Chaining Techniques for Automated Theorem Proving in Many-Valued Logics

Harald Ganzinger, Viorica Sofronie-Stokkermans
Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany
e-mail: {hg,sofronie}@mpi-sb.mpg.de

Abstract

We apply chaining techniques to automated theorem proving in many-valued logics. In particular, we show that superposition specializes to a refined version of the many-valued resolution rules introduced by Baaz and Fermüller, and that ordered chaining can be specialized to a refutationally complete inference system for regular clauses.

1. Introduction

A general method for automated theorem proving in finitely-valued logics is the *many-valued resolution* method by Baaz and Fermüller [1]. Their results have been extended in [7, 8], [10], and [2], where various versions of *signed resolution* are defined. Signed resolution rules have also been proposed for annotated logics by Kifer and Lozinskii [9] and Lu, Murray and Rosenthal [10]. Hähnle [8] has developed a hyper-resolution method for the so-called regular logics which is directly modeled after classical hyper-resolution. The completeness proofs are more or less directly derived from those for classical logic. The calculi in [10] are obtained by applying classical inference techniques.

In this paper we show that general saturation-based techniques for first-order theories of transitive relations, in particular congruences and partial or total orderings — the *chaining calculi* introduced by Bachmair and Ganzinger [3, 4] —, can easily be specialized to dealing with many-valued logics. With this method the previous results on automated theorem proving for many-valued logics can be greatly improved.

(i) Apart from reconstructing known completeness results for existing methods, including many-valued resolution [1], regular hyper-resolution [8], and annotated resolution [9, 10], the inference systems which we obtain are much more restricted, in particular by ordering constraints and selection functions.

(ii) The specialization of the general chaining inference systems is very direct and does not involve any sophisticated encodings.

(iii) The general concept of redundancy for clauses and inferences that is compatible with the chaining calculi allows us to equip the inference systems with strong techniques for simplification and for the elimination of global redundancies. Without simplification any theorem proving method, regardless of how much restricted the inferences are that are employed for proof search, will be hopelessly inefficient. Knowing what simplification techniques are admissible is, therefore, extremely important in practice. In particular, all our inference systems will be compatible with the eager rewriting of subformulas by "smaller" equivalent formulas.

(iv) The method we present allows us to use existing efficient implementations of chaining techniques including SPASS [11] and Saturate [6].

The paper is structured as follows. In Section 2 we introduce the specific terminology related to chaining inference systems. In Section 3 we describe our embedding of many-valued logic into first-order logic over transitive relations (congruences and/or orderings). In Sections 4 and 5 we show how superposition and ordered chaining can be adapted to deal with the type of clauses resulting from the translation to clause form.

2. Preliminaries

In order to avoid unnecessary complication in the presentation of our technical results we will in this paper only deal with the propositional variants of the various inference systems. That is, unless explicitly stated otherwise, all expressions (terms, literals, formulas) are assumed to be *ground*, that is, to not contain any variables. As the various completeness results that we cite from the literature, or prove in this paper, also hold for infinite sets of clauses, lifting can be done in the standard manner by viewing non-ground expressions to represent the set of their ground instances and by employing unification to avoid their explicit enumeration.

Clause logic. Literals are atomic formulae or negations thereof. Clauses are disjunctions of literals $C = L_1 \vee L_2 \vee$

$\ldots \vee L_n$; they can be also regarded as multisets, written $L_1, \ldots, L_n$. As usual, the symbols $\vee$ and $\neg$ denote disjunction and negation, respectively. Formal equality will be denoted by $\approx$, and atoms of the form $s \approx t$ are called equations. The symmetry of equality is built into the notation in that we do not distinguish between $s \approx t$ and $t \approx s$. Negative equations $\neg(s \approx t)$ are also written as $s \not\approx t$. Semantically, equality is a congruence. Consequently, a formula is called equationally satisfied in an interpretation I whenever the formula is satisfied in I, and the interpretation of $\approx$ in I is a congruence over the given signature, satisfying the respective set of *congruence axioms* Eq. We write $E[s]$ to indicate that s is a subterm of E at some position, and $E[t]$ for the result of replacing s by t at the indicated occurrence. Satisfiability and logical consequence in classical logic are defined as usual.

Orderings. Orderings on syntactic expressions play an important rôle in rewriting-based approaches to theorem proving. A (strict, partial) ordering $\succ$ is *well-founded* if there is no infinite decreasing sequence $t_1 \succ t_2 \succ \ldots$. An ordering on ground terms is called a *reduction ordering* if it is well-founded and whenever $s \succ t$ then $u[s] \succ u[t]$, for all terms u, s and t. Any ordering $\succ$ on a set S can be extended to an ordering $\succ_{mul}$ on finite multisets M over S in the following way: $M \succ_{mul} M'$ iff $M \neq M'$ and for every element a in S which occurs more often in M' than in M there exists an element $b \succ a$ in S which occurs more often in M than in M'. In other words, given a multiset M, a smaller multiset M' is obtained by (repeatedly) replacing an element b in M by finitely many (possibly zero) occurrences of a smaller element a. Usually we will drop the index mul, and use $\succ$ to denote both the ordering on elements as well as its multiset extension. If $\succ$ is total (resp. well-founded) so is its multiset extension. Thus, any literal ordering $\succ$ can be extended to a clause ordering by taking the multiset extension. We say that a literal L is maximal with respect to a clause C (denoted $L \succeq C$) if $L' \succ L$ for no literal L' in C; and that L is strictly maximal with respect to C (denoted $L \succ C$) if $L' \succeq L$ for no L' in C.

Many-valued logics and many-valued resolution. For a definition of first-order many-valued logics we refer to [1] and [2]. In [1], Baaz and Fermüller extended the resolution procedure to arbitrary finitely-valued logics: they described methods for translation to clause form, formulated a sound and complete many-valued resolution calculus, and showed that the completeness of the calculus is preserved when applying simplification rules such as subsumption and deletion of certain types of tautologies. Many-valued literals L^v are atomic formulae superscripted by truth values; many-valued clauses are disjunctions of many-valued literals.

Many-valued resolution. From $L^u \vee C_1$ and $L^v \vee C_2$ infer $C_1 \vee C_2$, provided that $u \neq v$.

Many-valued resolution has also been extended to literals signed by *sets* of truth values in [7], also see [2]. In particular, Hähnle [7, 8] has considered so-called *regular logics*. In a regular logic, the set of truth values $A = \{v_1, \ldots, v_n\}$ is totally ordered, and all sentences can be expressed in *signed clause form* where the clauses are *regular*, consisting of literals with signs of the form $\uparrow v_j := \{v \mid v \geq v_j\}$ (*positive* literals) or $\downarrow v_j := \{v \mid v \leq v_j\}$ (*negative* literals). Based on that data structure, Hähnle has proposed this many-valued variant of hyper-resolution:

Regular negative hyper-resolution. From $n \geq 1$ *electrons* $\downarrow u_1{:}L_1 \vee D_1, \ldots, \downarrow u_n{:}L_n \vee D_n$ and a *nucleus* $\uparrow w_1{:}L_1 \vee \ldots \vee \uparrow w_n{:}L_n \vee E$ derive $D_1 \vee \ldots \vee D_n \vee E$, where (i) $u_l < w_l$ for all $1 \leq l \leq n$, and (ii) all literals in $D_1, \ldots, D_n, E$ are negative.

A notion of regular signs has also been introduced in the context of annotated logics [9, 10] when the set A of truth values is a complete lattice with respect to an order $\leq$, with greatest element $\top$ and least element $\bot$. A regular sign is a a sign of the form $\uparrow v$ or $A \backslash \uparrow v$ (notation: $\sim\uparrow v$), where $v \in A$; a regular literal is a literal signed with a regular sign; a regular clause is a disjunction of regular literals. In this more general setting the following inference system was shown to be sound and refutationally complete:

Annotated resolution. From $\uparrow v_1{:}L \vee C_1$ and $\sim\uparrow v_2{:}L \vee C_2$ derive $C_1 \vee C_2$ provided that $v_1 \geq v_2$.

Annotated reduction. From $\uparrow v_1{:}L \vee C_1$ and $\uparrow v_2{:}L \vee C_2$ derive $\uparrow \mathsf{sup}(v_1, v_2){:}L \vee C_1 \vee C_2$, provided that v_1, v_2 are incomparable.

Elimination. From $\sim\uparrow\bot{:}L \vee C$ derive C.

(To all the systems above factoring has to be added if clauses are regarded as multisets rather than sets.)

The satisfiability of a set Φ of many-valued clauses (resp. regular clauses) can be defined in terms of many-valued Herbrand interpretations, $H = \{L^{v(L)} \mid L \in At(\Phi)\}$, where $v : At(\Phi) \to A$ is an assignment of truth values to the elements in the Herbrand base of Φ. A literal L^v is satisfied by such an interpretation H if $L^v \in H$. A literal $\uparrow v{:}L$ (resp. $\downarrow v{:}L$, $\sim\uparrow v{:}L$) is satisfied by H if there exists $v_L \in A$ such that $L^{v_L} \in H$ and $v \leq v_L$ (resp. $v \geq v_L$, $\neg(v \leq v_L)$). A clause is satisfied by H if at least one of its literals is satisfied by H. A set Φ of clauses is satisfiable if there exists a many-valued Herbrand interpretation which satisfies all its clauses.

3. Translation to classical logic

Many-valued and regular literals can be expressed in classical logic in a straightforward manner.

Set of truth values	Signed literal	Classical literal
arbitrary	L^v	$L \approx v$
partially ordered	$\uparrow v{:}L$ $\sim\uparrow v{:}L$	$v \leq L$ $\neg(v \leq L)$
totally ordered $(v_1 < \cdots < v_n)$	$\uparrow v_i{:}L$ $\downarrow v_i{:}L$	$v_i \leq L$ $L < v_{i+1}$ or $\neg(v_{i+1} \leq L)$

The encoding can be achieved by using a two-sorted language $\mathcal{L}_{ext}$ with sorts, ter (for terms) and for (for formulae of the many-valued logic). The set of function symbols in $\mathcal{L}_{ext}$ includes constants of sort for for all elements of A; the predicate symbols of the many-valued logic (viewed as function symbols with arguments of sort ter and result of sort for), the logical connectives of the many-valued logic as function symbols with all arguments and the result of sort for. We call the constants $v \in A$ *truth values*, and the terms of the form $R(t_1, \ldots, t_n)$, with R a predicate symbol in the language of the many-valued logic under consideration, are called *predicate terms*. Truth values will be denoted by u, v, w, s, t, and predicate terms by L. Everywhere in what follows we assume that the set of truth values is $A = \{v_1, \ldots, v_n\}$.

In the next sections we show that if Φ is a set of many-valued clauses (or of regular clauses), then we can construct a set Φ^c of classical clauses, such that Φ is unsatisfiable if and only if Φ^c is (classically) unsatisfiable. The sets Φ^c will contain additional clauses corresponding to the congruence axioms Eq (in the case of many-valued resolution) or axioms expressing the properties of $\leq$ (in the case of regular clauses). When specializing superposition (the major inference system for equational reasoning) and ordered chaining (an inference system for computing with non-symmetric transitive relations) to the sets of clauses obtained this way, many inferences will be impossible for syntactic reasons. For instance, if L is a predicate term and v a truth value, then neither L nor any of its subterms is unifiable with v. If u, v are both truth values, then they are unifiable if and only if they are (syntactically) identical. Two predicate terms L_1 and L_2 are unifiable if, and only if, they are unifiable as many-valued atoms. No proper subterm of L_1 is unifiable with L_2.

4. Many-valued resolution

Let Φ be a set of many-valued clauses, consisting of literals L^v signed by truth values. The *encoding of* Φ *in first-order logic*, Φ_1, is the set of clauses obtained from Φ by replacing every signed literal L^v by the equation $L \approx v$. Consider the following sets of clauses

$$\begin{array}{rcl} \Phi_A & = & \{u \not\approx v \mid u, v \in A, u \neq v\} \\ \mathsf{Fin} & = & \{s \approx v_1 \vee \ldots \vee s \approx v_n \mid s \text{ a term of sort } \mathsf{for}\}. \end{array}$$

They express that there are exactly n pairwise different (congruence classes of) truth values $v_1, \ldots, v_n$ in any equality Herbrand interpretation satisfying $\Phi_A \cup \mathsf{Fin}$.

Proposition 1 *A set Φ of many-valued clauses is satisfiable if and only if $\Phi_1 \cup \Phi_A \cup \mathsf{Fin}$ is (classically) equationally satisfiable.*

Definition 1 *An MV-*literal *is an equation of the form $L \approx v$, where L is a predicate term and v is a truth value. An MV-*clause *is a disjunction of MV-literals.*

We now show that a (more restricted) refinement of many-valued resolution arises as a special case of superposition.

4.1. The superposition calculus

The superposition calculus [3] encodes the congruence axioms by inference rules that are based on the computational paradigm of ordered rewriting. Superposition is actually a family of calculi, parametrized by admissible orderings $\succ$ on ground expressions. An ordering on ground expressions is called *admissible*, if (i) it is a total reduction ordering on ground terms, (ii) the ordering on ground literals is total and compatible with the ordering on terms, and (iii) the ordering on clauses is the multiset extension of the ordering on literals. For the literal ordering to be *compatible* with the term ordering, it has to satisfy (a) $L \succ L'$, if $\max(L) \succ \max(L')$, and (b) $L \succ L'$, if $\max(L) = \max(L')$ and L is negative and L' is positive, where $\max(L)$ denotes the maximal term of the equation or disequation L. The ground version of superposition consists of the following inference rules for equational clauses:[1]

Positive superposition. From $s \approx t \vee C$ and $w[s] \approx v \vee D$, derive $w[t] \approx v \vee C \vee D$ provided that conditions (i)–(v) hold.

Negative superposition. From $s \approx t \vee C$ and $w[s] \not\approx v \vee D$, derive $w[t] \not\approx v \vee C \vee D$ provided that (i)–(iii) and (iv') hold.

Reflexivity resolution. From $s \not\approx s \vee C$ derive C provided that (vi) holds.

Equality factoring. From $s \approx t \vee s \approx t' \vee C$ derive $t \not\approx t' \vee s \approx t' \vee C$ provided that (i), (iii) and (vii) hold.

Ordered factoring. From $s \approx t \vee s \approx t \vee C$ derive $s \approx t \vee C$ provided that $s \approx t$ is maximal with respect to C.

The restrictions are: (i) $s \succ t$; (ii) $w \succ v$; (iii) $s \approx t \succ C$; (iv) $w[s] \approx v \succ D$; (iv') $w[s] \not\approx v \succeq D$; (v) $w \approx v \succ s \approx t$; (vi) $s \not\approx s \succeq C$; (vii) $s \approx t \succ s \approx t'$.

Superposition is refutationally complete and compatible with a notion of redundancy for clauses and inferences which covers virtually all simplification and redundancy elimination techniques. Let T be a set of clauses intended to be built into an inference system.

[1] When lifting the calculus to clauses with variables, one can show that superposition into subterms at or below variables is not needed. In the application of superposition to many-valued (non-ground) clauses such inferences are already excluded for purely syntactic reasons.

Definition 2 *A clause D is called* T-redundant *with respect to a set N of clauses if there exist clauses $D_1, \dots, D_k$ in N such that $D \succ D_i$, $1 \le i \le k$, and for every (Herbrand) interpretation I which is a model of T, if $D_1, \dots, D_k$ are true in I then D is true in I.*

In the following definition the term *main premise* refers to the rightmost premise of an inference, while the other premises are called *side-premises*.

Definition 3 *An inference with main premise C, side premises $C_1, \dots, C_n$, and conclusion D is called* T-redundant *with respect to a set N of clauses if either some premise is redundant, or else there exist clauses $D_1, \dots, D_k \in N$ such that $C \succ D_i$, $1 \le i \le k$, and for every T-interpretation I such that $D_1, \dots, D_k, C_1, \dots, C_n$ are true in I, D is also true in I.*

A set N of clauses is called *saturated up to T-redundancy* with respect to an inference system if all inferences from non-redundant premises are redundant in N. The main result in [3] is that if $\succ$ is an admissible order and N is a set of clauses that is saturated up to Eq-redundancy with respect to superposition, either N is equationally satisfiable or N contains the empty clause.

The following simplification rules and elimination rules (where $\models$ denotes implication in equational logic) can be justified by the concept of redundancy, hence, when applied don't-care non-deterministically within a theorem proving process, refutational completeness of superposition is maintained.

Simplification. $N \cup \{C\} \vdash N \cup \{C'\}$ if $N \cup \{C\} \models C'$, $C \succ C'$, and C is redundant with respect to $N \cup \{C'\}$.

Tautology elimination. $N \cup \{C\} \vdash N$ if $\models C$.

Subsumption. $N \cup \{C, D\} \vdash N \cup \{C\}$ if $C \subset D$.

4.2. A superposition calculus for MV-clauses

In this section we show that, when applied to sets of MV-clauses, superposition specializes to a refinement of the many-valued resolution rule of Baaz and Fermüller [1]. Let SMV be the following calculus for ground clauses:

Positive MV-superposition. From $L \approx t \vee C$ and $L \approx v \vee D$ derive $C \vee D$ provided that $t \neq v$ and (i) $L \approx t \succ C$; (ii) $L \approx v \succ D$; (iii) $L \approx v \succ L \approx t$.

Ordered factoring. From $L \approx t \vee L \approx t \vee C$ derive $L \approx t \vee C$ provided that $L \approx t$ is maximal with respect to C.

For the remainder of this section we assume that $\succ$ is an admissible clause ordering such that $L \succ v$ for every predicate term L and truth value v, respectively.

Lemma 2 *Let N be a set of MV-clauses. If N is saturated up to $Eq \cup \Phi_A$-redundancy with respect to SMV then $N \cup \Phi_A \cup \mathsf{Fin}$ is saturated up to Eq-redundancy with respect to superposition.*

Proof: First we note that if a (non-empty) MV-clause C is Eq-irredundant in $N \cup \Phi_A \cup \mathsf{Fin}$ then C is $Eq \cup \Phi_A$-irredundant in N. For this, observe that all clauses in Φ_A are smaller in $\succ$ than C, containing at least one predicate term.

Assume that N is saturated up to $Eq \cup \Phi_A$-redundancy with respect to SMV. We show that all superposition inferences from irredundant clauses in $N \cup \Phi_A \cup \mathsf{Fin}$ are Eq-redundant in $N \cup \Phi_A \cup \mathsf{Fin}$.

Due to the two-level term structure, superposition into proper subterms is not needed. It is easy to see that in any superposition inference in which one of the premises (the main premise, in fact) is a clause in Φ_A the side premise must be a tautology in Fin which is redundant. Due to restrictions (i), (iii), and (v), superposition inferences with an MV-clause as main premise and a Fin clause as the side premise are not possible. Superposition inferences with a Fin clause as the main premise are of the form

$$\frac{C \vee L \approx v_i \qquad L \approx v_1 \vee \ldots \vee L \approx v_n}{C \vee v_i \approx v_1 \vee L \approx v_2 \vee \ldots \vee L \approx v_n}$$

and are redundant since the conclusion is either a tautology (if $v_i = v_1$), or else is implied by the side premise $C \vee L \approx v_i$.

Inferences by negative superposition and reflexivity resolution are not possible for MV-clauses. Equality factoring is also redundant since the conclusion has a negative literal of the form $u \not\approx v$, with $u \neq v$, which is implied by Φ_A.

The only remaining type of inferences is positive superposition between two non-redundant (in N) MV-clauses. Any such inference produces a clause of the form $C \vee D \vee u \approx v$. From the above observation about redundancy of clauses, and as N is saturated with respect to SMV, we may infer that the corresponding inference by positive MV-superposition is redundant. Hence, its conclusion $C \vee D$ is entailed (in $\Phi_A \cup Eq$-interpretations) by clauses in N smaller than the main premise. As all clauses in Φ_A are smaller than the main premise, the superposition inference is Eq-redundant in $N \cup \Phi_A \cup \mathsf{Fin}$. □

A direct consequence of the lemma is the refutational completeness of MV superposition.

Theorem 3 *Let N be a set of MV-clauses such that $N \setminus \mathsf{Fin}$ is saturated up to $Eq \cup \Phi_A$-redundancy with respect to SMV. Then either N contains the empty clause or else $N \cup \mathsf{Fin} \cup \Phi_A$ is equationally satisfiable.*

Consequently, inference with clauses in Fin, may they be present in the beginning or derived during inference, can safely be ignored. We also observe that our results about the compatibility of SMV with $Eq \cup \Phi_A$-redundancy allow us to employ the simplification and elimination rules presented in section 4.1 also for MV superposition. As superposition into subterms is not possible for MV-clauses, $\succ$ needs not be a reduction ordering on terms. We may choose for $\succ$ any well-founded and total ordering on terms.

In conclusion, the calculus SMV is an order-refinement of the many-valued resolution method of Baaz and Fermüller, and we have also shown its compatibility with $Eq \cup \Phi_A$-redundancy, and with the simplification techniques which redundancy justifies.

The next example illustrates how Saturate [6] proves the unsatisfiability of a set of MV-clauses by superposition.

Example 1 *Let* $A = \{\mathtt{t1}, \mathtt{t2}, \mathtt{tu}, \mathtt{f}\}$. *Let* N *be the set of* MV*-clauses corresponding to the set of many-valued clauses constructed in Example 5.2 in [1]:*

```
 1(1) : p(a)=f,p(a)=t1
 2(1) : p(b)=f,p(b)=t1
 3(1) : p(B)=t1,p(B)=tu,p(C)=tu
 4(1) : p(B)=t1,p(B)=tu,q(C)=tu
 5(1) : p(c)=t1,p(B)=tu
 6(1) : p(c)=t1,q(B)=tu
 7(1) : p(B)=t2,p(B)=tu,p(C)=tu
 8(1) : p(B)=t2,p(B)=tu,q(C)=tu
 9(1) : p(d)=t2,p(B)=tu
10(1) : p(d)=t2,q(B)=tu
```

and let Φ_A *be the set of clauses:*

```
11(1) :   f=t1 ->     12(1) :  f=t2 ->    13(1) :  f=tu ->
14(1) :  t1=t2 ->     15(1) : t1=tu ->    16(1) : t2=tu ->
```

Saturate finds the following proof of the empty clause from $N \cup \Phi_A$*, by using superposition, with tautology elimination, subsumption, condensement, and simplification by reduction as redundancy elimination techniques:*

```
 1 : p(a)=f,p(a)=t1           [input]
 3 : p(A)=t1,p(A)=tu,p(B)=tu  [input]
 7 : p(A)=t2,p(A)=tu,p(B)=tu  [input]
11 : f=t1 ->                  [input]
13 : f=tu ->                  [input]
14 : t1=t2 ->                 [input]
15 : t1=tu ->                 [input]
17 : p(A)=t1,p(A)=tu          [condensement of 3]
18 : p(A)=t2,p(A)=tu          [condensement of 7]
50 : t1=tu,p(a)=f,p(a)=t2     [chaining of 1 from 18]
54 : t1=t2,p(a)=f             [reduction of 50 by [15,1]]
55 : p(a)=f                   [reduction of 54 by [14]]
62 : tu=f,p(a)=t1             [chaining of 17 from 55]
64 : false                    [reduction of 62 by [13,55,11]]
Length = 14, Depth = 7; Total time:  1250 milliseconds
```

5. Resolution for regular clauses

Let $(A, \leq_A)$ be a finite partially ordered set, and $\mathsf{Min}(A)$ the set of minimal elements in A. Let Φ be a set of regular clauses, i.e. clauses containing only literals of the form $\uparrow v{:}L$ or $\sim\uparrow v{:}L$, where $v \in A$. The *encoding of* Φ *in first-order logic*, Φ_1, is the set of clauses obtained from Φ by replacing $\uparrow v{:}L$ by $v \leq L$ and $\sim\uparrow v{:}L$ by $v \not\leq L$, where $v \not\leq L$ is an abbreviation for $\neg(v{\leq}L)$. Consider the following additional sets of clauses:

$$\begin{aligned}
\Phi_A &= \{u \trianglelefteq v \mid u, v \in A, u \trianglelefteq_A v, \trianglelefteq \in \{\leq, \not\leq, \not\approx\}\} \\
\mathsf{Sup} &= \{(u \not\leq s) \vee (v \not\leq s) \vee (\mathsf{sup}(u,v) \leq s) \\
&\qquad \mid \ \mathsf{sup}(u,v) \text{ exists in } A, s \text{ a term of sort } \mathsf{for}\} \\
\mathsf{Min} &= \{ \bigvee_{m \in \mathsf{Min}(A)} (m{\leq}s) \mid s \text{ a term of sort } \mathsf{for}\}.
\end{aligned}$$

In the following we will only consider clauses with inequalities $s{\leq}t$ as atoms. In order to simplify notation, equalities $s \approx t$ will be used on the meta-level as an abbreviation for conjunctions $(s{\leq}t) \wedge (t{\leq}s)$. Fin will again denote the set of clauses (represented by)

$$\{s \approx v_1 \vee \ldots \vee s \approx v_n \mid s \text{ a term of sort } \mathsf{for}\}.$$

By Tr we denote the transitivity axiom for $\leq$:

$$(x \leq y) \wedge (y \leq z) \rightarrow (x \leq z).$$

By a *transitivity interpretation* we mean a model of Tr. We say that a set of clauses N is Tr*-satisfiable* if there exists a transitivity interpretation I that satisfies N. Otherwise N is Tr*-unsatisfiable*.

Proposition 4 *Let* Φ *be a set of regular clauses.*

(1) If $(A, \leq_A)$ *is a partially ordered set, then* Φ *is satisfiable if, and only if,* $\Phi_1 \cup \Phi_A \cup \mathsf{Fin}$ *is (classically)* Tr*-satisfiable.*

(2) If $(A, \leq_A)$ *is a sup-semilattice, then* Φ *is satisfiable if, and only if,* $\Phi_1 \cup \Phi_A \cup \mathsf{Sup} \cup \mathsf{Min}$ *is (classically)* Tr*-satisfiable.*

(3) If $(A, \leq_A)$ *is a totally-ordered set with minimal element* $\perp$ *then* Φ *is satisfiable if, and only if,* $\Phi_1 \cup \Phi_A \cup \{\perp{\leq}s \mid s$ *a term of sort* $\mathsf{for}\}$ *is (classically)* Tr*-satisfiable.*

Proposition 4 shows that if $(A, \leq_A)$ is a sup-semilattice then the set Fin can be replaced by $\mathsf{Min} \cup \mathsf{Sup}$. Results similar to those in Proposition 4(2) appear in [5] for the more restricted case of lattice-ordered regular Horn clauses.

Definition 4 *A* $\leq$-literal *is a literal of the form* $v \leq L$ *or* $v \not\leq L$*, where* L *is a predicate term and* v *is a truth value. A* $\leq$*-clause is a disjunction of* $\leq$*-literals.*

In [4] a chaining calculus for partial orderings is described. We now specialize it to $\leq$-clauses.

5.1. The ordered chaining calculus

Ordered chaining, like superposition, is a family of calculi, parametrized by admissible orderings $\succ$ on ground expressions. Admissibility of an ordering is defined like for the equational case, except that the notion of compatibility of the literal ordering with the term ordering has to be slightly extended to cope with the non-symmetry of inequations. More specifically, the literal ordering has to satisfy the conditions (a) and (b) in Section 4.1, and in addition, we require that $L \succ L'$, if $\max(L) = \max(L')$ and L and L' have the same polarity, and either $\max(L)$ occurs as the first argument of L whereas $\max(L)$ is not the first argument of L', or $\max(L)$ occurs on the same side of $\leq$ in both L and L' and $\min(L) \succ \min(L')$.

Additionally, selection can be used for controlling inferences. A *selection function* S assigns to each clause C a (possibly empty) multiset of negative literals, called the selected literals of C. With this, a negative literal L is called *eligible* in a clause C if either L is selected in C (by S), or else nothing is selected in C and L is a maximal literal in C with respect to $\succ$. A positive literal L is called *eligible* in a clause C if nothing is selected in C and L is strictly maximal in C with respect to $\succ$.

The inference rules of an ordered chaining calculus for one transitive relation $\leq$ are given in [4]. Here we adopt a slight extension with a hyper-variant of negative chaining.

Ordered chaining. $(u \leq s) \vee C$ and $(s \leq v) \vee D$ derive $(u \leq v) \vee C \vee D$ provided that condition (i) holds.

Negative Chaining 1. From $(u_1 \leq s_1) \vee C_1, \dots, (u_k \leq s_k) \vee C_k$ and $(u_1 \not\leq v_1) \vee \dots \vee (u_k \not\leq v_k) \vee D$ derive $(s_1 \not\leq v_1) \vee \dots \vee (s_k \not\leq v_k) \vee C_1 \vee \dots \vee C_k \vee D$, provided that (i')–(iii') hold.

Negative Chaining 2. From $(u_1 \leq s_1) \vee C_1, \dots, (u_k \leq s_k) \vee C_k$ and $(v_1 \not\leq s_1) \vee \dots \vee (v_k \not\leq s_k) \vee D$ derive $(v_1 \not\leq u_1) \vee \dots \vee (v_k \not\leq u_k) \vee C_1 \vee \dots \vee C_k \vee D$ provided that (i")–(iii") hold.

Ordered (positive) factoring. From $B \vee B \vee C$ derive $B \vee C$ provided that nothing is selected in C and B is maximal with respect to C

Composition resolution 1. From $(u \leq t) \vee D$ derive $(t \not\leq v) \vee (u \leq v) \vee D$, provided that $u \succ t$.

Composition resolution 2. From $(s \leq v) \vee D$ derive $(u \not\leq s) \vee (u \leq v) \vee D$ provided that (iv)–(vi) hold.[2]

Reflexivity resolution. From $(s \not\leq s) \vee C$ derive C, if $(s \not\leq s)$ is eligible.

The restrictions are: (i) $s \succ u$ and $s \succ v$ and both literals are eligible; (i') $(u \leq s_i)$ are eligible for every $1 \leq i \leq k$ in their respective side-premise; (ii') the literals $(u_i \not\leq v_i)$, $1 \leq i \leq k$, are the eligible literals in the main premise; (iii') $u_i \succ s_i$ and $u_i \succ v_i$ for every $1 \leq i \leq k$; (i") $(u_i \leq s_i)$ are eligible for every $1 \leq i \leq k$ in their respective side-premise; (ii") the literals $(v_i \not\leq s_i)$, $1 \leq i \leq k$, are the eligible literals of the main premise; (iii") $s_i \succ u_i$ and $s_i \succ v_i$ for every $1 \leq i \leq k$; $(u \leq t)$ is eligible in D; (iv) $(s \leq v)$ is eligible in D; (v) $v \succ s$; and (vi) $s \succ u$.

The non-hyper variant of ordered chaining has been proved refutationally complete for each admissible ordering $\succ$ and selection function S. Refutational completeness of the hyper-chaining variant is an easy modification. All remarks concerning redundancy and simplification made in Section 4.1 also apply to chaining inference systems.

5.2. Ordered chaining for regular clauses

Let $(A, \leq_A)$ be a finite partially ordered set, and let Φ be a set of regular clauses. By Proposition 4(1), Φ is unsatisfiable if and only if $\Phi_1 \cup \Phi_A \cup \mathit{Tr} \cup \mathsf{Fin}$ is classically unsatisfiable. Hence, Φ is unsatisfiable if and only if there is a derivation in the chaining calculus of the empty clause from $\Phi_1 \cup \Phi_A \cup \mathsf{Fin}$.

In what follows we restrict ourselves to some special cases in which inferences with clauses in Fin can be avoided.

5.2.1 $(A, \leq_A)$ is a sup-semilattice

Let CS be the following ground inference system:

Negative chaining for $\leq$-clauses. From $(u \leq L) \vee C$ and $(v \not\leq L) \vee D$ derive $C \vee D$ provided that $v \leq_A u$ and (i) holds.

Sup-reduction. From $(u \leq L) \vee C$ and $(v \leq L) \vee D$, where u and v are incomparable, derive $(\mathsf{sup}(u, v) \leq L) \vee C \vee D$ provided that (ii) holds.

Ordered (positive) factoring. From $B \vee B \vee C$ derive $B \vee C$ provided that B is maximal with respect to C.

The restrictions are: (i) $(u \leq L) \succ C$ and $(v \not\leq L) \succeq D$; (ii) $(u \leq L) \succ C$ and $(v \leq L) \succ D$.

For the remainder of this section we assume that $\succ$ is an admissible clause ordering such that $L \succ v$ for every predicate term L and truth value v, respectively; and such that for every two truth values $u, v \in A$, if $u <_A v$ then $u \succ v$. Moreover, S is assumed to be a selection function that selects both negative literals in Sup-clauses and no other literals in any other clause.

Lemma 5 *Let N be a set of $\leq$-clauses. If N is saturated up to $Tr \cup \Phi_A$-redundancy by CS then $N \cup \Phi_A \cup \mathsf{Sup}$ is saturated up to Tr-redundancy by ordered chaining.*

Proof: First we note that if a $\leq$-clause C is Tr-irredundant in $N \cup \Phi_A \cup \mathsf{Sup}$ then C is $Tr \cup \Phi_A$-irredundant in N. (As was the case previously, all clauses in Φ_A are smaller in $\succ$ than C, which contains at least one predicate term.)

Assume that N is saturated up to $Tr \cup \Phi_A$-redundancy by CS. We show that all chaining inferences (with selection function S) from irredundant clauses in $N \cup \Phi_A \cup \mathsf{Sup}$ are Tr-redundant in $N \cup \Phi_A \cup \mathsf{Sup}$.

Again, in chaining inferences in which one of the premises is a clause in Φ_A, either the other premises or the conclusion must be tautologies, and hence are redundant.

[2]Both cases of composition resolution can be further restricted, which is not essential in the present context.

The only possible inferences with a clause in Sup are (due to the selection strategy) inferences of the form:

$$\frac{C \vee (v_1 \leq L) \quad D \vee (v_2 \leq L) \quad (u \not\leq L) \vee (v \not\leq L) \vee (\mathsf{sup}(u,v) \leq L)}{C \vee D \vee (u \not\leq v_1) \vee (u \not\leq v_2) \vee (\mathsf{sup}(u,v) \leq L)} \quad (1)$$

where $(u \not\leq L)$ and $(v \not\leq L)$ are selected in the main premise, $u \leq_A v_1$, $v \leq_A v_2$, and $(v_1 \leq L)$ and $(v_2 \leq L)$ are eligible. By assumption, the CS inference:

$$\frac{C \vee (v_1 \leq L) \quad D \vee (v_2 \leq L)}{C \vee D \vee (\mathsf{sup}(v_1, v_2) \leq L)} \quad (2)$$

is $Tr \cup \Phi_A$-redundant in N. Hence, its conclusion $C \vee D \vee (\mathsf{sup}(v_1, v_2) \leq L)$ is entailed by clauses in N smaller than the main premise, together with Tr. It is easy to see that the main premise of inference (1) is larger than the main premise of inference (2). As $u \leq_A v_1$ and $v \leq_A v_2$, we obtain that $\mathsf{sup}(u,v) \leq_A \mathsf{sup}(v_1, v_2)$. From this and the fact that all clauses in Φ_A are smaller in $\succ$ than any $\leq$-clause, it follows therefore that the conclusion of inference (1) is entailed by clauses in $N \cup \Phi_A$ smaller that the main premise of (1), together with Tr. Therefore, the inference (1) is Tr-redundant in $N \cup \Phi_A \cup \mathsf{Sup}$.

Inferences by ordered chaining, the first version of negative chaining and the first version of composition resolution are syntactically excluded for $\leq$-clauses. The second version of composition resolution is also redundant since the conclusion has a negative literal of the form $(u \not\leq s)$, with $s \succ u$. By the choice of $\succ$, if $s \succ u$ then $(u \not\leq s)$ is implied by Φ_A.

The only remaining type of inferences is the second version of negative chaining between two non-redundant (in N) $\leq$-clauses. Any such inference produces a clause of the form $C \vee D \vee (u \not\leq v)$. From the above observation about redundancy of clauses, and as N is saturated with respect to CS, we may infer that the corresponding inference by negative chaining for $\leq$-clauses is redundant. Hence, its conclusion $C \vee D$ is entailed by clauses in N smaller than the main premise, together with the transitivity axioms Tr. Therefore the second negative chaining inference is Tr-redundant in $N \cup \Phi_A \cup \mathsf{Sup}$. □

A direct consequence of the lemma is the refutational completeness of the CS calculus.

Theorem 6 *Let N be a set of $\leq$-clauses such that N is saturated up to $Tr \cup \Phi_A$-redundancy with respect to CS. Then either N contains the empty clause or else $N \cup \mathsf{Sup} \cup \Phi_A$ is Tr-satisfiable.*

Corollary 7 *Let Φ be a set of regular clauses over a set A of truth values which is a sup-semilattice with minimal elements $\mathsf{Min}(A)$. Then Φ is unsatisfiable if and only if the empty clause can be derived from $\Phi_1 \cup \mathsf{Min}$ by a finite number of applications of inference rules in CS.*

We can define a hyper-variant CS^h of CS as follows. Let S be a selection function that selects all negative literals in all clauses. CS^h is obtained from CS by replacing the negative chaining rule for $\leq$-clauses with the following hyper-inference (where eligibility is defined with respect to $\succ$ and S):

Negative hyper-chaining. From $k \geq 1$ electrons $(u_1 \leq L_1) \vee C_1, \ldots, (u_k \leq L_k) \vee C_k$ and a nucleus $(v_1 \not\leq L_1) \vee \ldots \vee (v_k \not\leq L_k) \vee D$ derive $C_1 \vee \ldots \vee C_k \vee D$ provided that $v_i \leq_A u_i$ for every $1 \leq i \leq k$ and condition (i) holds.

The restrictions are: (i) the literals $(u_i \leq L_i)$ are eligible for every $1 \leq i \leq k$ in their respective side-premise; and the literals $(v_i \not\leq L_i)$, $1 \leq i \leq k$, are the eligible literals in the main premise.

Theorem 8 *Let N be a set of $\leq$-clauses such that N is saturated up to $Tr \cup \Phi_A$-redundancy with respect to CS^h. Then either N contains the empty clause or else $N \cup \mathsf{Sup} \cup \Phi_A$ is Tr-satisfiable.*

Proof: The proof uses the completeness of the variant of the chaining calculus presented in Section 5.1 and arguments similar to the ones in Lemma 5. □

5.2.2 $(A, \leq_A)$ is a complete lattice

Let $(A, \leq_A)$ be a finite complete lattice with minimal element $\bot$. In this case $\mathsf{Min} = \{\bot \leq s \mid s \text{ term of sort } \mathsf{for}\}$. By Corollary 7, every set Φ of regular clauses is unsatisfiable if and only if the empty clause can be derived from $\Phi_1 \cup \mathsf{Min}$ by a finite number of applications of inference rules in CS. In [9] the following inference rule is also considered:

Elimination. From $(\bot \not\leq L) \vee C$ derive C.

The elimination rule can be modeled by resolution with a clause in Min. In [10, 5] the assumption is made that no clause contains a literal of the form $\bot \not\leq L$ or $\bot \leq L$ (this can be achieved if all clauses containing $\bot \not\leq L$ are simplified, in a pre-processing step, by using the elimination rule, and all clauses containing $\bot \leq L$ are removed).

Theorem 9 *Let N be a set of $\leq$-clauses such that (i) N is saturated up to $Tr \cup \Phi_A$-redundancy with respect to CS; (ii) no literal in N contains the truth value $\bot$. Then either N contains the empty clause or else $N \cup \mathsf{Sup} \cup \mathsf{Min} \cup \Phi_A$ is Tr-satisfiable.*

Proof: Assume that N satisfies conditions (i) and (ii). We start by proving that $N \cup \mathsf{Min}$ is saturated up to $Tr \cup \Phi_A$-redundancy with respect to CS. Let C, D be two irredundant clauses in $N \cup \mathsf{Min}$. If both C, D are in N then, by (i), all inferences having them as premises are $Tr \cup \Phi_A$-redundant with respect to N, and, hence, also with respect to $N \cup \mathsf{Min}$. Assume that at least one of the clauses is in Min, i.e. is of the form $\bot \leq L$. It is easy to see that in any negative chaining inference in which one of the premises (the side premise, in fact) is a clause in Min the literal resolved upon in the main premise has to be of the form $\perp \not\leq L$. But no clause in $N \cup$ Min may contain literals of this form. No inferences by sup-reduction in which one of the premises is the clause $\perp \leq L$ are possible because $\perp$ is comparable with every element in A. Therefore, $N \cup$ Min is saturated up to $Tr \cup \Phi_A$-redundancy with respect to CS. By Theorem 6, either $N \cup$ Min (hence, N) contains the empty clause or else $N \cup$ Sup $\cup$ Min $\cup\ \Phi_A$ is Tr-satisfiable. □

Corollary 10 *Let Φ be a set of regular clauses over a finite set of truth values which is a complete lattice with minimal element $\perp$. Let Φ_1 be the encoding of Φ in first-order logic, and let Φ_2 be obtained from Φ_1 by removing all literals of the form $\perp \not\leq L$ and all clauses containing a literal of the form $\perp \leq L$. Then Φ is unsatisfiable if and only if there exists a derivation in CS of the empty clause from Φ_2.*

Proof: By Proposition 4, Φ is unsatisfiable if and only if $\Phi_1 \cup \Phi_A \cup$ Sup $\cup$ Min is Tr-unsatisfiable. It is easy to see that $\Phi_1 \cup$ Min and $\Phi_2 \cup$ Min are logically equivalent. No CS inference between clauses not containing $\perp$ has as conclusion a clause containing $\perp$; no CS inferences are possible between clauses not containing $\perp$ and clauses in Min or between clauses in Min. By the remarks above and Theorem 9 it follows therefore that $\Phi_2 \cup \Phi_A \cup$ Sup $\cup$ Min is Tr-unsatisfiable if and only if there is a finite derivation in CS of the empty clause from Φ_2. □

Theorem 9 shows that the calculus CS is an order-refinement of the annotated resolution calculus in [10].

5.2.3 Total orderings

Assume now that $(A, \leq_A)$ is a totally ordered set. Let CT be the inference system consisting of all inference rules in CS except sup-reduction, which never applies. The refutational completeness of the CT calculus is a direct consequence of Theorem 6.

A similar completeness result (direct consequence of Theorem 8) holds also for a hypervariant of CT, CT^h, consisting of all inference rules in CS^h except sup-reduction. In the case when Φ consists of regular Horn clauses only, CT^h specializes to the *positive unit resolution for regular Horn clauses* given in [5]. The completeness of this calculus is therefore a direct consequence of Theorem 8. Systematic renaming of positive into negative atoms provides further flexibility for the application of selection functions, and allows us to also model the negative regular hyperresolution rule in [8].

The next example illustrates how Saturate [6] proves the unsatisfiability of a set of $\leq$-clauses by ordered chaining.

Example 2 *Let $A = \{\mathtt{t}, \mathtt{u}, \mathtt{f}\}$ be a totaly ordered set of truth values, where $\mathtt{f} < \mathtt{u} < \mathtt{t}$. Let N be the set of $\leq$-clauses corresponding to the set of regular clauses that can be constructed from the formula in Example 5.1 of [1]:*

```
1(1) : B>=B
2(1) : B>=C,C>=D -> B>=D
3(1) : p(B)>=t,p(C)>=u
4(1) : p(B)>=t,p(a)>=u
5(1) : p(a)>=t -> p(B)>=t
6(1) : p(b)>=f
7(1) : p(b)>=u ->
```

and, in addition the set Φ_A and Min

```
 8(1) : t>=u       9(1) : u>=t ->    10(1) : t>=f
11(1) : f>=t ->   12(1) : u>=f      13(1) : f>=u ->
14(1) : B>=f
```

Saturate finds the following proof of the empty clause from $N \cup \Phi_A \cup$ Min by using ordered chaining, with tautology elimination, subsumption, condensement, and simplification by reduction as redundancy elimination techniques:

```
 3 :  p(A)>=t,p(B)>=u   [input]
 7 :  p(b)>=u ->        [input]
 8 :  t>=u              [input]
36 :  t>=u -> p(A)>=u   [negative chaining of 3 from 7]
46 :  p(A)>=u           [reduction of 36 by [8]]
49 :  false             [reduction of 7 by [46]]
Length = 6, Depth = 4; Total time:  760 milliseconds
```

References

[1] M. Baaz and C. Fermüller. Resolution-based theorem proving for many-valued logics. *J. of Symbolic Computation*, 19:353–391, 1995.

[2] M. Baaz, C. Fermüller, and G. Salzer. Automated deduction for many-valued logics. In A. Robinson and A. Voronkow, editors, *Handbook of Automated Reasoning*. Elsevier, to appear 1999.

[3] L. Bachmair and H. Ganzinger. On restrictions of ordered paramodulation with simplification. In *Proc. of CADE-10*, LNCS 449, pages 427–441. Springer, 1990.

[4] L. Bachmair and H. Ganzinger. Ordered chaining calculi for first-order theories of transitive relations. *Journal of the ACM*, 45(6):1007–1049, 1998.

[5] B. Beckert, R. Hähnle, and F. Manya. Transformations between signed and classical clause logic. In *Proc. of ISMVL-99*, pages 248–255. IEEE Press, 1999.

[6] H. Ganzinger, R. Nieuwenhuis, and P. Nivela. The Saturate system. Available on the World-Wide Web under URL http://www.mpi-sb.mpg.de/SATURATE/Saturate.html, 1994.

[7] R. Hähnle. Short conjunctive normal forms in finitely valued logics. *J. of Logic and Computation*, 4(6):905–927, 1994.

[8] R. Hähnle. Exploiting data dependencies in many-valued logics. *J. of Appl. Non-Classical Logics*, 6(1):49–69, 1996.

[9] M. Kifer and M. Lozinskii. A logic for reasoning with inconsistency. *J. of Automated Reasoning*, 9:179–215, 1992.

[10] J. Lu, N. Murray, and E. Rosenthal. A framework for reasoning in multiple-valued logics. *J. of Automated Reasoning*, 21(1):39–67, 1998.

[11] C. Weidenbach, B. Afshordel, U. Brahm, C. Cohrs, T. Engel, E. Keen, C. Theobalt, and D. Topic. System description: SPASS version 1.0.0. In H. Ganzinger, editor, *Proc. of CADE-16*, LNAI 1632, pages 314–318. Springer, 1999.