(\mathsf{a})))$ as H_{GRD}. Lifting is for $\mathcal{F} = \{\mathsf{f}\}$ and $\mathcal{G} = \{\mathsf{a}, \mathsf{g}\}$ with $t_1 = \mathsf{a}$, $t_2 = \mathsf{f}(\mathsf{a})$ and $t_3 = \mathsf{g}(\mathsf{f}(\mathsf{a}))$. It yields $H = \forall v_1 \exists v_2 \forall v_3\, \mathsf{p}(v_1, v_2, v_3)$.

3 Interpolation and Range-Restriction

We now develop our main result on strengthenings of Craig interpolation for range-restricted formulas.

[3] So far, the interpolation method is a variation of well-known methods for sequent systems [52,55] and analytic tableaux [20] when restricted to propositional formulas.

3.1 CNF and DNF with Some Assumed Syntactic Properties

Following [59] we will consider a notion of range-restriction defined in terms of properties of two prenex formulas that are equivalent to the original formula, have both the same quantifier prefix but matrices in CNF and DNF, respectively.

INPUT: First-order formulas F and G such that $F \models G$.

METHOD:

1. *Free variables to placeholder constants.* Let F_c and G_c be the sentences obtained from F and G by replacing each free variable with a dedicated fresh constant.
2. *Skolemization and clausification.* Apply there conversion to prenex form and second-order Skolemization independently to F_c and to $\neg G_c$, resulting in disjoint sets of fresh Skolem functions $\mathcal{F}', \mathcal{G}'$, clausal formulas F', G', and sets $\mathcal{U}' = \mathcal{V}ar(F'), \mathcal{V}' = \mathcal{V}ar(G')$ of variables such that

   (a) $F_c \equiv \exists \mathcal{F}' \forall \mathcal{U}' F'$ and $\neg G_c \equiv \exists \mathcal{G}' \forall \mathcal{V}' G'$.
   (b) $\mathcal{V}oc^{\pm}(F') \subseteq \mathcal{V}oc^{\pm}(F_c) \cup \mathcal{F}'$ and $\mathcal{V}oc^{\pm}(\neg G') \subseteq \mathcal{V}oc^{\pm}(G_c) \cup \mathcal{G}'$.
   (c) $\forall \mathcal{U}' \forall \mathcal{V}' (F' \wedge G') \models \bot$.

   In case F' or G' contains the empty clause, exit with result $H \stackrel{\text{def}}{=} \bot$ or $H \stackrel{\text{def}}{=} \top$, respectively.
3. *Tableau computation.* Compute a leaf-closed clausal tableau for the clausal formula $F' \wedge G'$. This can be obtained, for example, from a clausal tableaux prover for clausal first-order formulas.
4. *Tableau grounding.* Instantiate all variables of the tableau with ground terms built up from functions in $F' \wedge G'$ and possibly also fresh functions $\mathcal{S} = \mathcal{S}_1 \uplus \mathcal{S}_2$. Observe that the grounded tableau is still a leaf-closed tableau for $F' \wedge G'$.
5. *Side assignment.* Convert the ground tableau to a two-sided tableau for F' and G' by attaching appropriate *side* labels to all nodes except the root. This is always possible because every clause of the tableau is an instance of a clause in F' or in G'.
6. *Ground interpolant extraction.* Let H_{GRD} be the value of $\mathsf{ipol}(N_0)$, where N_0 is the root of the tableau.
7. *Interpolant lifting.* Let $\mathcal{F} \stackrel{\text{def}}{=} \mathcal{F}' \cup (\mathcal{F}un(F) \setminus \mathcal{F}un(G)) \cup \mathcal{S}_1$ and let $\mathcal{G} \stackrel{\text{def}}{=} \mathcal{G}' \cup (\mathcal{F}un(G) \setminus \mathcal{F}un(F)) \cup \mathcal{S}_2$. Let $\mathcal{FG}$ stand for $\mathcal{F} \cup \mathcal{G}$. An *$\mathcal{FG}$-maximal occurrence* of an $\mathcal{FG}$-term in a formula is an occurrence that is not within another $\mathcal{FG}$-term. Let $\{t_1, \ldots, t_n\}$ be the set of the $\mathcal{FG}$-terms with an $\mathcal{FG}$-maximal occurrence in H_{GRD}, ordered such that if t_i is a subterm of t_j, then $i < j$. Let $\{v_1, \ldots, v_n\}$ be a set of fresh variables. For $i \in \{1, \ldots, n\}$ define the quantifiers Q_i as $\exists$ if $t_i \in \mathcal{F}$-terms and as $\forall$ if $t_i \in \mathcal{G}$-terms. Let

   $$H_c \stackrel{\text{def}}{=} Q_1 v_1 \ldots Q_n v_n \, H'_{\text{GRD}},$$

   where H'_{GRD} is obtained from H_{GRD} by replacing all $\mathcal{FG}$-maximal occurrences of terms t_i with variable v_i, simultaneously for all $i \in \{1, \ldots, n\}$.
8. *Placeholder constants to free variables.* Let H be H_c after replacing any constants that were introduced in step 1 with their corresponding variables.

OUTPUT: Return H, a Craig-Lyndon interpolant of the input formulas F and G.

Fig. 2. The CTIF Procedure for Craig-Lyndon Interpolation [62].

Although not syntactically unique, we refer to them functionally as $\mathsf{cnf}(F)$ and $\mathsf{dnf}(F)$ since we only rely on specific – easy to achieve – syntactic properties that are stated in the following Proposition 4–6.

Proposition 4. *For all formulas F it holds that $\mathcal{V}ar(\mathsf{cnf}(F)) \subseteq \mathcal{V}ar(F)$; $\mathcal{V}oc^{\pm}(\mathsf{cnf}(F)) \subseteq \mathcal{V}oc^{\pm}(F)$; $\mathcal{V}ar(\mathsf{dnf}(F)) \subseteq \mathcal{V}ar(F)$; $\mathcal{V}oc^{\pm}(\mathsf{dnf}(F)) \subseteq \mathcal{V}oc^{\pm}(F)$.*

For prenex formulas F with an NNF matrix let $\mathsf{dual}(F)$ be the formula obtained from F by switching quantifiers $\forall$ and $\exists$, connectives $\wedge$ and $\vee$, truth-value constants $\top$ and $\bot$, and literals with their complement.

Proposition 5. *For all formulas F it holds that $\mathsf{cnf}(F) = \mathsf{dual}(\mathsf{dnf}(\neg F))$; $\mathsf{dnf}(F) = \mathsf{dual}(\mathsf{cnf}(\neg F))$; $\mathsf{cnf}(\neg F) = \mathsf{dual}(\mathsf{dnf}(F))$; $\mathsf{dnf}(\neg F) = \mathsf{dual}(\mathsf{cnf}(F))$.*

Proposition 6. *Let $F_1, F_2, \ldots, F_n$ be NNF formulas. Then* (i) *Each clause in $\mathsf{cnf}(\bigwedge_{i=1}^{n} F_i)$ is in some $\mathsf{cnf}(F_j)$.* (ii) *Each conjunctive clause in $\mathsf{dnf}(\bigvee_{i=1}^{n} F_i)$ is in some $\mathsf{dnf}(F_j)$.* (iii) *Formulas F_j that are literals are in each clause in $\mathsf{cnf}(\bigvee_{i=1}^{n} F_i)$.* (iv) *Formulas F_j that are literals are in each conjunctive clause in $\mathsf{dnf}(\bigwedge_{i=1}^{n} F_i)$.* (v) *If S is a set of variables such that for all $i \in \{1, \ldots, n\}$ and clauses C in $\mathsf{cnf}(F_i)$ it holds that $\mathcal{V}ar(C) \cap S \subseteq \mathcal{V}ar^{-}(C)$, then for all clauses C in $\mathsf{cnf}(\bigvee_{i=1}^{n} F_i)$ it holds that $\mathcal{V}ar(C) \cap S \subseteq \mathcal{V}ar^{-}(C)$.* (vi) *If S is a set of variables such that for all $i \in \{1, \ldots, n\}$ and conjunctive clauses D in $\mathsf{dnf}(F_i)$ it holds that $\mathcal{V}ar(D) \cap S \subseteq \mathcal{V}ar^{+}(D)$, then for all conjunctive clauses D in $\mathsf{dnf}(\bigwedge_{i=1}^{n} F_i)$ it holds that $\mathcal{V}ar(D) \cap S \subseteq \mathcal{V}ar^{+}(D)$.*

3.2 Used Notions of Range-Restriction

The following definition renders the characteristics of the range-restricted formulas as considered by Van Gelder and Topor in [59, Theorem 7.2] (except for the special consideration of equality in [59]).

Definition 7. A formula F with free variables $\mathcal{X}$ is called *VGT-range-restricted* if $\mathsf{cnf}(F) = Q\, M_{\mathsf{C}}$ and $\mathsf{dnf}(F) = Q\, M_{\mathsf{D}}$, where Q is a quantifier prefix (the same in both formulas) upon universally quantified variables $\mathcal{U}$ and existentially quantified variables $\mathcal{E}$ (in arbitrary order), and M_{C}, M_{D} are quantifier-free formulas in CNF and DNF, respectively, such that

1. For all clauses C in M_{C} it holds that $\mathcal{V}ar(C) \cap \mathcal{U} \subseteq \mathcal{V}ar^{-}(C)$.
2. For all conjunctive clauses D in M_{D} it holds that $\mathcal{V}ar(D) \cap \mathcal{E} \subseteq \mathcal{V}ar^{+}(D)$.
3. For all conjunctive clauses D in M_{D} it holds that $\mathcal{X} \subseteq \mathcal{V}ar^{+}(D)$.

For VGT-range-restricted formulas it is shown in [59] that these can be translated via two intermediate formula classes to a relational algebra expression. Related earlier results include [17,18,40,41]. The constraint on universal variables is also useful on its own as a weaker variation of range-restriction, defined as follows.

Definition 8. A formula F is called *U-range-restricted* if $\mathsf{cnf}(F) = Q\, M_{\mathsf{C}}$ where Q is a quantifier prefix upon of the universally quantified variables $\mathcal{U}$ (there may also be existentially quantified variables in Q) and M_{C} is a quantifier-free formula in CNF such that for all clauses C in M_{C} it holds that $\mathcal{V}ar(C) \cap \mathcal{U} \subseteq \mathcal{V}ar^{-}(C)$.

For formulas without free variables, U-range-restriction and VGT-range-restriction are related as follows.

Proposition 9. *Let F be a sentence. Then* (i) *F is VGT-range-restricted iff F and $\neg F$ are both U-range-restricted.* (ii) *If F is universal (i.e., in prenex form with only universal quantifiers), then F is VGT-range-restricted iff F is U-range-restricted.* (iii) *If F is existential (i.e., in prenex form with only existential quantifiers), then F is VGT-range-restricted iff $\neg F$ is U-range-restricted.*

U-range-restriction covers well-known restrictions of knowledge bases and inputs of bottom-up calculi for first-order logic and fragments of it that are naturally represented by clausal formulas [3]. First-order representations of tuple-generating dependencies (TGDs) are VGT-range-restricted sentences: conjunctions of sentences of the form $\forall \mathcal{X}\mathcal{Y}\,(A(\mathcal{X}\mathcal{Y}) \rightarrow \exists \mathcal{Z}\, B(\mathcal{Y}\mathcal{Z}))$, where A is a possibly empty conjunction of relational atoms, B is a nonempty conjunction of relational atoms and the free variables of A and B are exactly those in the sequences $\mathcal{X}\mathcal{Y}$ and $\mathcal{Y}\mathcal{Z}$, respectively. Also certain generalizations, e.g., to disjunctive TGDs, where B is built up from atoms, $\wedge$ and $\vee$, are VGT-range-restricted.

3.3 Results on Range-Restricted Interpolation

The following theorem shows three variations for obtaining range-restricted interpolants from range-restricted inputs.

Theorem 10 (Interpolation and Range-Restriction). *Let F and G be formulas such that $F \models G$.*

(i) *If F is U-range-restricted, then there exists a U-range-restricted Craig-Lyndon interpolant H of F and G. Moreover, H can be effectively constructed from a clausal tableau proof of $F \models G$.*

(ii) *If F and G are sentences such that F and $\neg G$ are U-range-restricted, then there exists a VGT-range-restricted Craig-Lyndon interpolant H of F and G. Moreover, H can be effectively constructed from a clausal tableau proof of $F \models G$.*

(iii) *If F and $\neg G$ are U-range-restricted, $\mathcal{V}ar(F) = \mathcal{V}ar(G) = \mathcal{X}$, and (1) no clause in $\mathsf{cnf}(F)$ has only negative literals; (2) for all clauses C in $\mathsf{cnf}(\neg G)$ with only negative literals it holds that $\mathcal{X} \subseteq \mathcal{V}ar^{-}(C)$; (3) for all clauses C in $\mathsf{cnf}(\neg G)$ it holds that $\mathcal{V}ar(C) \cap \mathcal{X} \subseteq \mathcal{V}ar^{-}(C)$, then there exists a VGT-range-restricted Craig-Lyndon interpolant H of F and G. Moreover, H can be effectively constructed from a clausal tableau proof of $F \models G$.*

Observe that Theorem 10.i requires range-restriction only for F, the first of the two interpolation arguments. Theorem 10.iii aims at applications for query reformulation that in a basic form are expressed as interpolation task for input formulas $F = K \wedge Q(\mathcal{X})$ and $G = \neg K' \vee Q'(\mathcal{X})$. Here K expresses background knowledge and constraints as a U-range-restricted sentence and $Q(\mathcal{X})$ represents a query to be reformulated, with free variables $\mathcal{X}$. Formulas K' and Q' are copies

of K and Q, respectively, where predicates not allowed in the interpolant are replaced by primed versions. If the query Q is Boolean, i.e., $\mathcal{X}$ is empty, and Q is VGT-range-restricted, then Theorem 10.ii already suffices to justify the construction of a VGT-range-restricted interpolant. If $\mathcal{X}$ is not empty, the fine-print preconditions of Theorem 10.iii come into play. Precondition (1) requires that $\mathsf{cnf}(K)$ does not have a clause with only negative literals, which is satisfied if K represents TGDs. Also $\mathsf{cnf}(Q)$ is not allowed to have a clause with only negative literals. By precondition (2) all the free variables $\mathcal{X}$ must occur in all those clauses of $\mathsf{cnf}(\neg Q)$ that only have negative literals, which follows if Q meets condition (3.) of the VGT-range-restriction (Definition 7). By precondition (3) for all clauses C in $\mathsf{cnf}(\neg Q)$ it must hold that $\mathcal{V}ar(C) \cap \mathcal{X} \subseteq \mathcal{V}ar^{-}(C)$. A sufficient condition for Q to meet all these preconditions is that $\mathsf{dnf}(Q)$ has a purely existential quantifier prefix and a matrix with only positive literals where each query variable, i.e., member of $\mathcal{X}$, occurs in each conjunctive clause.

3.4 Proving Range-Restricted Interpolation – The Hyper Property

We will prove Theorem 10 by showing how the claimed interpolants can be obtained with CTIF. As a preparatory step we match items from the specification of CTIF (Fig. 2) with the constraints of range-restriction. The following notion gathers intermediate formulas and sets of symbols of CTIF.

Definition 11. An *interpolation context* is a tuple $\langle F, G, F', G', \mathcal{F}, \mathcal{G}, \mathcal{E}, \mathcal{U}, \mathcal{C}, V\rangle$, where F, G are formulas, F', G' are clausal formulas, $\mathcal{C}$ is a set of constants, $\mathcal{F}, \mathcal{G}$ are sets of functions, and $\mathcal{E}, \mathcal{U}, \mathcal{V}$ are sets of terms such that the following holds. (i) $F \models G$. (ii) Let F_c and G_c be F and G after replacing each free variable with a dedicated fresh constant. Let $\mathcal{C}$ be those constants that were used there to replace a variable that occurs in both F and G. F' and G' are the matrices of $\mathsf{cnf}(F_c)$ and of $\mathsf{cnf}(\neg G_c)$, after replacing existentially quantified variables with Skolem terms. (iii) $\mathcal{F}$ is the union of the set of the Skolem functions introduced for existential quantifiers of $\mathsf{cnf}(F_c)$, the set of functions occurring in F_c but not in G_c and, possibly, further functions freshly introduced in the grounding step of CTIF. Analogously, $\mathcal{G}$ is the union of the set of the Skolem functions introduced for $\mathsf{cnf}(\neg G_c)$, the set of functions occurring in G_c but not in F_c, and, possibly, further functions introduced in grounding. (iv) $\mathcal{E}$ and $\mathcal{U}$ are the sets of all terms with outermost function symbol in $\mathcal{F}$ and $\mathcal{G}$, respectively. (v) $\mathcal{V}$ is $\mathcal{E} \cup \mathcal{U} \cup \mathcal{C}$.

The following statements about an interpolation context are easy to infer.

Lemma 12. *Let $\langle F, G, F', G', \mathcal{F}, \mathcal{G}, \mathcal{E}, \mathcal{U}, \mathcal{C}, V\rangle$ be an interpolation context. Then* (i) *No member of $\mathcal{G}$ occurs in F'.* (ii) *No member of $\mathcal{F}$ occurs in G'.* (iii) *If F is U-range-restricted, then for all clauses C in F' it holds that if a variable occurs in C in a position that is not within an $\mathcal{E}$-term it occurs in C in a negative literal, in a position that is not within an $\mathcal{E}$-term.* (iv) *If $\neg G$ is U-range-restricted, then for all clauses C in G' it holds that if a variable occurs in C in a position that is not within an $\mathcal{U}$-term, it occurs in C in a negative literal, in a position that is*

not within an $\mathcal{U}$-term. (v) *If G satisfies condition (3) of Theorem 10.iii, then for all clauses C in G' it holds that any member of $\mathcal{C}$ that occurs in C in a position that is not within an $\mathcal{U}$-term occurs in C in a negative literal in a position that is not within an $\mathcal{U}$-term.*

CTIF involves conversion of terms to variables at lifting (step 7) and at replacing placeholder constants (step 8). We introduce a notation to identify those terms that will be converted there to variables. It mimics the notation for the set of free variables of a formula but applies to a set of terms, those with occurrences that are "maximal" with respect to a given set S of terms, i.e., are not within another term from S. For NNF formulas F define $S\text{-}\mathcal{M}ax(F)$ as the set of S-terms that occur in F in a position other than as subterm of another S-term. Define $S\text{-}\mathcal{M}ax^+(F)$ ($S\text{-}\mathcal{M}ax^-(F)$, respectively) as the set of S-terms that occur in F in a positive (negative, respectively) literal in a position other than as subterm of another S-term. We can now conclude from Lemma 12 the following properties of instances of clauses used for interpolant construction.

Lemma 13. *Let $\langle F, G, F', G', \mathcal{F}, \mathcal{G}, \mathcal{E}, \mathcal{U}, \mathcal{C}, V\rangle$ be an interpolation context. Then*

(i) *If F is U-range-restricted, then for all instances C of a clause in F' it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{U} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^-(C)$.*
(ii) *If $\neg G$ is U-range-restricted, then for all instances C of a clause in G' it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{E} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^-(C)$.*
(iii) *If condition (1) of Theorem 10.iii holds, then no instance C of a clause in F' has only negative literals.*
(iv) *If condition (2) of Theorem 10.iii holds, then for all instances C of a clause in G' with only negative literals it holds that $\mathcal{C} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^-(C)$.*
(v) *If $\neg G$ is U-range-restricted and condition (3) of Theorem 10.iii holds, then for all instances C of a clause in G' it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{C} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^-(C)$.*

The following proposition adapts Props. 6.v and 6.vi to $S\text{-}\mathcal{M}ax$.

Proposition 14. *Let $F_1, F_2, \ldots, F_n$ be NNF formulas and let T be a set of terms. Then* (i) *If S is a set of terms such that for all $i \in \{1, \ldots, n\}$ and clauses C in $\mathsf{cnf}(F_i)$ it holds that $T\text{-}\mathcal{M}ax(C) \cap S \subseteq T\text{-}\mathcal{M}ax^-(C)$, then for all clauses C in $\mathsf{cnf}(\bigvee_{i=1}^n F_i)$ it holds that $T\text{-}\mathcal{M}ax(C) \cap S \subseteq T\text{-}\mathcal{M}ax^-(C)$.* (ii) *If S is a set of terms such that for all $i \in \{1, \ldots, n\}$ and conjunctive clauses D in $\mathsf{dnf}(F_i)$ it holds that $T\text{-}\mathcal{M}ax(D) \cap S \subseteq T\text{-}\mathcal{M}ax^+(D)$, then for all conjunctive clauses D in $\mathsf{dnf}(\bigwedge_{i=1}^n F_i)$ it holds that $T\text{-}\mathcal{M}ax(D) \cap S \subseteq T\text{-}\mathcal{M}ax^+(D)$.*

The key to obtain range-restricted interpolants from CTIF is that the tableau must have a specific form, which we call *hyper*, as it resembles proofs by hyper-resolution [46] and hypertableaux [2].

Definition 15. A clausal tableau is called *hyper* if the nodes labeled with a negative literal are exactly the leaf nodes.

While hyperresolution and related approaches, e.g., [2,3,11,36,46], consider DAG-shaped proofs with non-rigid variables, aiming at interpolant extraction we consider the hyper property for tree-shaped proofs with rigid variables. The *hyper* requirement is w.l.o.g. because arbitrary closed clausal tableaux can be converted to tableaux with the hyper property, as we will see in Sect. 5.

The proof of Theorem 10 is based on three properties that invariantly hold for all nodes, or for all inner nodes, respectively, stated in the following lemma.

Lemma 16. *Let* $\langle F, G, F', G', \mathcal{F}, \mathcal{G}, \mathcal{E}, \mathcal{U}, \mathcal{C}, V\rangle$ *be an interpolation context and assume a leaf-closed and hyper two-sided clausal ground tableau for* F' *and* G'*.*

(i) *If* F *is U-range-restricted, then for all nodes* N *the property* $\mathsf{INV_C}(N)$ *defined as follows holds:* $\mathsf{INV_C}(N) \stackrel{\text{def}}{=}$ *For all clauses* C *in* $\mathsf{cnf}(\mathsf{ipol}(N))$ *it holds that* $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{U} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{-}(C) \cup \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path_F}(N))$.

(ii) *If* $\neg G$ *is U-range-restricted, then for all nodes* N *the property* $\mathsf{INV_D}(N)$ *defined as follows holds:* $\mathsf{INV_D}(N) \stackrel{\text{def}}{=}$ *For all conjunctive clauses* D *in* $\mathsf{dnf}(\mathsf{ipol}(N))$ *it holds that* $\mathcal{V}\text{-}\mathcal{M}ax(D) \cap \mathcal{E} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{+}(D) \cup \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path_G}(N))$.

(iii) *If* $\neg G$ *is U-range-restricted and conditions (1)–(3) Theorem 10.iii hold, then for all inner nodes* N *the property* $\mathsf{INV_X}(N)$ *defined as follows holds:* $\mathsf{INV_X}(N) \stackrel{\text{def}}{=}$ *For all conjunctive clauses* D *in* $\mathsf{dnf}(\mathsf{ipol}(N))$ *it holds that* $\mathcal{C} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{+}(D) \cup \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path_G}(N))$.

Each of Lemma 16.i, 16.ii and 16.iii can be proven independently by an induction on the tableau structure, but for the same tableau, such that the properties claimed by them can be combined. In proving these three sub-lemmas it is sufficient to use their respective preconditions only to justify the application of matching sub-lemmas of Lemma 13. That lemma might thus be seen as an abstract interface that delivers everything that depends on these preconditions and is relevant for Theorem 10.

We show here the proof of Lemma 16.i. Lemma 16.ii can be proven in full analogy. The proof of Lemma 16.iii is deferred to [63, App. A]. In general, recall that the tableau in Lemma 16 is a two-sided tableau for F' and G' that is leaf-closed and hyper. Hence literal labels of leaves are negative, while those of inner nodes are positive. All tableau clauses are ground and with an associated *side* in $\{\mathsf{F}, \mathsf{G}\}$ such that a tableau clause with side F is an instance of a clause in F' and one with side G is an instance of a clause in G'.

Proof (Lemma 16.i). By induction on the tableau structure.

Base case where N *is a leaf.* If N and $\mathsf{tgt}(N)$ have the same side, then $\mathsf{ipol}(N)$ is a truth value constant, hence $\mathcal{V}\text{-}\mathcal{M}ax(\mathsf{ipol}(N)) = \emptyset$, implying $\mathsf{INV_C}(N)$. If N has side F and $\mathsf{tgt}(N)$ has side G, then $\mathsf{ipol}(N) = \mathsf{lit}(N)$, which, because N is a leaf, is a negative literal. Thus $\mathcal{V}\text{-}\mathcal{M}ax(\mathsf{ipol}(N)) = \mathcal{V}\text{-}\mathcal{M}ax^{-}(\mathsf{ipol}(N))$, which implies $\mathsf{INV_C}(N)$. If N has side G and $\mathsf{tgt}(N)$ has side F, then $\mathsf{ipol}(N) = \mathsf{lit}(\mathsf{tgt}(N))$, which, because N is a leaf, is a positive literal. Thus $\mathcal{V}\text{-}\mathcal{M}ax(\mathsf{ipol}(N)) \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path_F}(N))$, implying $\mathsf{INV_C}(N)$.

Induction Step. Let $N_1, \ldots, N_n$, where $1 \leq n$, be the children of N. Assume as induction hypothesis that for $i \in \{1, \ldots, n\}$ it holds that $\mathsf{INV}_{\mathsf{C}}(N_i)$. Consider the case where the side of the children is F. Then

(1) $\mathsf{ipol}(N) = \bigvee_{i=1}^{n} \mathsf{ipol}(N_i)$.

Assume that $\mathsf{INV}_{\mathsf{C}}(N)$ does not hold. Then there exists a clause K in $\mathsf{cnf}(\mathsf{ipol}(N))$ and a term t such that (2) $t \in \mathcal{U}$; (3) $t \in \mathcal{V}\text{-}\mathcal{M}ax(K)$; (4) $t \notin \mathcal{V}\text{-}\mathcal{M}ax^{-}(K)$; (5) $t \notin \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path}_{\mathsf{F}}(N))$. To derive a contradiction, we first show that given (2), (4) and (5) it holds that

(6) For all children N' of N: $t \notin \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path}_{\mathsf{F}}(N'))$.

Statement (6) can be proven as follows. Assume to the contrary that there is a child N' of N such that $t \in \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path}_{\mathsf{F}}(N'))$. By (5) it follows that $t \in \mathcal{V}\text{-}\mathcal{M}ax(\mathsf{lit}(N'))$ and $\mathsf{lit}(N')$ is positive. By Lemma 13.i and (2) there is another child N'' of N such that $\mathsf{lit}(N'')$ is negative and $t \in \mathcal{V}\text{-}\mathcal{M}ax(\mathsf{lit}(N''))$. Since the tableau is closed, it follows from (5) that $\mathsf{tgt}(N'')$ has side G, which implies that $\mathsf{ipol}(N'') = \mathsf{lit}(N'')$. Hence $t \in \mathcal{V}\text{-}\mathcal{M}ax(\mathsf{ipol}(N''))$. Since $\mathsf{ipol}(N'')$ is a negative literal and a disjunct of $\mathsf{ipol}(N)$, it follows from (1) and Prop. 6.iii that for all clauses C in $\mathsf{cnf}(\mathsf{ipol}(N))$ it holds that $t \in \mathcal{V}\text{-}\mathcal{M}ax^{-}(C)$, contradicting assumption (4). Hence (6) must hold.

From (6), (2) and the induction hypothesis it follows that for all children N' of N and clauses C' in $\mathsf{cnf}(\mathsf{ipol}(N'))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C') \cap \{t\} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{-}(C')$. Hence, by (1) and Prop. 14.i it follows that for all clauses C in $\mathsf{cnf}(\mathsf{ipol}(N))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \{t\} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{-}(C)$. This, however, contradicts our assumption of the existence of a clause K in $\mathsf{cnf}(\mathsf{ipol}(N))$ that satisfies (3) and (4). Hence $\mathsf{INV}_{\mathsf{C}}(N)$ must hold.

We conclude the proof of the induction step for $\mathsf{INV}_{\mathsf{C}}(N)$ by considering the case where the side of the children of N is G. Then

(7) $\mathsf{ipol}(N) = \bigwedge_{i=1}^{n} \mathsf{ipol}(N_i)$.
(8) For all children N' of N: $\mathsf{path}_{\mathsf{F}}(N) = \mathsf{path}_{\mathsf{F}}(N')$.

$\mathsf{INV}_{\mathsf{C}}(N)$ follows from the induction hypothesis, (8), (7) and Prop. 6.i. □

The invariant properties of tableau nodes shown in Lemmas 16.i–16.iii apply in particular to the tableau root. We now apply this to prove Theorem 10.

Proof (Theorem 10). Interpolants with the stated properties are obtained with CTIF, assuming w.l.o.g. that the CNF computed in step 2 meets the requirement of Sect. 3.1, and that the closed clausal tableau computed in step 3 is leaf-closed and has the hyper property. That CTIF constructs a Craig-Lyndon interpolant has been shown in [62]. It remains to show the further claimed properties of the interpolant. Let $\langle F, G, F', G', \mathcal{F}, \mathcal{G}, \mathcal{E}, \mathcal{U}, \mathcal{C}, V \rangle$ be the interpolation context for the input formulas F and G and let N_0 be the root of the tableau computed in step 3. Since N_0 is the root, $\mathsf{path}_{\mathsf{F}}(N_0) = \mathsf{path}_{\mathsf{G}}(N_0) = \top$ and thus the expressions $\mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path}_{\mathsf{F}}(N_0))$ and $\mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path}_{\mathsf{G}}(N_0))$ in the specifications of $\mathsf{INV}_{\mathsf{C}}(N_0)$, $\mathsf{INV}_{\mathsf{D}}(N_0)$ and $\mathsf{INV}_{\mathsf{X}}(N_0)$ all denote the empty set. The claims made in the particular sub-theorems can then be shown as follows.

(10.i) By Lemma 16.i it follows that $\mathsf{INV}_{\mathsf{C}}(N_0)$. Hence, for all clauses C in $\mathsf{cnf}(\mathsf{ipol}(N_0))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{U} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{-}(C)$. It follows that the result of the interpolant lifting (step 7) of CTIF applied to $\mathsf{ipol}(N_0)$ is U-range-restricted. Placeholder constant replacement (step 8) does not alter this.

(10.ii) As for Theorem 10.i it follows that for all clauses C in $\mathsf{cnf}(\mathsf{ipol}(N_0))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{U} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{-}(C)$. By Lemma 16.ii it follows that $\mathsf{INV}_{\mathsf{D}}(N_0)$. Hence, for all conjunctive clauses D in $\mathsf{dnf}(\mathsf{ipol}(N_0))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(D) \cap \mathcal{E} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{+}(D)$. It follows that the result of the interpolant lifting of CTIF applied to $\mathsf{ipol}(N_0)$ is U-range-restricted. Since F and G have no free variables, placeholder constant replacement has no effect.

(10.iii) As for Theorem 10.ii it follows that for all clauses C in $\mathsf{cnf}(\mathsf{ipol}(N_0))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{U} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{-}(C)$ and for all conjunctive clauses D in $\mathsf{dnf}(\mathsf{ipol}(N_0))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(D) \cap \mathcal{E} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{+}(D)$. By Lemma 16.iii it follows that $\mathsf{INV}_{\mathsf{X}}(N_0)$. Hence, for all conjunctive clauses D in $\mathsf{dnf}(\mathsf{ipol}(N_0))$ it holds that $\mathcal{C} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{+}(D)$. It follows that the result of the interpolant lifting of CTIF applied to $\mathsf{ipol}(N_0)$ followed by placeholder constant replacement, now applied to $\mathcal{C}$, is VGT-range-restricted. □

4 Horn Interpolation

A *Horn clause* is a clause with at most one positive literal. A *Horn formula* is built up from Horn clauses with the connectives $\wedge$, $\exists$ and $\forall$. Horn formulas are important in countless theoretical and practical respects. Our interpolation method on the basis of clausal tableaux with the hyper property can be applied to obtain a Horn interpolant under the precondition that the first argument formula F of the interpolation problem is Horn. The following theorem makes this precise. It can be proven by an induction on the structure of a clausal tableau with the hyper property (see [63, App. B]).

Theorem 17 (Interpolation from a Horn Formula). *Let F be a Horn formula and let G be a formula such that $F \models G$. Then there exists a Craig-Lyndon interpolant H of F and G that is a Horn formula. Moreover, H can be effectively constructed from a clausal tableau proof of $F \models G$.*

An apparently weaker property than Theorem 17 has been shown in [38, § 4] with techniques from model theory: For *two* universal Horn formulas F and G there exists a universal Horn formula that is like a Craig interpolant, except that function symbols are not constrained. A *universal* Horn formula is there a prenex formula with only universal quantifiers and a Horn matrix. For CTIF, the corresponding strengthening of the interpolant to a universal formula can be read-off from the specification of interpolant lifting (step 7 in Fig. 2).

The following corollary shows that Theorem 17 can be combined with Theorem 10 to obtain interpolants that are both Horn and range-restricted.

Corollary 18 (Range-Restricted Horn Interpolants). *Theorems 10.i, 10.ii and 10.iii can be strengthened: If F is a Horn formula, then there exists*

a Craig-Lyndon interpolant H with the properties shown in the respective theorem and the additional property that it is Horn. Moreover, H can be effectively constructed from a clausal tableau proof of $F \models G$.

Proof. Can be shown by combining the proof of Theorem 10.i, 10.ii and 10.iii , respectively, with the proof of interpolation from a Horn sentence, Theorem 17. The combined proofs are based on inductions on the same closed tableau with the hyper property. □

5 Obtaining Proofs with the Hyper Property

Our new interpolation theorems, Theorems 10 and 17, depend on the hyper property of the underlying closed clausal tableaux from which interpolants are extracted. We present a proof transformation that converts any closed clausal tableau to one with the hyper property. The transformation can be applied to a clausal tableau as obtained directly from a clausal tableaux prover. Moreover, it can be also be indirectly applied to a resolution proof. To this end, the resolution deduction *tree* [12] of the binary resolution proof is first translated to a closed clausal ground tableau in *cut normal form* [31, Sect. 7.22]. There the inner clauses are atomic cuts, tautologies of the form $\neg p(t_1, \ldots, t_n) \vee p(t_1, \ldots, t_n)$ or $p(t_1, \ldots, t_n) \vee \neg p(t_1, \ldots, t_n)$, corresponding to literals upon which a (tree) resolution step has been performed. Clauses of nodes whose children are leaves are instances of input clauses. Our hyper conversion can then be applied to the tableau in cut normal form. It is easy to see that a regular leaf-closed tableau with the hyper property can not have atomic cuts. Hence the conversion might be viewed as an elimination method for these cuts.

We specify the hyper conversion in Fig. 3 as a procedure that destructively manipulates a tableau. A *fresh copy* of an ordered tree T is there an ordered tree T' with fresh nodes and edges, related to T through a bijection c such that any node N of T has the same labels (literal label and side label) as node $c(N)$ of T' and such that the i-th edge originating in node N of T ends in node M if and only if the i-th edge originating in node $c(N)$ of T' ends in node $c(M)$. The procedure is performed as an iteration that in each round chooses an inner node with negative literal label and then modifies the tableau. Hence, at termination there is no inner node with negative literal, which means that the tableau is hyper. Termination of the procedure can be shown with a measure that strictly decreases in each round (Prop. 20 in [63, App. C]). Figures 4 and 5 show example applications of the procedure.

Since the hyper conversion procedure copies parts of subtrees it is not a polynomial operation.[4] To get an idea of its practical feasibility, we experimented with an unbiased set of proofs of miscellaneous problems. For this we took those 112 *CASC-J11* [54] problems that could be proven with Prover9 [37] in 400 s per

[4] A thorough complexity analysis should take calculus- or strategy-dependent properties of the input proofs into account. And possibly also the blow-up from resolution to tree resolution underlying the cut normal form tableaux.

INPUT: A closed clausal tableau.

METHOD: Simplify the tableau to leaf-closing and regular form (Sect. 2.2). Repeat the following operations until the resulting tableau is hyper.

1. Let N' be the first node visited in pre-order with a child that is an inner node with a negative literal label. Let N be the leftmost such child.
2. Create a fresh copy U of the subtree rooted at N'. In U remove the edges that originate in the node corresponding to N.
3. Replace the edges originating in N' with the edges originating in N.
4. For each leaf descendant M of N' with $\mathsf{lit}(M) = \overline{\mathsf{lit}(N)}$: Create a fresh copy U' of U. Change the origin of the edges originating in the root of U' to M.
5. Simplify the tableau to leaf-closing and regular form (Sect. 2.2).

OUTPUT: A leaf-closed, regular and hyper clausal tableau whose clauses are clauses of the input tableau.

Fig. 3. The *hyper conversion* proof transformation procedure.

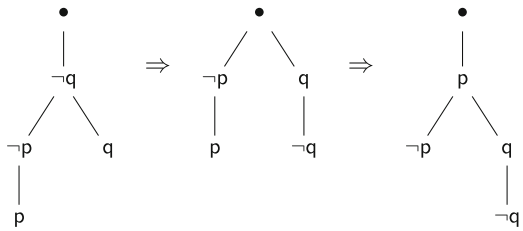

Fig. 4. Hyper conversion of a closed clausal tableau in two rounds.

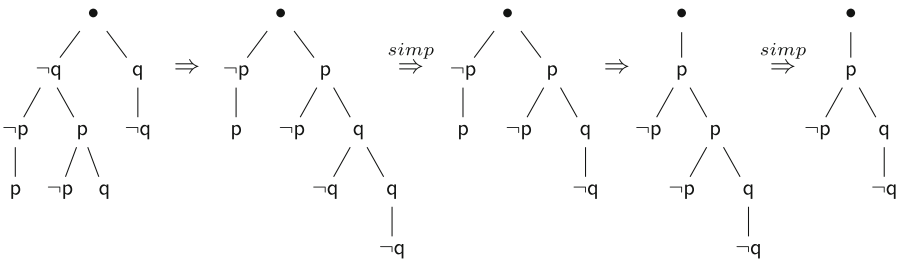

Fig. 5. Hyper conversion of a closed clausal tableau in cut normal form in two rounds. For each round the result after procedure steps 1–4 is shown and then the result after step 5, simplification, applied here to achieve regularity.

problem, including a basic proof conversion with Prover9's tool Prooftrans.[5] The hyper conversion succeeded on 107 (or 96%) of these, given 400 s timeout per proof, where the actual median of used time was only 0.01 s. It was applied to a tableau in cut normal form that represents the proof tree of Prover9's proof. The two intermediate steps, translation of paramodulation to binary resolution and expansion to cut normal form, succeeded in fractions of a second, except for one case where the expansion took 121 s and two cases where it failed due to memory exhaustion. The hyper conversion then failed in three further cases. For all except two proofs the hyper conversion reduced the proof size, where the overall median of the size ratio hyper-to-input was 0.39. See [63, App. D] for details.

6 Conclusion

We conclude with discussing related work, open issues and perspectives. Our interpolation method CTIF [62] is complete for first-order logic with function symbols. Vampire's native interpolation [22,23], targeted at verification, is like all local methods incomplete [28]. Princess [10,47] implements interpolation with a sequent calculus that supports theories for verification and permits uninterpreted predicates and functions. Suitable proofs for our approach can currently be obtained from CMProver (clausal tableaux) and Prover9 (resolution/paramodulation). With optimized settings, Vampire [27] and E [49] as of today only output proofs with gaps. This seems to improve [48] or might be overcome by re-proving with Prover9 using lemmas from the more powerful systems.

So far we did not address special handling of equality in the context of range-restriction, a topic on its own, e.g., [3,59]. We treat it as predicate, with axioms for reflexivity, symmetry, transitivity and substitutivity. CTIF works smoothly with these, respecting polarity constraints of equality in interpolants [62, Sect. 10.4]. With exception of reflexivity these axioms are U-range-restricted. We do not interfere with the provers' equality handling and just translate in finished proofs paramodulation into binary resolution with substitutivity axioms.

The potential bottleneck of conversion to clausal form in CTIF may be remedied with structure-preserving (aka *definitional*) normal forms [19,44,50,58].

Our *hyper* property might be of interest for proof presentation and exchange, since it gives the proof tree a constrained shape and in experiments often shortens it. Like hyperresolution and hypertableaux it can be generalized to take a "semantics" into account [51] [12, Chap. 6] [26, Sect. 4.5]. To shorten interpolants, it might be combined with proof reductions (e.g., [64]).

For query reformulation, interpolation on the basis of general first-order ATP was so far hardly considered. Most methods are sequent calculi [6,56] or analytic tableaux systems [5,21,25,57]. Experiments with ATP systems and propositional inputs indicate that requirements are quite different from those

[5] On a Linux notebook with 12th Gen Intel® Core™ i7-1260P CPU and 32 GB RAM.

in verification [4]. An implemented system [25,57] uses analytic tableaux with dedicated refinements for enumerating alternate proofs/interpolants corresponding to query plans for heuristic choice. In [5] the focus is on interpolants that are sentences respecting binding patterns, which, like range-restriction, ensures database evaluability. Our interpolation theorems show fine-grained conditions for passing variations of range-restriction and the Horn property on to interpolants. Matching these with the many formula classes considered in knowledge representation and databases is an issue for future work. A further open topic is adapting recent synthesis techniques for nested relations [6] to the clausal tableaux proof system.

Methodically, we exemplified a way to approach operations on proof structures while taking efficient automated first-order provers into account. Feasible implementations are brought within reach, for practical application and also for validating abstract claims and conjectures with scrutiny. The prover is a black box, given freedom on optimizations, strategy and even calculus. For interfacing, the overall setting incorporates clausification and Skolemization. Requirements on the proof structure do not hamper proof search, but are ensured by transformations applied to proofs returned by the efficient systems.

Acknowledgments. The author thanks Michael Benedikt for bringing the subtleties of range-restriction in databases to attention, Cécilia Pradic for insights into subtleties of proof theory, and anonymous reviewers for helpful suggestions to improve the presentation.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley, Boston (1995)
2. Baumgartner, P., Furbach, U., Niemelä, I.: Hyper tableaux. In: Alferes, J.J., Pereira, L.M., Orlowska, E. (eds.) JELIA 1996. LNCS, vol. 1126, pp. 1–17. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61630-6_1
3. Baumgartner, P., Schmidt, R.A.: Blocking and other enhancements for bottom-up model generation methods. J. Autom. Reasoning **64**, 197–251 (2020). https://doi.org/10.1007/11814771_11
4. Benedikt, M., Kostylev, E.V., Mogavero, F., Tsamoura, E.: Reformulating queries: theory and practice. In: Sierra, C. (ed.) IJCAI 2017, pp. 837–843. ijcai.org (2017). https://doi.org/10.24963/ijcai.2017/116
5. Benedikt, M., Leblay, J., ten Cate, B., Tsamoura, E.: Generating Plans from Proofs: The Interpolation-based Approach to Query Reformulation. Morgan & Claypool, San Rafael (2016). https://doi.org/10.1007/978-3-031-01856-5
6. Benedikt, M., Pradic, C., Wernhard, C.: Synthesizing nested relational queries from implicit specifications. In: PODS '23, pp. 33–45 (2023). https://doi.org/10.1145/3584372.3588653
7. Bibel, W.: Automated Theorem Proving, 2nd edn. Vieweg, Braunschweig (1987). https://doi.org/10.1007/978-3-322-90102-6. First edition 1982
8. Bibel, W., Otten, J.: From Schütte's formal systems to modern automated deduction. In: The Legacy of Kurt Schütte, pp. 217–251. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49424-7_13

9. Bonacina, M.P., Johansson, M.: On interpolation in automated theorem proving. J. Autom. Reasoning **54**(1), 69–97 (2014). https://doi.org/10.1007/s10817-014-9314-0
10. Brillout, A., Kroening, D., Rümmer, P., Wahl, T.: Beyond quantifier-free interpolation in extensions of Presburger arithmetic. In: Jhala, R., Schmidt, D. (eds.) VMCAI 2011. LNCS, vol. 6538, pp. 88–102. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-18275-4_8
11. Bry, F., Yahya, A.H.: Positive unit hyperresolution tableaux and their application to minimal model generation. J. Autom. Reasoning **25**(1), 35–82 (2000). https://doi.org/10.1023/A:1006291616338
12. Chang, C.L., Lee, R.C.T.: Symbolic Logic and Automated Theorem Proving. Academic Press, Cambridge (1973)
13. Craig, W.: Linear reasoning. A new form of the Herbrand-Gentzen theorem. J. Symb. Log. **22**(3), 250–268 (1957). https://doi.org/10.2307/2963593
14. Craig, W.: Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. J. Symb. Log. **22**(3), 269–285 (1957). https://doi.org/10.2307/2963594
15. Craig, W.: The road to two theorems of logic. Synthese **164**(3), 333–339 (2008). https://doi.org/10.1007/s11229-008-9353-3
16. Dahn, I., Wernhard, C.: First order proof problems extracted from an article in the Mizar mathematical library. In: Bonacina, M.P., Furbach, U. (eds.) FTP'97, pp. 58–62. RISC-Linz Report Series No. 97–50, Joh. Kepler Univ., Linz (1997). https://www.logic.at/ftp97/papers/dahn.pdf
17. Demolombe, R.: Syntactical characterization of a subset of domain independent formulas. Technical report, ONERA-CERT, Toulouse (1982)
18. Demolombe, R.: Syntactical characterization of a subset of domain independent formulas. JACM **39**, 71–94 (1992). https://doi.org/10.1145/147508.147520
19. Eder, E.: An implementation of a theorem prover based on the connection method. In: Bibel, W., Petkoff, B. (eds.) AIMSA'84, pp. 121–128. North-Holland (1985)
20. Fitting, M.: First-Order Logic and Automated Theorem Proving, 2nd edn. Springer, Cham (1995). https://doi.org/10.1007/978-1-4612-2360-3
21. Franconi, E., Kerhet, V., Ngo, N.: Exact query reformulation over databases with first-order and description logics ontologies. JAIR **48**, 885–922 (2013). https://doi.org/10.1613/jair.4058
22. Hoder, K., Holzer, A., Kovács, L., Voronkov, A.: Vinter: a Vampire-based tool for interpolation. In: Jhala, R., Igarashi, A. (eds.) APLAS 2012. LNCS, vol. 7705, pp. 148–156. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35182-2_11
23. Hoder, K., Kovács, L., Voronkov, A.: Interpolation and symbol elimination in Vampire. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS (LNAI), vol. 6173, pp. 188–195. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14203-1_16
24. Huang, G.: Constructing Craig interpolation formulas. In: Du, D.-Z., Li, M. (eds.) COCOON 1995. LNCS, vol. 959, pp. 181–190. Springer, Heidelberg (1995). https://doi.org/10.1007/BFb0030832
25. Hudek, A., Toman, D., Weddell, G.: On enumerating query plans using analytic tableau. In: De Nivelle, H. (ed.) TABLEAUX 2015. LNCS (LNAI), vol. 9323, pp. 339–354. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24312-2_23
26. Hähnle, R.: Tableaux and related methods. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 1, chap. 3, pp. 101–178. Elsevier (2001). https://doi.org/10.1016/b978-044450813-3/50005-9

27. Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 1–35. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_1
28. Kovács, L., Voronkov, A.: First-order interpolation and interpolating proof systems. In: Eiter, T., Sands, D. (eds.) LPAR-21. EPiC, vol. 46, pp. 49–64. EasyChair (2017). https://doi.org/10.29007/1qb8
29. Letz, R.: Clausal tableaux. In: Bibel, W., Schmitt, P.H. (eds.) Automated Deduction - A Basis for Applications, vol. I, pp. 43–72. Kluwer Academic Publishers (1998)
30. Letz, R.: First-order tableau methods. In: D'Agostino, A., Gabbay, D.M., Hähnle, R., Posegga, J. (eds.) Handbook of Tableau Methods, pp. 125–196. Springer, Dordrecht (1999)
31. Letz, R.: Tableau and Connection Calculi. Structure, Complexity, Implementation. Habilitationsschrift, TU München (1999). http://www2.tcs.ifi.lmu.de/~letz/habil.pdf. Accessed 19 July 2023
32. Letz, R., Schumann, J., Bayerl, S., Bibel, W.: SETHEO: a high-performance theorem prover. J. Autom. Reasoning **8**(2), 183–212 (1992). https://doi.org/10.1007/BF00244282
33. Letz, R., Stenz, G.: Model elimination and connection tableau procedures. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 1, pp. 2015–2114. Elsevier (2001)
34. Loveland, D.W.: Automated Theorem Proving: A Logical Basis. North-Holland, Amsterdam (1978)
35. Lyndon, R.: An interpolation theorem in the predicate calculus. Pac. J. Math. **9**, 129–142 (1959). https://doi.org/10.2140/pjm.1959.9.129
36. Manthey, R., Bry, F.: SATCHMO: A theorem prover implemented in Prolog. In: Lusk, E., Overbeek, R. (eds.) CADE 1988. LNCS, vol. 310, pp. 415–434. Springer, Heidelberg (1988). https://doi.org/10.1007/BFb0012847
37. McCune, W.: Prover9 and Mace4 (2005–2010). http://www.cs.unm.edu/~mccune/prover9. Accessed 19 July 2023
38. McNulty, G.F.: Fragments of first order logic, I: universal Horn logic. J. Symb. Log. **42**(2), 221–237 (1977). https://doi.org/10.2307/2272123
39. Nash, A., Segoufin, L., Vianu, V.: Views and queries: determinacy and rewriting. ACM Trans. Database Syst. **35**(3), 1–41 (2010). https://doi.org/10.1145/1806907.1806913
40. Nicolas, J.M.: Logics for improving integrity checking in relational data bases. Technical report, ONERA-CERT, Toulouse (1979)
41. Nicolas, J.M.: Logics for improving integrity checking in relational data bases. Acta Informatica **18**(3), 227–253 (1982). https://doi.org/10.1007/BF00263192
42. Otten, J.: Restricting backtracking in connection calculi. AI Commun. **23**(2–3), 159–182 (2010). https://doi.org/10.3233/AIC-2010-0464
43. Otten, J., Bibel, W.: leanCoP: lean connection-based theorem proving. J. Symb. Comput. **36**(1–2), 139–161 (2003). https://doi.org/10.1016/S0747-7171(03)00037-3
44. Plaisted, D.A., Greenbaum, S.: A structure-preserving clause form translation. J. Symb. Comput. **2**, 293–304 (1986). https://doi.org/10.1016/S0747-7171(86)80028-1
45. Rawson, M., Wernhard, C., Zombori, Z., Bibel, W.: Lemmas: generation, selection, application. In: Ramanayake, R., Urban, J. (eds.) TABLEAUX 2023. LNCS (LNAI), vol. 14278, pp. 153–174. Springer, Heidelberg (2023). https://doi.org/10.1007/978-3-031-43513-3_9

46. Robinson, J.A.: Automatic deduction with hyper-resolution. Int. J. Comput. Math. **1**(3), 227–234 (1965)
47. Rümmer, P.: A constraint sequent calculus for first-order logic with linear integer arithmetic. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 274–289. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89439-1_20
48. Schulz, S.: Credo Quia absurdum (?) – proof generation and challenges of proof generation. In: PAMLTP/DG4D^3 (2023), workshop presentation. https://europroofnet.github.io/_pages/WG5/Prague23/pres/Schulz.pdf
49. Schulz, S., Cruanes, S., Vukmirović, P.: Faster, higher, stronger: E 2.3. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 495–507. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_29
50. Scott, D.: A decision method for validity of sentences in two variables. J. Symb. Log. **27**(4), 477 (1962)
51. Slagle, J.R.: Automatic theorem proving with renamable and semantic resolution. JACM **14**(4), 687–697 (1967). https://doi.org/10.1145/321420.321428
52. Smullyan, R.M.: First-Order Logic. Springer, New York (1968). also republished with corrections by Dover publications (1995)
53. Stickel, M.E.: A Prolog technology theorem prover: implementation by an extended Prolog compiler. J. Autom. Reasoning **4**(4), 353–380 (1988). https://doi.org/10.1007/BF00297245
54. Sutcliffe, G., Desharnais, M.: The 11th IJCAR automated theorem proving system competition - CASC-J11. AI Commun. (2023). https://doi.org/10.3233/AIC-220244
55. Takeuti, G.: Proof Theory, second edn. North-Holland (1987)
56. Toman, D., Weddell, G.: Fundamentals of Physical Design and Query Compilation. Morgan & Claypool, San Rafael (2011). https://doi.org/10.1007/978-3-031-01881-7
57. Toman, D., Weddell, G.: An interpolation-based compiler and optimizer for relational queries (system design report). In: Eiter, T., Sands, D., Sutcliffe, G., Voronkov, A. (eds.) IWIL 2017 Workshop and LPAR-21 Short Presentations. Kalpa, vol. 1. EasyChair (2017). https://doi.org/10.29007/53fk
58. Tseitin, G.S.: On the complexity of derivation in propositional calculus. In: Slisenko, A.O. (ed.) Studies in Constructive Mathematics and Mathematical Logic, vol. Part II, pp. 115–125. Steklov Mathematical Institute (1970)
59. Van Gelder, A., Topor, R.W.: Safety and translation of relational calculus queries. ACM Trans. Database Syst. **16**(2), 235–278 (1991). https://doi.org/10.1145/114325.103712
60. Wernhard, C.: The PIE system for proving, interpolating and eliminating. In: Fontaine, P., Schulz, S., Urban, J. (eds.) PAAR 2016. CEUR Workshop Proc., vol. 1635, pp. 125–138. CEUR-WS.org (2016). http://ceur-ws.org/Vol-1635/paper-11.pdf
61. Wernhard, C.: Facets of the *PIE* environment for proving, interpolating and eliminating on the basis of first-order logic. In: Hofstedt, P., Abreu, S., John, U., Kuchen, H., Seipel, D. (eds.) INAP/WLP/WFLP -2019. LNCS (LNAI), vol. 12057, pp. 160–177. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-46714-2_11
62. Wernhard, C.: Craig interpolation with clausal first-order tableaux. J. Autom. Reasoning **65**(5), 647–690 (2021). https://doi.org/10.1007/s10817-021-09590-3
63. Wernhard, C.: Range-restricted and Horn interpolation through clausal tableaux. CoRR abs/2306.03572 (2023). https://doi.org/10.48550/arXiv.2306.03572

64. Wernhard, C., Bibel, W.: Learning from Łukasiewicz and Meredith: investigations into proof structures. In: Platzer, A., Sutcliffe, G. (eds.) CADE 2021. LNCS (LNAI), vol. 12699, pp. 58–75. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79876-5_4