Commutative residual algebras the inclusion–exclusion principle

Vincent van Oostrom
University of Innsbruck, Austria

Abstract

We present a version of the inclusion–exclusion principle (IE) that can be stated and proven for commutative residual algebras (CRAs). By lifting CRAs to lattice-ordered groups the usual formulation of the IE is recovered. This provides a uniform proof of IE that applies to natural numbers with both (cut-off) subtraction or division, and for the CRAs of (measurable) (multi)sets.

2012 ACM Subject Classification Mathematics of computing → Permutations and combinations; Theory of computation → Equational logic and rewriting

Keywords and phrases multiset, commutative, residual, algebra, inclusion–exclusion, BCK

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Acknowledgements I thank A. Visser and B. Luttik and my colleagues at Utrecht University and Vrije Universiteit Amsterdam for discussions on CRAs at their inception, in 2000–2005.

1 Introduction

Multisets are formal structures frequently occurring in computation and deduction. To give a few uses of multisets: sorting a list preserves the underlying multiset, the fundamental theorem of arithmetic asserts every positive natural number is represented by a unique multiset of prime numbers, there is a multiset model of the π-calculus, and in rewriting multisets are the basis for various termination and confluence methods.

Given their prominence one would expect a relatively well-developed and -established body of multiset theory to be available. In the 1990s when working on my PhD thesis, I found this was not the case so developed the algebraic laws on multisets needed there in an ad hoc way [23, Sect. 1.4]. In the early 2000s I realised that a more principled algebraic approach was enabled by requiring composition to be *commutative* in the residual systems [26, Sect. 8.7] I had introduced, giving rise to a class of algebras dubbed *commutative residual algebras* (CRAs) [19, Sect. 5]. Multisets constitute CRAs, but initially it was open whether useful results on multisets could be established via CRAs, and whether those could be automated.

On the practical side, a first confirmation of the former was that correctness of sorting could be factored through CRAs.[1] On the theory side, the first result indicating CRAs had potential was developed by Albert Visser, who showed a representation theorem stating that any finite CRA is isomorphic to (an initial segment of) the *multiset* CRA of indecomposables, i.e. their elements *are* multisets, a result we recapitulate in the preliminaries.

In this paper we provide further evidence to the potential of CRAs, foremost, in Sect. 3, by showing that a version of the IE, i.e. the inclusion–exclusion principle, can be stated and proven for CRAs. Somewhat surprisingly, the usual inclusion–exclusion principle for (measurable) sets then is a *consequence* of that for (measurable) multisets. Embedding CRAs in lattice-ordered groups allows us to recover the IE in its usual formulation. In Sect. 4 we indicate related and future work, in particular, we show CRAs equivalent to Dvurečenskij and

[1] We showed Coq's multisets do constitute a CRA and then relied for correctness of sorting on that.

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:30
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Graziano's commutative BCK algebras with relative cancellation [13] and discuss potential automation and formalisation.

2 Preliminaries

We recapitulate commutative residual algebras from [19, Sect. 5], in particular we present the natural order and the derived (partial) operations of meet, product, and join and their core structural properties in Sect. 2.2, and the representation theorem for well-founded CRAs in Sect. 2.3. To illustrate these and also later notions, constructions, and results, we introduce in Sect. 2.1 our running examples of CRAs. Products and joins, as defined below, are in general only *partial* functions. To enable convenient reasoning about expressions in which such partial functions occur, we employ Kleene equality $\doteq$. That is, for f a partial function and expressions $e_1, \ldots, e_n$, the expression $e := f(e_1, \ldots, e_n)$ *denotes* v, if e_i denotes v_i and $(v_1, \ldots, v_n)$ is in the domain of f, and f applied to it has value v.[2] *Kleene* equality $e \doteq e'$ asserts that if either of e, e' denotes then so does the other and then their denotations are equal. This section does not contain novel material[3] (compared to [19, Sect. 5] or partially also [13], [26, Sect. 8.7]). It is meant to be a short introduction to CRAs.[4]

▶ **Definition 1.** *A* commutative residual *algebra is an algebra* $\langle A, 1, / \rangle$ *with*[5] *constant* 1 *and binary* residual *or* residuation *function* / *such that for all* $a, b, c \in A$*:*

$$a/1 = a \tag{1}$$

$$(a/b)/(c/b) = (a/c)/(b/c) \tag{4}$$

$$(a/b)/a = 1 \tag{5}$$

$$a/(a/b) = b/(b/a) \tag{6}$$

▶ Remark 2. Each of the CRA laws is independent of the others as easy models show.[6] We have not numbered the laws consecutively because we have omitted the derivable[7] ones:

$$a/a = 1 \tag{2}$$

$$1/a = 1 \tag{3}$$

and algebras satisfying laws (1)–(4) are interesting in their own right: They are *residual* algebras (RAs), the algebras corresponding to residual *systems* (RSs [26, Sect. 8.7]).[8] More precisely, such RAs correspond to RSs over a rewrite system having exactly one object, hence all results for residual systems, e.g. [26, Table 8.5], directly apply to RAs and CRAs. Where RAs have objects $a, b, c, \ldots$, RSs have steps $\phi, \psi, \chi, \ldots$. Steps allow for an intuitive visualisation of laws. For instance, why law (4) is aka the *cube* law[9] is clear from its visualisation in Fig. 1. Despite that, as discussed below, laws (5),(6) force sources and targets

---

2 We take denoting to be *strict*; e.g. $0 \cdot \frac{1}{0}$ does not denote because its sub-expression $\frac{1}{0}$ does not.
3 Maybe with the exception of the CRA of *measurable* multisets; a quick search only yielded [3].
4 Omitted proofs can also be found by ATP. See App. B for illustrative examples.
5 We use multiplicative instead of additive notation. We pronounce 1 as *unit* or *one* and a/b as *a after b*.
6 E.g. taking for / the constant-1 function satisfies all laws except (1).
7 By $a/a \overset{(1)}{=} (a/1)/a \overset{(5)}{=} 1$ and $1/a \overset{(2)}{=} (a/a)/a \overset{(5)}{=} 1$ respectively, so only using (1) and (5).
8 The correspondence is intended to be helpful for people familiar with *some* notion of *causal* equivalence [26, Sect. 8.3.1] (cf. causal invariance of [27, Sect. 5.2]) as modelled by *derivates* [22, Sect. 8], or *derivatives* [15], or *residuals* [26, Sect. 8.7] in rewrite systems and in concurrent transition systems [25].
9 Due to Lévy for rewrite systems and to Stark for concurrent transition systems, see [26, Remark 8.7.1].

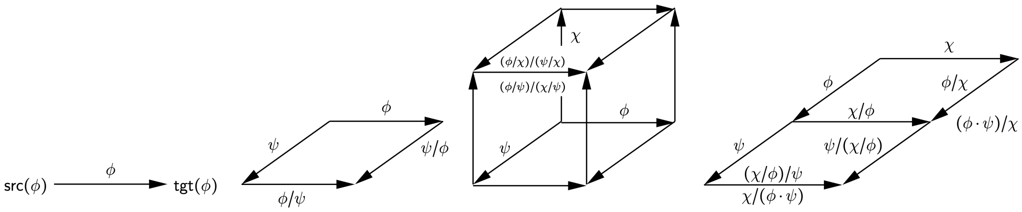

Figure 1 Rewrite step, residuation diamond, cube law (4), and composite laws (7,8). (4) states residuation of ϕ via the back (χ) and top (ψ/χ), and via the bottom (ψ) and front (χ/ψ), coincide

to coincide, trivialising the notion of step, several of our constructions do not depend on them and we will visualise such constructions in a way similar to that in which Fig. 1 visualises (4). To give a flavour of CRA reasoning we show two simple but interesting (cf. Theorem 42) laws whose proofs being not quite trivial illustrates that CRA proofs are best left to ATPs.

▶ **Proposition 3.** *$(a/b)/c = (a/c)/b$ and $(a/b)/(b/a) = a/b$.*

Proof. Abbreviating $a/(a/b)$ to $a \wedge b$ (cf. Def. 10), the former is seen to hold by

$$(a/b)/c \overset{(1,5)}{=} ((a/b)/c)/((c/b)/c) \overset{(\text{i})}{=} ((a/c)/(b/c))/(c \wedge b) \overset{(\text{ii})}{=} (a/c)/b$$

where (i) and (ii) are derived as (instances of) respectively:

$$((a/b)/c)/((c/b)/c) \overset{(4)}{=} ((a/b)/(c/b))/(c/(c/b)) \overset{(4),\text{def}}{=} ((a/c)/(b/c))/(c \wedge b)$$

$$(a'/(b/c))/(c \wedge b) \overset{(6),\text{def}}{=} (a'/(b/c))/(b \wedge c) \overset{\text{def},(4)}{=} (a'/b)/((b/c)/b) \overset{(5,1)}{=} a'/b$$

The latter, expressing parts a/b and b/a of the symmetric difference are disjoint, holds by:

$$(a/b)/(b/a) \overset{(1),(5)}{=} ((a/b) \wedge a)/((b/a) \wedge b) \overset{\text{def},(6)}{=} (a/(b \wedge a))/(b/(b \wedge a)) \overset{(4),(5),(1)}{=} a/b \quad ◀$$

2.1 Examples of CRAs

We show that some ubiquitous structures constitute CRAs. These will serve to illustrate our various operations, constructions, and results for CRAs in subsequent sections. Since among our examples the CRAs are determined by their carrier, we will refer to them via the latter.

▶ **Example 4.** The natural numbers $\mathbb{N}$ with zero 0 and monus[10] $\dot{-}$ constitute the CRA $\langle \mathbb{N}, 0, \dot{-} \rangle$. More precisely, that for all $n, m, k \in \mathbb{N}$:

$$n \dot{-} 0 = n$$

$$(n \dot{-} m) \dot{-} (k \dot{-} m) = (n \dot{-} k) \dot{-} (m \dot{-} k)$$

$$(n \dot{-} m) \dot{-} n = 0$$

$$n \dot{-} (n \dot{-} m) = m \dot{-} (m \dot{-} n)$$

can be checked by distinguishing cases on the $\leq$-order of the various sub-expressions. For instance $3 \leq 5$, so $5 \dot{-} (5 \dot{-} 3) = 3 = 3 \dot{-} 0 = 3 \dot{-} (3 \dot{-} 5)$. CRAs are also obtained when changing the carrier to the non-negative real numbers $\mathbb{R}_{\geq 0}$ and/or restricting it to an initial segment $\mathbb{N}_{\leq N}$ of numbers smaller–than–or–equal–to a given number N.

[10] **M**onus and **d**ovision are short for cut-**o**ff m**i**nus and d**i**vision, with the latter defined by $n {\cdot\!/} m := \frac{n}{\gcd(n,m)}$.

▶ Remark 5. Adjoining a fresh top to the natural numbers will not yield a CRA as (6) then fails.[11] Instead 'stacking' a reverse copy of $\mathbb{N}$ on top (having the copy of 0 as top) does work.

▶ **Example 6.** The multisets over A with empty multiset $\emptyset$ and difference $-$ constitute the CRA $\langle \mathsf{Mst}(A), \emptyset, - \rangle$. That for all $M, N, L \in \mathsf{Mst}(A)$:

$$\begin{aligned} M - \emptyset &= M \\ (M - N) - (L - N) &= (M - L) - (N - L) \\ (M - N) - M &= \emptyset \\ M - (M - N) &= N - (N - M) \end{aligned}$$

follows from the previous example by pointwise extension and viewing $\mathsf{Mst}(A)$ as $A \to \mathbb{N}$. CRAs are also obtained restricting to the sets $\wp(A)$ over A, i.e. to multisets having multiplicities ≤ 1, and/or requiring supports to be finite $\mathsf{Mst}_{\mathsf{fin}}(A)$, where for a multiset M and $a \in A$, we refer to $M(a)$ as the *multiplicity* of a and to $\{a \in A \mid M(a) > 0\}$ as the *support* of M.

▶ **Example 7.** The positive natural numbers Pos with one 1 and d*o*vision[10] $\cdot/$ constitute the CRA $\langle \mathsf{Pos}, 1, \cdot/ \rangle$. That the CRA laws hold follows from the previous example, viewing each positive natural number as its multiset of prime factors, unique by the fundamental theorem of arithmetic. In this view dovision corresponds to monus (pointwise, on the exponents of the factors). A CRA is again obtained for any initial segment $\mathsf{Pos}_{\leq N}$ of the positive numbers.

Measurable multisets constitute a less standard example. We use a minimalistic set-up: we are only concerned with binary unions, not countable ones as in general measure theory.

▶ **Definition 8.** *An* algebra[12] $\mathcal{A}$ *is a collection of subsets of an ambient set A containing A and closed under union and complement with respect to A. A multiset M is $\mathcal{A}$-*measurable *if*

- $M^i \in \mathcal{A}$ *for each i, with $M^i := \{a \mid M(a) = i\}$ (the* set *at* height *i of M); and*
- $M^{>i} = \emptyset$ *for some i, with $M^{>i} := \bigcup_{j>i} M^j = \{a \mid M(a) > i\}$ (least i is the* height *of M)*

The idea is that those multisets are measurable at each height. Note that the M^i partition A, that the support of M can be written as $M^{>0}$, and that M is empty iff its height is 0.

▶ **Example 9.** The $\mathcal{A}$-measurable multisets $\mathsf{Mst}(\mathcal{A})$ constitute a CRA. By the above it suffices to show the multiset CRA operations preserve measurability. For M, N $\mathcal{A}$-measurable:

- $\emptyset^0 = A \in \mathcal{A}$ and $\emptyset^{>0} = \emptyset = A - A \in \mathcal{A}$; and
- $(M - N)^i = \bigcup_{j \dot{-} k = i} M^j \cap N^k \in \mathcal{A}$ and $M - N$ has height below M.

2.2 Natural order, meet, product, and join

We recapitulate the natural order, the derived operations meet, product, join, and their basic properties, illustrated in Table 1. We assume an arbitrary, fixed CRA $\langle A, 1, / \rangle$.

▶ **Definition 10.** *The* natural *order is $a \leqslant b := a/b = 1$. The* meet *$a \wedge b$ of a, b is $a/(a/b)$.*

Thus (2) expresses $\leqslant$ is reflexive, (3) that 1 is $\leqslant$-least, and (6) that $\wedge$ is commutative.

▶ **Lemma 11.**
- *$\leqslant$ is a partial order; and*
- *$\langle A, \wedge \rangle$ is a meet-semilattice, and $a \leqslant b \iff a = a \wedge b$.*

[11] As usual subtraction does not behave well on 'infinities' like such a top.
[12] In measure theory terminology; in universal algebra $\mathcal{A}$ is a sub-algebra of the Boolean algebra $\wp(A)$.

CRA	$\mathbb{N}$	$\mathbb{R}_{\geq 0}$	Mst	Pos
natural order $\leqslant$	less–than–or–equal $\leq$	idem	sub-multiset $\subseteq$	divisibility $\mid$
total?	✓	✓		
well-founded?	✓		✓(on finite)	✓
meet $\wedge$	minimum min	idem	intersection $\cap$	greatest–common–divisor gcd
product $\cdot$	sum $+$	idem	sum $\uplus$	product $\cdot$
join $\vee$	maximum max	idem	union $\cup$	least–common–multiple lcm

Table 1 The natural order, meet, product and join exemplified

Proof. Quasi-orderedness holds for residual *systems* [26, Lem. 8.7.23], anti-symmetry by:
- $a \leqslant b \leqslant a \iff (a/b = 1 \text{ and } b/a = 1) \implies a \overset{(1),\text{ass}}{=} a/(a/b) \overset{(6)}{=} b/(b/a) \overset{\text{ass},(1)}{=} b$; and
- Idempotence and commutativity are trivial. We only show associativity: $a \wedge (b \wedge c) \overset{\text{com,def}}{=} (b \wedge c)/((b \wedge c)/a) \overset{\text{def},(*)}{=} (b/(b/c))/((b/a)/(b/c)) \overset{(4)}{=} (b/(b/a))/((b/c)/(b/a)) \overset{(*),\text{def}}{=} (b \wedge a)/((b \wedge a)/c) \overset{\text{def,com}}{=} (a \wedge b) \wedge c$ using $(a/b)/c \overset{(*)}{=} (a/c)/b$ twice. ◀

▶ **Definition 12.** *c is a* product *of a, b if $a \leqslant c, c/a = b$, and a* join *if a product of a, b/a.* Products, and hence joins, are unique if they exist: suppose c and d are both products of a and b. Then by (anti-)symmetry of $\leqslant$ it suffices to show $c \leqslant d$ and that follows from $c/d \overset{(1),\text{ass}}{=} (c/d)/(a/d) \overset{(4),\text{ass}}{=} (c/a)/b \overset{\text{ass},(2)}{=} 1$. Below we employ $\cdot$ and $\vee$ to denote the (partial) product and join functions. They are exemplified in Table 1.

▶ **Example 13.** In the CRA $\wp(A)$ of subsets of A product, i.e. disjoint union, is partial; products exist iff sets are disjoint. For the CRA of (measurable) multisets $\uplus$ and $\cup$ are total.

▶ **Lemma 14.** *if $a \cdot b$ denotes then, see Fig. 1:*

$$c/(a \cdot b) = (c/a)/b \tag{7}$$

$$(a \cdot b)/c \doteq (a/c) \cdot (b/(c/a)) \tag{8}$$

- *$\langle A, 1, \cdot\rangle$ is a partial commutative monoid, and $a \leqslant b \iff b \doteq a \cdot (b/a)$;*
- *$\langle A, \vee\rangle$ is a partial join-semilattice with neutral 1, and $a \leqslant b \iff a \vee b \doteq b$;*
- *if $c \leqslant a$ and $a \cdot b$ denotes so does $c \cdot b$, and the same for $\vee$.*

Proof. See [19, Lemmata 74–76]. To give a flavour of reasoning with Kleene equality we show $\cdot$ commutative, i.e. $a \cdot b \doteq b \cdot a$. Assume $c \doteq a \cdot b$, i.e. $a/c = 1$ and $c/a = b$. Then $b/c \overset{\text{ass},(5)}{=} 1$ and $c/b \overset{\text{ass},(6)}{=} a/(a/c) \overset{\text{ass},(1)}{=} a$, so also $c \doteq b \cdot a$. That is, based on the lhs we constructed a purported witness (c) for the rhs and showed it indeed satisfied being a product of b and a. In that spirit, compatibility of product as in the last item is witnessed by $(a \cdot b)/(a/c)$. ◀

▶ Remark 15. Motivated by that multiset sum is commutative, we originally arrived at commutative residual algebras, laws (5),(6), based on the following attempt to *make* composition commutative in residual systems with composition [26, Def. 8.7.38] using laws (7),(8):

$$(a \cdot b)/(b \cdot a) \overset{(7),(8)}{=} \overbrace{((a/b)/a)}^{(5)} \cdot \overbrace{((b/(b/a))/(a/(a/b)))}^{(6)} \overset{?}{=} 1 \cdot 1 = 1$$

2.3 The multiset representation theorem for well-founded CRAs

We recapitulate from [19, Sect. 5] that well-founded CRAs can be represented as multiset CRAs, by an appeal to the unique decomposition theorem for decomposition orders (the main result of [19]). We assume an arbitrary but fixed partial commutative monoid $\langle A, 1, \cdot \rangle$.

▶ **Definition 16.**
- *a is* indecomposable[13] *if $a \neq 1$ and $a = b \cdot c$ implies $b = 1$ or $c = 1$;*
- *multiset $[a_1, \ldots, a_n]$ is a* decomposition *of a if each a_i is indecomposable and $a \doteq a_1 \cdot \ldots \cdot a_n$;*
- divisibility *is defined by $a \leqslant b$ if $b \doteq a \cdot c$ for some c.*

These notions apply to CRAs via the partial commutative monoid of their product and the natural order of the CRA then coincides with the divisibility order (Lem. 14).

▶ **Definition 17.** *a partial order $\preccurlyeq$ is a* decomposition order *if*

*(*well-founded*) there are no infinite descending $\prec$-chains;*
*(*least*) $1 \preccurlyeq a$ for all a;*
*(*strictly compatible*) if $a \prec b$ and $b \cdot c$ denotes, then $a \cdot c$ denotes and $a \cdot c \prec b \cdot c$;*
*(*Riesz decomposition*) if $a \preccurlyeq b \cdot c$, then $a = b' \cdot c'$ for some $b' \preccurlyeq b$ and $c' \preccurlyeq c$;*
*(*Archimedean*) if a^n defined and $a^n \prec b$ for all n, then $a = 1$.*

Having *unique decompositions* means that decompositions exist and are unique. It trivially fails for $\mathbb{R}_{\geq 0}$ in the absence of indecomposables; its natural order $\leqslant$ is not well-founded.

▶ **Theorem 18** ([19]). *Unique decomposition holds iff there exists a decomposition order, in particular if divisibility is well-founded, strictly compatible, and has Riesz decomposition.*

Having a partial commutative monoid suffices; neither a ring structure, nor having cancellation as in the standard abstract algebraic approach to the fundamental theorem of arithmetic (FTA; for unique factorisation domains), nor totality of products, are needed. As a consequence the proof of Thm. 18 is very different from the usual proofs of the FTA (it is based on *Milner's technique*). Decomposition orders were designed, and have been applied, to show that *every process can be uniquely decomposed as the parallel composition of sequential processes* for process calculi such as BPP, ACP$^{\epsilon}$, and the π-calculus (search [19] for pointers) but, as they are complete, they also cover the FTA, separation algebras,[14], and well-founded CRAs:

▶ **Corollary 19** ([19]). *Well-founded CRAs have unique decomposition.*

Proof. By the if-part of Thm. 18 using Lem. 14: well-foundedness is immediate; strict compatibility holds since if $b \cdot c$ denotes and $a < b$, then $a \cdot c$ denotes and $a \cdot c \leqslant b \cdot c$ by compatibility, so $a \cdot c < b \cdot c$ as $(b \cdot c)/(a \cdot c) \stackrel{\text{com},(7),(8),(2),(1)}{=} b/a \neq 1$ by assumption; and finally Riesz decomposition holds since if $a \preccurlyeq b \cdot c$ setting $b' := b/d$ and $c' := c/(d/b)$ where $d := (b \cdot c)/a$ is seen to work; e.g., $a \stackrel{\text{ass}}{=} a/(a/(b \cdot c)) \stackrel{(6)}{=} (b \cdot c)/d \stackrel{(8)}{=} b' \cdot c'$. ◀

For the CRA $\mathbb{N}$ this boils down to the triviality $n = \overbrace{1 + \ldots + 1}^{n}$. For Pos we recover[15] FTA.

▶ **Theorem 20** ([19]). *A well-founded CRA $\langle A, 1, / \rangle$ is isomorphic to the CRA $\langle A', \emptyset, - \rangle$, with A' the initial segment wrt. sub-multiset $\subseteq$, of finite multisets of indecomposables of A.*

---

[13] For rings this is known as being *irreducible*.
[14] *Substate* is well-founded for the partial functions with finite domain in [6]; indecomposables are singletons.
[15] Thm. 18 should be applied directly though to avoid circularity; we *used* FTA in showing Pos a CRA.

Proof. Let h map $a \in A$ to the finite multiset $h(a) = [a_1, \dots, a_n]$ of indecomposables a_i such that $a \doteq a_1 \cdot \dots \cdot a_n$. Observe that for any a, b we have $a \doteq (a/b) \cdot (a/(a/b))$, so if a is indecomposable then a/b is 1 if $a = b$, and a otherwise.[16] Hence if $h(a) = [a_1, \dots, a_n]$ and $h(b) = [b_1, \dots, b_m]$, then $h(a/b) = [a_1, \dots, a_n] - [b_1, \dots, b_m]$ is seen to hold by repeated cancellation, using (7),(8), of the b_j occurring among the a_i in $(a_1 \cdot \dots \cdot a_n)/(b_1 \cdot \dots \cdot b_m)$. ◀

Thus, elements of well-founded CRAs *are* finite multisets in the same way positive natural numbers *are* multisets of prime numbers, (The CRA need not be finite though; e.g. $\mathbb{N}$ is not.)

3 The inclusion–exclusion principle

A basic tool in combinatorics is the inclusion–exclusion principle going back to de Moivre, da Silva, and Sylvester in the 17/18th century. In some standard formulation it reads:

▶ **Theorem 21.** *For a finite family $A_I := (A_i)_{i \in I}$ of finite sets*

$$\left|\bigcup A_I\right| = \sum_{\emptyset \subset J \subseteq I} (-1)^{|J| \dot{-} 1} \cdot \left(\left|\bigcap A_J\right|\right)$$

Spelling that out for index sets of sizes 2 and 3 gives, for finite sets A, B, C, the well-known:

$$\begin{aligned} |A \cup B| &= |A| + |B| - |A \cap B| \\ |A \cup B \cup C| &= |A| + |B| + |C| - |A \cap B| - |B \cap C| - |C \cap A| + |A \cap B \cap C| \end{aligned}$$

For instance, for $I := \{1, 2, 3\}$, $a_1 := \{x, y\}$, $a_2 := \{y, z\}$, and $a_3 := \{z, x\}$,

$$|\{x, y, z\}| = 3 = |\{x, y\}| + |\{y, z\}| + |\{z, x\}| - |\{y\}| - |\{z\}| - |\{x\}| + |\emptyset|$$

The inclusion–exclusion principle and its standard binomials-based proof have been generalised to various other settings, e.g. to probabilities and to multisets. Our starting point here is the observation that analogues of the IE hold in each of the CRAs in Sect. 2.1. For instance, for $a_1 := 6$, $a_2 := 15$, and $a_3 := 10$ in $\langle \mathbb{N}, 1, / \rangle$,

$$\max(6, 15, 10) = 6 + 15 + 10 - \min(6, 15) - \min(15, 10) - \min(10, 6) + \min(6, 15, 10)$$

Since CRAs only deal with *natural* resources we formulate a version of the IE where the positive/negative resources (for index sets of odd/even cardinality) are grouped together, with the former as large as the latter. Since products need not exist we use Kleene equality. Our proof of IE relies on how CRA operations interact with others, as summarised in:

▶ **Lemma 22.** **1.** $(b/a) \wedge (c/a) = (c/a)/(c/b) = (b \wedge c)/(a \wedge c)$;
2. $(a \cdot b)/(c \cdot d) = (a/c)/(d/b)$, *if* $c \leqslant a$, $b \leqslant d$, *and* $a \cdot b$ *and* $c \cdot d$ *denote;*
3. $(a \cdot b) \wedge c \doteq (a \wedge c) \cdot (b \wedge (c/a))$, *if* $a \cdot b$ *denotes;*
4. $(a \vee b) \wedge c \doteq (a \wedge c) \vee (b \wedge c)$, *if* $a \vee b$ *denotes; and*
5. $a \vee (a \wedge b) \doteq a$ *and* $a \wedge (a \vee b) \doteq a$, *if* $a \vee b$ *denotes.*
If product is total CRAs are distributive lattices, not necessarily bounded as shown by $\mathbb{N}$.

▶ **Theorem 23.** *If $a_I := (a_i)_{i \in I}$ is a finite family and $\prod_{J_o \subseteq I} \bigwedge a_J$, $\prod_{\emptyset \subset J_e \subseteq I} \bigwedge a_J$ denote:*[17]

$$\bigvee a_I \doteq \left(\prod_{J_o \subseteq I} \bigwedge a_J\right) / \left(\prod_{\emptyset \subset J_e \subseteq I} \bigwedge a_J\right)$$

[16] That is, indecomposables are *orthogonal letters* in the sense of [26, Example 8.7.13].
[17] The subscripts 'o'/'e' to the subset-symbol '$\subseteq$' indicate restriction to subsets of odd/even cardinality.

Proof. We mimic the standard inductive proof of IE adapting it as needed to deal with partiality of product and join in CRAs. More precisely, letting O and E be the first and second argument of the / in the rhs, i.e. the odd and even products, we show that if O, E denote, then $\bigvee a_I \doteq O/E$ and $1 = E/O$ by induction on the cardinality of the index set I.

As the base case, $I = \emptyset$, is trivial, consider the step-case for $I \cup \{k\}$, so that $O := \prod_{J_o \subseteq I \cup \{k\}} \bigwedge a_J$ and $E := \prod_{\emptyset \subset J_e \subseteq I \cup \{k\}} \bigwedge a_J$. We show that the rhss O/E and E/O of the left and right conjuncts can be stepwise transformed into their respective lhss. To that end, we first split the products in O, E into ones that do and do not contain a_k, so that O is transformed into $a \cdot b \cdot a_k$ and E into $c \cdot d$ for

$$a := \prod_{J_o \subseteq I \cup \{k\}} \bigwedge a_J, b := \prod_{\emptyset \subset J_e \subseteq I} \bigwedge (a_j \wedge a_k)_{j \in J}, c := \prod_{\emptyset \subset J_e \subseteq I} \bigwedge a_J, d := \prod_{J_o \subseteq I} \bigwedge (a_j \wedge a_k)_{j \in J}$$

using $\cdot$ is a partial monoid (to rearrange factors) and $\wedge$ a meet-semilattice (to distribute a_k).

Using that, we transform the rhs O/E of the left conjunct $\bigvee a_I \doteq O/E$ as $((a \cdot b) \cdot a_k)/(c \cdot d) \overset{(8)}{\doteq} ((a \cdot b)/(c \cdot d)) \cdot (a_k/((c \cdot d)/(a \cdot b))) \overset{\text{com,Lem. 22(2)}}{\doteq} ((a/c)/(d/b)) \cdot (a_k/((d/b)/(a/c)))$, where the conditions $c \leqslant a$ and $b \leqslant d$ of Lem. 22(2) are satisfied by the right conjunct of the IH for the families a_I respectively $(a_i \wedge a_k)_{i \in I}$. We see that, for the same families, the left conjunct of the IH applies to the occurrences of a/c and d/b in $((a/c)/(d/b)) \cdot (a_k/((d/b)/(a/c)))$ giving

$$((\bigvee a_I)/(\bigvee (a_i \wedge a_k)_{i \in I})) \cdot (a_k/((\bigvee (a_i \wedge a_k)_{i \in I})/(\bigvee a_I)))$$

From this we conclude, using $\vee$ is a join-semilattice and distributivity of $\wedge$ over $\vee$, by

$$((\bigvee a_I)/(a_k \wedge \bigvee a_I)) \cdot a_k = ((\bigvee a_I)/a_k) \cdot a_k = a_k \vee \bigvee a_I = \bigvee a_{I \cup \{k\}}$$

Further transforming the rhs E/O of the right conjunct as $(c \cdot d)/((a \cdot b) \cdot a_k) \overset{\text{(7),com,Lem. 22(2)}}{=} ((d/b)/(a/c))/a_k$, we see that, for the same families as above, the right conjunct of the IH applies to the occurrences of a/c and d/b, and then we conclude by

$$((\bigvee (a_i \wedge a_k)_{i \in I})/(\bigvee a_I))/a_k \overset{\text{Lem. 22(4)}}{=} ((a_k \wedge \bigvee a_I)/(\bigvee a_I))/a_k = 1$$ ◀

This theorem entails all the versions of the inclusion–exclusion principle we know of.

3.1 The inclusion–exclusion principle for (measurable) sets

Although the inclusion–exclusion principle for CRAs does not *directly* cover the standard one for (measurable) sets as it does not refer to cardinalities/measures, we show it can be recovered by showing such sets can be embedded into the CRA of (measurable) multisets.

▶ **Definition 24.** *The* cardinality $|M|$ *of multiset* M *over* A *is* $\sum_{a \in A} M(a)$ *([23, Def. 1.4.3]).*

This definition is such that viewing a set as a multiset preserves its cardinality. That, if $N \subseteq M$ for finite multisets M, N then $|M - N| = |M| \dot{-} |N|$, follows from:

▶ **Lemma 25.** *For finite multisets* M, N, $|M \uplus N| = |M| + |N|$.

▶ **Theorem 26.** *for a non-empty finite family* $a_I := (a_i)_{i \in I}$ *of finite sets*

$$\left|\bigcup A_I\right| = \left(\sum_{J_o \subseteq I} \left|\bigcap A_J\right|\right) \dot{-} \left(\sum_{\emptyset \subset J_e \subseteq I} \left|\bigcap A_J\right|\right)$$

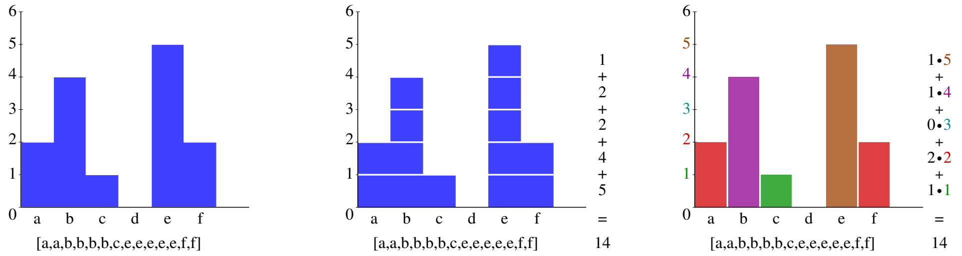

Figure 2 $\sum_i \mu(M^{>i}) = \sum_j j \cdot \mu(L^j)$, horizontal = vertical ('Lebesgue = Riemann')

Proof. Viewing sets as multisets Thm. 23 yields the left equality in:

$$\left|\bigcup A_I\right| = \left|\left(\biguplus_{J_o\subseteq I}\bigcap A_J\right) - \left(\biguplus_{\emptyset\subset J_e\subseteq I}\bigcap A_J\right)\right| = \left(\sum_{J_o\subseteq I}\left|\bigcap A_J\right|\right) \dot{-} \left(\sum_{\emptyset\subset J_e\subseteq I}\left|\bigcap A_J\right|\right)$$

with the right equality following from Lem. 25. ◀

▶ **Definition 27.** *A function μ from an algebra $\mathcal{A}$ to $\mathbb{R}_{\geq 0}$ is a* measure *if $\mu(\emptyset) = 0$ and $\mu(A \cup B) = \mu(A) + \mu(B)$ for disjoint $A, B \in \mathcal{A}$. For multisets M, $\mu(M) := \sum_i \mu(M^{>i})$.*

▶ **Lemma 28.** *For μ a measure and multisets M, N, $\mu(M \uplus N) = \mu(M) + \mu(M)$.*

Proof. Based on that $\sum_i \mu(M^{>i}) = \sum_j j \cdot \mu(L^j)$, see Fig. 2, we conclude by

$$\mu(M \uplus N) = \sum_{j,k} (j+k) \cdot \mu(M^j \cap N^k) = \mu(M) + \mu(M)$$ ◀

Replacing cardinalities by measures and 25 by 28 in the proof of Thm. 26 shows:

▶ **Theorem 29.** *for a non-empty finite family $a_I := (a_i)_{i\in I}$ of measurable sets*

$$\mu\left(\bigcup A_I\right) = \left(\sum_{J_o\subseteq I} \mu\left(\bigcap A_J\right)\right) \dot{-} \left(\sum_{\emptyset\subset J_e\subseteq I} \mu\left(\bigcap A_J\right)\right)$$

3.2 The inclusion–exclusion principle in lattice-ordered groups

By the very nature of CRAs being about *natural* resources there is still a discrepancy between the standard *formulation* of the IE in Thm. 21 and the one of Thm. 26; they are statements about the (group of) integers respectively the (monoid of) natural numbers. We show the standard formulation of the IE can be regained by embedding CRAs into lattice-ordered groups, in a way analogous to the representation of rational numbers as fractions, pairs of integers. We assume an arbitrary but fixed CRA $\langle A, 1, / \rangle$ and for simplicity that products exist turning (7) and (8) into ordinary equalities (cf. RSs *with composition* [26, Def. 8.7.38]).

▶ **Definition 30.** *A* fraction *is a pair (a, b), usually written as $\frac{a}{b}$.*

An element a of the CRA is embedded as the fraction $\frac{a}{1}$, so 1 is embedded as $\frac{1}{1}$. Fractions constitute an involutive monoid, i.e. a monoid with *reciprocal* $(\ )^{-1}$ that is an *involution* $(f^{-1})^{-1} = f$ and *anti-automorphic* $(f \cdot g)^{-1} = g^{-1} \cdot f^{-1}$ for all fractions f, g. The involutive monoid is not yet a (commutative) group. To that end we then consider fractions up to normalisation, in the same way that fractions representing rationals are normalised.

▶ Remark 31. The construction of the involutive monoid does not need commutativity of the residual algebra. (More precisely, among the laws (1)–(8) the laws (5),(6) for commutativity are not needed.) Indeed, fractions can be defined for residual systems with composition [26, Def. 8.7.38] as *valleys*, i.e. reductions having the same target (*co-spans* or *fractions* in category theory). In line with Remark 2 we employ this below, in Fig. 3, to visualise the proof of associativity of the product of fractions. That visualisation also provides the intuition for the product as juxtaposition followed by *turning the resulting peak into a valley* by means of *confluence* (cf. that confluence is equivalent to transitivity of joinability).

▶ **Lemma 32.** $\langle A\times A, 1, \cdot, (\ )^{-1}\rangle$ *is an involutive monoid for* $\frac{a}{b}\cdot\frac{a'}{b'} := \frac{a\cdot(a'/b)}{b'\cdot(b/a')}$ *and* $(\frac{a}{b})^{-1} := \frac{b}{a}$.

Proof. Reciprocal $(\ )^{-1}$ is an involution by $((\frac{a}{b})^{-1})^{-1} = (\frac{b}{a})^{-1} = \frac{a}{b}$ and anti-automorphic by $(\frac{a}{b}\cdot\frac{a'}{b'})^{-1} = \left(\frac{a\cdot(a'/b)}{b'\cdot(b/a')}\right)^{-1} = \frac{b'\cdot(b/a')}{a\cdot(a'/b)} = \frac{b'}{a'}\cdot\frac{b}{a} = (\frac{a'}{b'})^{-1}\cdot(\frac{a}{b})^{-1}$. Associativity is in Fig. 3. ◀

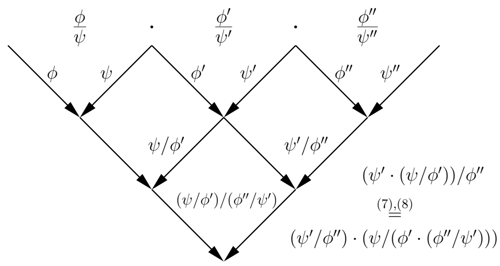

Figure 3 Associativity of product for fractions, aka *associativity by orthogonality*

▶ **Example 33.** The CRA $\mathbb{N}$ has all sums. Fractions are pairs (n, m) which we may think of as comprising assets n and debts m.[18] Their components do not cancel, they do not constitute a group, and commutativity fails as illustrated by $(2,7)+(7,2) = (2,2) \neq (7,7) = (7,2)+(2,7)$.

For CRAs normalising fractions suffices to obtain a commutative group.[19]

▶ **Lemma 34.** $\langle (A\times A)/{\equiv}, 1, \cdot, (\ )^{-1}\rangle$ *is a commutative group, embedding the monoid* $\langle A, 1, \cdot\rangle$, *where* $\equiv$ *relates fractions having the same normalisation where the* normalisation *of* $\frac{a}{b}$ *is* $\frac{a/b}{b/a}$.

Proof. We have an embedding of the CRA as follows from that $a\cdot b$ embeds as $\frac{a\cdot b}{1} = \frac{a\cdot(b/1)}{1\cdot(1/b)}$, the product of $\frac{a}{1}$, $\frac{b}{1}$. Normalisation is the identity on embeddings. We show $\equiv$ is a congruence for the operations, obtaining an involutive monoid by Lem. 32 and quotienting $\equiv$ out. Both the additional law needed to constitute a group, $f^{-1}\cdot f \equiv 1$, and commutativity hold:

All and only fractions of shape $\frac{a}{a}$ normalise to the unit 1, i.e. $\frac{1}{1}$. From this the group law is seen to hold for a fraction $f := \frac{a}{b}$ by $f^{-1}\cdot f = \frac{b}{a}\cdot\frac{a}{b} = \frac{b}{b} \equiv 1$. To see $\equiv$ is a congruence for reciprocal suppose $f' := \frac{a'}{b'}$ such that $f \equiv f'$. By definition of normalisation then $a/b = a'/b'$ and $b/a = b'/a'$, hence $f^{-1} = \frac{b}{a} \equiv \frac{b/a}{a/b} = \frac{b'/a'}{a'/b'} \equiv \frac{b'}{a'} = f'^{-1}$. For reasons of space we omit the proof of congruence and commutativity of product.[20] ◀

[18] Albert Visser dubbed them *stack* numbers.
[19] For RAs normalisation need not be idempotent. For CRAs $\frac{a}{b}$ is normalised iff $a\wedge b = 1$.
[20] Prover9 mostly takes a few minutes to generate the proofs; see App. B.

▶ **Example 35.** For the CRA $\mathbb{N}$, the group of normalised fractions comprises pairs of natural numbers at least one of which is 0, i.e. the usual integers constructed out of the natural numbers. The CRA Pos gives rise to the group of positive rationals represented by normalised fractions. The multiset CRA induces multisets having integer multiplicities; the *signed* multisets of [4, Sect. 7] arise by restricting to having finite support.

▶ Remark 36. Normalisation of $\frac{a}{b}$ consists in cancelling $a \wedge b$ common to a, b. Instead of basing oneself on cancellation one may alternatively rely on taking products (gcd vs. lcm): $\frac{\phi}{\psi} \equiv' \frac{\phi'}{\psi'}$ if $\phi/\phi' = \psi/\psi'$ and $\phi'/\phi = \psi'/\psi$. For instance, rationals $\frac{10}{15}$ and $\frac{14}{21}$ are seen equivalent by taking their products with $14 \cdot\!/\, 10 = 7$ and $10 \cdot\!/\, 14 = 5$.[21] Identifying fractions in this way is standard in category theory; here both ways coincide[20], cf. [7]. Interestingly, showing $\frac{a}{b} \equiv' \frac{a/b}{b/a}$ hinges exactly on the extra laws (5) and (6) CRAs have compared to RAs.

▶ **Lemma 37.** *Defining* meet $\frac{a}{b} \wedge \frac{c}{d}$ *as* $\frac{a \wedge c}{b \vee d}$ *and* join $\frac{a}{b} \vee \frac{c}{d}$ *as* $\frac{a \vee c}{b \wedge d}$ *makes the group lattice ordered for the* natural *order* $\leqslant$ *defined by* $f \leqslant g$ *if* $f = f \wedge g$ *(equivalently, if* $f \vee g = g$*).*

Proof. First observe that we may work exclusively with normalised fractions since these are preserved by joins and meets (if f and g are normalised, then so are $f \vee g$ and $f \wedge g$), hence all sub-expressions of the lattice laws yield normalised fractions as well. Next note that these laws, commutativity, associativity, idempotence, and absorption, for fractions, follow from the same laws for their numerators and denominators separately, i.e. for CRAs, which were shown above (absorption in Lem. 22(5) and the others in the preliminaries).

Since product is commutative to verify the group is $\leqslant$-ordered it suffices to show $\frac{a}{b} \cdot \frac{e}{f} \leqslant \frac{c}{d} \cdot \frac{e}{f}$ if $\frac{a}{b} \leqslant \frac{c}{d}$. Again, this can be reduced to checking CRA properties of the numerators and denominators separately. More precisely, under the assumptions $a \leqslant c$ and $d \leqslant b$ one shows:[20]

$$\begin{aligned}(f \cdot (b/e))/(a \cdot (e/b)) &= ((f \cdot (b/e))/(a \cdot (e/b))) \wedge ((f \cdot (d/e))/(c \cdot (e/d))) \\ (a \cdot (e/b))/(f \cdot (b/e)) &= ((a \cdot (e/b))/(f \cdot (b/e))) \wedge ((c \cdot (e/d))/(f \cdot (d/e)))\end{aligned}$$ ◀

▶ **Example 38.** On the integers (induced by the CRA $\mathbb{N}$) the natural order is the less–than–or–equal, on the positive rationals (induced by Pos) $\frac{a}{b} \leqslant \frac{a'}{b'}$ iff $a \mid a'$ and $b' \mid b$, so $\frac{1}{4} \leqslant \frac{1}{2}$ but not $\frac{1}{3} \leqslant \frac{1}{2}$, and on signed multisets it is pointwise less–than–or–equal of integer multiplicities.

▶ Remark 39. The natural order allows to reconstruct the CRA within the group as its positive cone $\{f \mid 1 \leqslant f\}$, and *division* $f \,/\, g$ defined by $g^{-1} \cdot f$ embeds residuation a/b for $b \leqslant a$ (defined in this way division makes sense for the involutive monoid; $f \cdot g^{-1}$ would not).

We have now introduced enough to formulate and prove an inclusion–exclusion principle for *integer* resources (lattice-ordered groups) instead of for *natural* resources (CRAs).

▶ **Theorem 40.** *For a finite family* $a_I := (a_i)_{i \in I}$ *of elements of* A *embedded as fractions*

$$\bigvee a_I = \prod_{\emptyset \subset J \subseteq I} \left(\bigwedge a_J\right)^{(-1)^{|J| \dot{-} 1}}$$

Proof. Since we have a group we may rearrange the rhs into O/E as in the proof of Thm. 23:

$$\left(\prod_{J_o \subseteq I} \bigwedge a_J\right) \Big/ \left(\prod_{\emptyset \subset J_e \subseteq I} \bigwedge a_J\right)$$

We conclude by Thm. 23 and Remark 39, noting residuation in the CRA coincides with division in the group, using that $E \leqslant O$ as shown in the proof of Thm. 23. ◀

[21] Pairs $(\frac{10}{15}, 7)$ and $(\frac{14}{21}, 5)$ of a stack number and a factor were dubbed *triples* by Albert Visser.

The inclusion–exclusion principle for cardinalities (Thm. 21 and and similarly for measurable sets) are obtained analogously, using the integers being a group to rearrange summands, relying on the CRA version of inclusion–exclusion for (measurable) sets (Theorems 26 and 29).

4 Related and future work

As already indicated by the many footnotes, this work has lots of (potential) connections (as is obvious when viewing multisets as a generalisation of sets). We give a limited account of related and future work, limited by the knowledge we have, focusing on CRAs.

4.1 Another specification: cBCK algebras with relative cancellation

CRAs have the same equational theory as commutative BCK (cBCK) algebras with relative cancellation [13]. BCI and BCK algebras are algebraic structures introduced in [18, 17, 1] unifying set difference and (reverse) implication in propositional logic. Many variations have been studied, but here we will exclusively be concerned with *commutative BCK algebras with relative cancellation*[22] as introduced by Dvurečenskij and Graziano, and refer the interested reader to [13, 12, 10, 11] for more on their background, results, and applications.

▶ **Definition 41.** *$\langle A, 1, /\rangle$ is a cBCK algebra with relative cancellation if for all a, b, c*

$$(a/b)/(a/c) \leqslant c/b \tag{9}$$
$$a/(a/b) \leqslant b \tag{10}$$
$$a \leqslant a \tag{11}$$
$$a = b \quad \text{if } a \leqslant b \text{ and } b \leqslant a \tag{12}$$
$$1 \leqslant a \tag{13}$$
$$a \wedge b = b \wedge a \tag{14}$$
$$b = c \quad \text{if } a \leqslant b, c \text{ and } b/a = c/a \tag{15}$$

where, as for CRAs, $a \leqslant b$ if $a/b = 1$ and $a \wedge b$ abbreviates $a/(a/b)$.

▶ **Theorem 42.** *$\langle A, 1, /\rangle$ is a CRA iff it is a cBCK algebra with relative cancellation.*

Proof. We employ the following equational specification of cBCK algebras with relative cancellation given in [10]:[23]

$$a/a = 1 \tag{16}$$
$$a/1 = a \tag{17}$$
$$(a/b)/c = (a/c)/b \tag{18}$$
$$a/(a/b) = b/(b/a) \tag{19}$$
$$(a/b)/(b/a) = a/b \tag{20}$$

That these laws hold for CRAs is either immediate or follows from Proposition 3. For reasons of space we omit the proof of the other direction.[20]. ◀

---

[22] Here *commutative* corresponds to (14), *with relative cancellation* to (15), and a *BCK* algebra distinguishes itself from a *BCI* algebra in that it has (13) instead of the law $a = 1$ if $a \leqslant 1$.

[23] On page 5 of [12] and also in the proof of Thm. 5.2.29 of [10], $1/a = 1$ is given instead of (17), which clearly is a typo as then we would not even have a commutative BCK algebra; a 2-point model with / interpreted as the constant-1-function shows that then (17) would not hold, but it should by (10)–(13).

By the theorem, results for such cBCK algebras can be transferred to CRAs and vice versa. For instance, [10, Lemma 5.2.12] entails that if x_I and y_J are finite families of non-negative real numbers such that $\sum x_I$ and $\sum y_J$ denote, then there is a family $z_{I\times J}$ such that $x_i = \sum_{j\in J} z_{i,j}$ for all $i \in I$, and $y_j = \sum_{i\in I} z_{i,j}$ for all $j \in J$, i.e. even if the natural order $\leqslant$ is not well-founded and FTA does not hold, a Riesz decomposition result does. Except for recapitulating basic results in the preliminaries, we have tried to avoid redundancy. In particular, the main application to the inclusion–exclusion principle is novel, we think, and also the way we constructed the lattice-ordered group from the CRA via the involutive monoid is (although constructing lattice-ordered groups from cBCK algebras is well-studied). Finally, arriving at the notion (cBCK algebras with relative cancellation were introduced shortly before the turn of the century, CRAs shortly after independently) from different perspectives lends support to the theory being of interest.

4.2 Another example: EWD 1313

Having introduced a notion one tends to stumble upon it everywhere. The multiset representation theorem, the inclusion–exclusion principle, and commutative BCK algebras with relative cancellation have been our main encounters with CRAs *in the wild*, but we had several others. Here we report on one which we like because it is short and simple and at first sight connected neither to sets nor to multisets.

The note [9] addresses the question whether there is a nice calculational proof of the fact that, stated using the conventions of the present paper, for all $n, m, k \in \mathsf{Pos}$:

$$\gcd(n, m) = 1 \quad\Longrightarrow\quad \gcd(n, m \cdot k) = \gcd(n, k)$$

As it turns out, this can be stated and proven for CRAs.

▶ **Proposition 43.** *if $a \wedge b = 1$ and $b \cdot c$ denotes, then $a \wedge (b \cdot c) = a \wedge c$.*

Proof. If $a \wedge b = 1$ and $b \cdot c$ denotes, $a \overset{\text{def},(5)}{=} (a/b) \cdot (a/(a/b)) \overset{\text{def}}{=} (a/b) \cdot (a \wedge b) \overset{\text{ass}}{=} a/b$, hence $a \wedge d \overset{\text{def}}{=} a/(a/d) \overset{(1)}{=} a/((a/d)/1) \overset{\text{ass}}{=} a/((a/d)/(b/d)) \overset{(4)}{=} a/((a/b)/(d/b)) = a/(a/(d/b)) \overset{\text{ass,def}}{=} a \wedge c$, where d is the denotation of $b \cdot c$ so that $d/b = c$ and $b/d = 1$. ◀

Instantiating the proposition for the multiset CRA yields $M \cap N = \emptyset \implies M \cap (N \uplus L) = M \cap L$. For the CRA Pos it provides the desired calculational proof. Whether it is *nice* depends on what algebraic laws one accepts, but we note that the analysis in [9] was inconclusive. Suggesting a possible way forward the author there ends with: *I would not be amazed if the uniqueness of the prime factorization were needed.* Although above we indeed used the FTA to verify that Pos is a CRA, the proof of Prop. 43 itself does *not* require unique decomposition. For instance, we may instantiate it for $\mathbb{R}_{\geq 0}$, not having unique decomposition, yielding the simple fact that for non-negative real numbers $\min(x, y) = 0 \implies \min(x, y + z) = \min(x, z)$.

4.3 Formalisation and automation

Since the 1990s a substantial amount of multiset theory has been developed and incorporated into proof assistants, see e.g. the multiset theories of Isabelle and Coq.[24] Despite the wealth

[24] In Isabelle `https://isabelle.in.tum.de/library/HOL/HOL-Library/Multiset.html` and in Coq `https://coq.inria.fr/library/Coq.Sets.Multiset.html` (with further rewriting-related results in IsaFor: `http://cl2-informatik.uibk.ac.at/rewriting/mercurial.cgi/IsaFoR/file/77914abd83e8/thys/Auxiliaries/Multiset2.thy` respectively in CoLoR: `http://color.inria.fr/doc/CoLoR.Util.Multiset.MultisetCore.html`.

of results there still seems to be room for improvement in several ways: i) there is a certain lack of structuring/abstraction;[25] concrete representations are chosen and results are proven for those, whereas different representations of multisets, e.g., as lists or as maps, each having its purpose, exist; ii) the developments support *either* multisets having finite support[26] *or* multisets having arbitrary support, but not both whereas both constitute CRAs; and iii) the theories seem to miss out on several lemmata corresponding to key CRA and cBCK algebra laws such as (4), (8) and (20). For these reasons we think it could be interesting to factor results using multisets through an abstract algebraic interface based on CRAs.[27] An interesting test-case for the construction of a lattice-ordered group out of a CRA would be to see whether the results on *signed* multisets in [4, Sect. 7] could be factored through it.[28] Formalisation should become even more interesting if finding/checking CRA laws could be automated, i.e. if some of the following could be answered in the affirmative:

- is the equational theory of CRAs decidable (for some interesting fragment)?
- if so, what is the complexity (is it worthwhile to implement this)?
- if so, can it be decided by a complete TRS?
- what is a minimal equational base?

We leave investigating these questions to future research,[29] guessing that no complete TRS exists, and noting there are simple equational bases other than CRAs and cBCK algebras with relative cancellation, e.g., (1), (5), (20) combined with

$$(a/b)/(a/c) \quad = \quad (c/b)/(c/a) \tag{21}$$

4.4 Gradification

This section assumes familiarity with rewriting. Our residual *algebras* were obtained by forgetting the sources and targets of steps in the residual *systems* of [26, Sect. 8.7]. To make the correspondence more clear we now consider the reverse direction, enriching the *objects* of our residual algebras to *steps* of rewrite systems [22][30], a process we dub *gradification*:[31]

- the *carrier* A of *objects* is lifted to a *rewrite system* $\to$ [26, Def. 8.2.2] of *steps* (Fig. 1);
- the *one*-object 1 is lifted to *loop*-steps 1_a for each object a;
- residuation / is lifted to pairs of steps requiring them to have the same sources, and targets should be preserved by exchanging steps, i.e. the Skolemised *diamond* property:
  ▶ **Proposition 44.** *$\to$ has the diamond property (Fig. 1) iff it has a residuation (App. A);*
- product $\cdot$ is lifted to *composition*; the target of the 1st step is the source of the 2nd;
- join $\vee$ lifts to pairs of steps with the same source and yields the *diagonal* of their diamond;

Proceeding like this gives rise to residual systems as in [26, Sect. 8.7]:

---

[25] A comment in the Coq theory file, seemingly without follow-up, reads *Here we should make multiset an abstract datatype, by hiding Bag, munion, multiplicity; all further properties are proved abstractly.* Cf. also the frequent usage of multiset *union* where multiset *sum* is meant.
[26] In themselves well-motivated, say by the wish for the multiset-extension to be well-founded, but making that e.g. the inclusion–exclusion principle for measurable multisets can not even be stated.
[27] Our formalisation of constructing groups from CRAs in Coq in 2001 is obsolete (not typeclass-based).
[28] E.g. is $\alpha \cdot (\gamma - \beta) + \alpha \cdot \beta = \alpha \cdot (\beta - \gamma) + \alpha \cdot \gamma$ for *truncating subtraction*, first speculated to hold and then derived there, used for the associativity of ordinal multiplication, entailed by commutativity of join?
[29] It could well be that one or more questions have been answered in the literature/have easy answers.
[30] Rewrite *systems* relate to rewrite *relations* (endorelations) as categories relate to quasi-orders.
[31] From *gradus* step. This is analogous to how monoids relate to *typed* monoids in [24]. We are primarily interested in steps and residuation, even in the absence of composition, so do not target categories.

▶ **Definition 45.** $\langle \to, 1, / \rangle$ *is a* residual system *if for co-initial* ϕ, ψ, χ *in rewrite system* $\to$*:*

$$\phi/1 = \phi \tag{1}$$

$$\phi/\phi = 1 \tag{2}$$

$$1/\phi = 1 \tag{3}$$

$$(\phi/\psi)/(\chi/\psi) = (\phi/\chi)/(\psi/\chi) \tag{4}$$

It is a residual system with composition*, for a* $\cdot$ *such that also (now for* ϕ, ψ *composable): (7)* $\chi/(\phi \cdot \psi) = (\chi/\phi)/\psi$*, (8)* $(\phi \cdot \psi)/\chi = (\phi/\chi) \cdot (\psi/(\chi/\phi))$*, and* $1 \cdot 1 = 1$*.*

Examples of rewrite systems that can be naturally equipped with residual structure abound.

▶ **Example 46.** For each of the following rewrite systems residuation is induced by the proof of the diamond property, as given in the works cited, e.g. the Tait–Martin-Löf proof that $\geq_1$ has the diamond property in the $\lambda\beta$-calculus [2]: i) β-steps in the *linear* $\lambda\beta$-calculus; ii) $\geq_1$-steps in the $\lambda\beta$-calculus [2]; iii) parallel steps $\twoheadrightarrow\!\!\!\!\!\!+$/multisteps $\multimap$ in orthogonal first/higher-order term rewrite systems [16] or [26, Sect. 8.7],[5]; iv) positive braids with parallel crossings of strands [26, Sect. 8.9]; and v) multi-redexes/treks in axiomatic residual theory [21].[32]

Although none of the residual systems in the example have compositions,[33] a residual system with composition can always be *induced* by considering finite *reductions* (*formal* compositions of steps) and *defining* residuation via the composition laws ((7),(8)) and quotienting out the equivalence induced by the natural order [26, Lem. 8.7.47,Prop. 8.7.48]. Analogously, any CRA induces a CRA with composition by considering finite *multisets* of objects. For instance, the CRA $\mathbb{B}$ of bits, i.e. $\mathbb{N}_{\leq 1}$, does not have composition, but induces the CRA $\mathbb{N}$, which does. Conversely, the construction of Lem. 32 to turn a residual algebra with composition into an involutive monoid, e.g. turning $\mathbb{N}$ into $\mathbb{Z}$, turns a residual system with composition, i.e. on *reductions*, into a *typed* involutive monoid[31] on *valleys* (instead of just on *conversions*). We intend to study this construction and more generally involutive monoids, as we think they are of interest to rewriting, cf. [14, 7].[34]

▶ Remark 47. As an indication that involutive monoids are interesting in and of themselves, note that starting from a specification of groups, the usual complete TRS [26, Tab. 7.5] for groups obtained by completion, comprises intermediate *complete* sub-TRSs obtained simply by orienting equations: first for *monoids* (by rules $1 \cdot x \to x$, $x \cdot 1 \to x$, $(x \cdot y) \cdot z \to x \cdot (y \cdot z)$), then for *involutive* monoids (adjoining $1^{-1} \to 1$, $(x \cdot y)^{-1} \to y^{-1} \cdot x^{-1}$, $(x^{-1})^{-1} \to x$ [14, App. A]), and only finally for *groups* (adjoining $x \cdot x^{-1} \to 1$, $x^{-1} \cdot x \to 1$, $x \cdot (x^{-1} \cdot y) \to y$ $x^{-1} \cdot (x \cdot y) \to y$) there are two *extended* rules, the last two, not simply obtained by orienting.

5 Conclusion

We have presented the inclusion–exclusion principle as a use-case for CRAs. Apart from the questions about deciding, automation, and formalisation raised above, we would be interested in investigating whether/how the approach could be extended to handle the *multiset extension* of orders, or could be adapted to non-well-founded multisets [8].

[32] As we will show elsewhere, the axioms of [21] are sufficient but not necessary obtain the main results of [21] *via* the theory of the residual systems in [26, Sect. 8.7].
[33] All have joins except for β-steps in linear λ-calculus and $\twoheadrightarrow\!\!\!\!\!\!+$ in orthogonal first-order term rewriting.
[34] Already strings do not just constitute a monoid but an involutive one. Going further to typed groups i.e. *groupoids*, seems to be too much in rewriting where the notion of interest is that of a *conversion*.

References

1. Y. Arai, K. Iséki, and S. Tanaka. Characterizations of BCI, BCK-algebras. *Proc. Japan Acad.*, 42(2):105–107, 1966. doi:10.3792/pja/1195522126.
2. H.P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 2nd revised edition, 1984.
3. M. Bhargava. The density of discriminants of quintic rings and fields. *Ann. Math. (2)*, 172(3):1559–1591, 2010.
4. J.C. Blanchette, M. Fleury, and D. Traytel. Nested multisets, hereditary multisets, and syntactic ordinals in Isabelle/HOL. In D. Miller, editor, *2nd International Conference on Formal Structures for Computation and Deduction, FSCD 2017, September 3-9, 2017, Oxford, UK*, volume 84 of *LIPIcs*, pages 11:1–11:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.FSCD.2017.11.
5. H.J.S. Bruggink. Residuals in higher-order rewriting. In R. Nieuwenhuis, editor, *Rewriting Techniques and Applications, 14th International Conference, RTA 2003, Valencia, Spain, June 9–11, 2003, Proceedings*, volume 2706 of *Lecture Notes in Computer Science*, pages 123–137. Springer, 2003. doi:10.1007/3-540-44881-0_10.
6. C. Calcagno, P.W. O'Hearn, and H. Yang. Local action and abstract separation logic. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10–12 July 2007, Wroclaw, Poland, Proceedings*, pages 366–378. IEEE Computer Society, 2007. doi:10.1109/LICS.2007.30.
7. F. Clerc and S. Mimram. Presenting a Category Modulo a Rewriting System. In M. Fernández, editor, *26th International Conference on Rewriting Techniques and Applications (RTA 2015)*, volume 36 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 89–105, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.RTA.2015.89.
8. G. D'Agostino and A. Visser. Finality regained: A coalgebraic study of Scott-sets and multisets. *Archive for Mathematical Logic*, 41(3):267–298, 2002. doi:10.1007/s001530100110.
9. E.W. Dijkstra. The gcd and the minimum. EWD1313, November 2001. URL: https://www.cs.utexas.edu/users/EWD/transcriptions/EWD13xx/EWD1313.html.
10. A. Dvurečenskij and S. Pulmannová. *BCK-algebras*, pages 293–377. Springer Netherlands, Dordrecht, 2000. doi:10.1007/978-94-017-2422-7_6.
11. A. Dvurečenskij and S. Pulmannová. *BCK-algebras in Applications*, pages 379–446. Springer Netherlands, Dordrecht, 2000. doi:10.1007/978-94-017-2422-7_7.
12. A. Dvurečenskij. On categorical equivalences of commutative BCK-algebras. Preprint 16/1998, June 1998.
13. A. Dvurečenskij and M.G. Graziano. Commutative BCK-algebras and lattice ordered groups. *Mathematica japonicae*, 49(2):159–174, March 1999. URL: https://ci.nii.ac.jp/naid/10010236889/en/.
14. B. Felgenhauer and V. van Oostrom. Proof orders for decreasing diagrams. In F. van Raamsdonk, editor, *24th International Conference on Rewriting Techniques and Applications, RTA 2013, June 24-26, 2013, Eindhoven, The Netherlands*, volume 21 of *LIPIcs*, pages 174–189. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013. doi:10.4230/LIPIcs.RTA.2013.174.
15. R. Hindley. An abstract form of the Church–Rosser theorem. I. *Journal of Symbolic Logic*, 34(4):545–560, 1969. doi:10.1017/S0022481200128439.
16. G. Huet and J.-J. Lévy. Computations in orthogonal rewriting systems, Part I + II. In J.L. Lassez and G.D. Plotkin, editors, *Computational Logic – Essays in Honor of Alan Robinson*, pages 395–443, Cambridge MA, 1991. MIT Press. Update of: Call-by-need computations in non-ambiguous linear term rewriting systems, 1979.
17. Y. Imai and K. Iséki. On axiom systems of propositional calculi, xiv. *Proc. Japan Acad.*, 42(1):19–22, 1966. doi:10.3792/pja/1195522169.

18 K. Iséki. An algebra related with a propositional calculus. *Proc. Japan Acad.*, 42(1):26–29, 1966. `doi:10.3792/pja/1195522171`.

19 S.P. Luttik and V. van Oostrom. Decomposition orders—another generalisation of the fundamental theorem of arithmetic. *Theoretical Computer Science*, 335(2):147–186, 2005. `doi:https://doi.org/10.1016/j.tcs.2004.11.019`.

20 W. McCune. Prover9 and mace4. `http://www.cs.unm.edu/~mccune/prover9/`, 2005–2010. URL: `http://www.cs.unm.edu/~mccune/prover9`.

21 P.-A. Melliès. Axiomatic rewriting theory VI residual theory revisited. In S Tison, editor, *Rewriting Techniques and Applications, 13th International Conference, RTA 2002, Copenhagen, Denmark, July 22–24, 2002, Proceedings*, volume 2378 of *Lecture Notes in Computer Science*, pages 24–50. Springer, 2002. `doi:10.1007/3-540-45610-4\_4`.

22 M.H.A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43:223–243, 1942. `doi:10.2307/2269299`.

23 V. van Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, Vrije Universiteit, Amsterdam, March 1994. URL: `https://research.vu.nl/en/publications/confluence-for-abstract-and-higher-order-rewriting`.

24 D. Pous. Untyping typed algebraic structures and colouring proof nets of cyclic linear logic. In A. Dawar and H. Veith, editors, *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 484–498. Springer, 2010. `doi:10.1007/978-3-642-15205-4\_37`.

25 E.W. Stark. Concurrent transition systems. *Theoretical Computer Science*, 64:221–269, 1989.

26 Terese. *Term Rewriting Systems*. Cambridge University Press, 2003.

27 S. Wolfram. A class of models with the potential to represent fundamental physics, April 2020. URL: `https://www.wolframphysics.org/technical-introduction/`.

A Proofs omitted from the main text

Proof of Lem. 22. **1.** the left equality follows from:

$$\begin{aligned}
(b/a) \wedge (c/a) &\overset{\text{(i)}}{=} ((b/a) \wedge (c/a))/((c/b) \wedge (a/b)) \\
&\overset{\text{com,def,(4)}}{=} ((c/a)/((c/b)/(a/b)))/((c/b) \wedge (a/b)) \\
&\overset{\text{(ii)}}{=} (c/a)/(c/b)
\end{aligned}$$

where (ii) follows from $(a'/(b'/c'))/(b' \wedge c') = a'/b'$ which holds[35] by definition of $\wedge$, (4), (5), and (1), and (i) is, after unfolding $\wedge$s, an instance[36] of $(a'/b')/c' = ((a'/b')/c')/(d' \wedge (b'/a'))$ which follows by Prop. 3 from $a'/b' = (a'/b')/(d' \wedge (b'/a'))$ which holds by

$$\begin{aligned}
a'/b' &\overset{\text{(ii)}}{=} (a'/(b'/d'))/(b' \wedge d') \\
&\overset{\text{Prop. 3}}{=} ((a'/(b'/d'))/(b' \wedge d'))/((b' \wedge d')/(a'/(b'/d'))) \\
&\overset{\text{(ii)}}{=} (a'/b')/((b' \wedge d')/(a'/(b'/d'))) \\
&\overset{\text{def,(4),Prop. 3,com}}{=} (a'/b')/(d' \wedge (b'/a'))
\end{aligned}$$

The right equality holds by $(c/a)/(c/b) = (c/(a \wedge c))/(c/b) \overset{\text{Prop. 3}}{=} (c/(c/b))/(a \wedge c) \overset{\text{def,com}}{=} (b \wedge c)/(a \wedge c)$;

[35] This can be seen as a consequence of the decomposition law $a \doteq (a/b) \cdot (a \wedge b)$, allowing to write any a as the sum (see below) of its residual and intersection with an arbitrary b.
[36] For $a' := b$, $b' := a$, $c' := (b/a)/(c/a)$, and $d' := c/b$.

2. under the assumptions, $(a \cdot b)/(c \cdot d) \overset{(7)}{\doteq} ((a \cdot b)/c)/d \overset{(8)}{\doteq} ((a/c) \cdot (b/(c/a)))/d \overset{\text{ass},(1)}{\doteq} ((a/c) \cdot b)/d \overset{\text{com}}{\doteq} (b \cdot (a/c))/d \overset{(8)}{\doteq} (b/d) \cdot ((a/c)/(d/b)) \overset{\text{ass},(1)}{\doteq} (a/c)/(d/b)$;

3. assuming $a \cdot b$ denotes,

$$\begin{aligned}
(a \cdot b) \wedge c &\overset{\text{def},(8),(7),(8)}{\doteq} ((a/(a/c)) \cdot (b/((a/c)/a)))/(b/(c/a)) \\
&\overset{\text{def,com},(5),(1)}{\doteq} (b \cdot (a \wedge c))/(b/(c/a)) \\
&\overset{(8)}{\doteq} (b/(b/(c/a))) \cdot ((a \wedge c)/((b/(c/a))/b)) \\
&\overset{\text{def},(5),(1),\text{com}}{\doteq} (a \wedge c) \cdot (b \wedge (c/a))
\end{aligned}$$

4. it suffices to show that $(a \vee b) \wedge c$ satisfies the two conditions for being the join $(a \wedge c) \vee (b \wedge c)$, i.e. for being the product of $a \wedge c$ and $(b \wedge c)/(a \wedge c)$. We check both in turn.

   The first condition $a \wedge c \leqslant (a \vee b) \wedge c$ holds since, under the assumption, $(a \vee b) \wedge c \doteq (a \cdot \ldots) \wedge c \doteq (a \wedge c) \cdot \ldots$ by Lem. 22(3) and (1), (2), and (7).

   The second condition is seen to hold under the assumption, by

$$\begin{aligned}
((a \vee b) \wedge c)/(a \wedge c) &\overset{\text{def}}{=} ((a \cdot (b/a)) \wedge c)/(a \wedge c) \\
&\overset{\text{Lem. 22(3)}}{\doteq} ((a \wedge c) \cdot ((b/a) \wedge (c/a)))/(a \wedge c) \\
&\overset{(8),(1),(2)}{\doteq} (b/a) \wedge (c/a) \\
&\overset{\text{Lem. 22(1)}}{=} (b \wedge c)/(a \wedge c)
\end{aligned}$$

5. the first absorption law does not need the assumption. For it, verify that a meets the conditions for being the join of a and $a \wedge b$, both of which follow trivially from $a/a = 1 = (a \wedge b)/a$. For the second absorption law we compute $a \wedge (a \vee b) = a \wedge (a \cdot (b/a)) = a$. ◀

That all items can be shown by ATP is exemplified in App. B for distributivity (item 4).

Proof of Prop. 44. The if-direction follows immediately from that ϕ/ψ and ψ/ϕ are required to have the same target, for steps ϕ, ψ having the same source.

For the only–if-direction, first note that the diamond property (cf. [26, Lem. 8.7.11]) states that for all co-initial steps ϕ, ψ, there exist co-final steps ψ', ϕ', such that ϕ is composable with ψ' and ψ with ϕ'. By Skolemisation this is equivalent to the existence of functions f, g such that for all co-initial steps ϕ, ψ, the steps $g(\phi, \psi)$, $f(\phi, \psi)$ are co-final, ϕ and $g(\phi, \psi)$ are composable, and so are ψ and $f(\phi, \psi)$.

Then let R be any asymmetric relation, total on pairs of distinct steps (such relations exist, e.g. by the well-ordering theorem), and define ϕ/ψ to be $f(\phi, \psi)$ if $\phi \; R \; \psi$ and $g(\psi, \phi)$ otherwise. We verify / has the properties required of residuation:

- if $\phi \; R \; \psi$, then $\phi/\psi = f(\phi, \psi)$ and by asymmetry $\psi/\phi = g(\phi, \psi)$. By assumption, $f(\phi, \psi)$ is composable to ψ and co-final to $g(\phi, \psi)$; and
- if not $\phi \; R \; \psi$, then $\phi/\psi = g(\psi, \phi)$ and by totality and asymmetry $\psi/\phi = f(\psi, \phi)$. By assumption, $g(\psi, \phi)$ is composable to ψ and co-final to $f(\psi, \phi)$. ◀

B Selected Prover9 proofs of properties of CRA operations

In this appendix we provide Prover9 [20] proofs of selected results from the main text.[37] All proofs were generated without further guidance. The proofs provided here should allow interested readers to reconstruct the other proofs omitted from the main text by means of ATP themselves. To that end, we provide the input-file used as an example for the first, trivial, proposition below. For the others, similar representations of the statements were used, and only the resulting proofs are given. In each case the initial part of the output allows to reconstruct (the assumptions used of) the input. To keep proofs, relatively, short we freely add already derived equations to the assumptions.

To illustrate the Prover9 input and output we make use the following proposition that was omitted from the main text, but has a short and easy to understand proof.

▶ **Proposition 48.** $\leqslant$ *is transitive in BCI algebras.*

Proof. To prove the statement we supplied Prover9 a file with contents:

```
formulas(sos).

((x / y) / (x / z)) / (z / y) = 1.
(x / (x / y)) / y = 1.
x / x = 1.
-(x / y = 1) | -(y / x = 1) | x = y.
-(x / 1 = 1) | x = 1.
-P(x,y) | x / y = 1.
-(x / y = 1) | P(x,y).

end_of_list.

formulas(goals).

-P(x,y) | -P(y,z) | P(x,z).

end_of_list.
```

upon which Prover9 provided the following proof:[38]

```
============================== PROOF =================================

% Proof 1 at 0.01 (+ 0.00) seconds.
% Length of proof is 22.
% Level of proof is 6.
% Maximum clause weight is 13.000.
% Given clauses 32.

1 -P(x,y) | -P(y,z) | P(x,z) # label(non_clause) # label(goal).  [goal].
2 ((x / y) / (x / z)) / (z / y) = 1.  [assumption].
3 (x / (x / y)) / y = 1.  [assumption].
4 x / x = 1.  [assumption].
5 x / y != 1 | y / x != 1 | x = y.  [assumption].
6 x / 1 != 1 | x = 1.  [assumption].
7 x / 1 != 1 | 1 = x.  [copy(6),flip(b)].
8 -P(x,y) | x / y = 1.  [assumption].
9 x / y != 1 | P(x,y).  [assumption].
10 P(c1,c2).  [deny(1)].
11 P(c2,c3).  [deny(1)].
12 -P(c1,c3).  [deny(1)].
24 (x / 1) / x = 1.  [para(4(a,1),3(a,1,1,2))].
27 x / (x / 1) = 1.  [hyper(7,a,3,a),flip(a)].
31 c1 / c2 = 1.  [hyper(8,a,10,a)].
32 c2 / c3 = 1.  [hyper(8,a,11,a)].
33 c1 / c3 != 1.  [ur(9,b,12,a)].
71 ((x / c3) / (x / c2)) / 1 = 1.  [para(32(a,1),2(a,1,2))].
82 x / 1 = x.  [para(24(a,1),5(a,1)),rewrite([27(6)]),xx(a),xx(b)].
85 (x / c3) / (x / c2) = 1.  [back_rewrite(71),rewrite([82(7)])].
173 c1 / c3 = 1.  [para(31(a,1),85(a,1,2)),rewrite([82(5)])].
174 $F.  [resolve(173,a,33,a)].

============================== end of proof ==========================
```

[37] To be precise, we used Prover9 version LADR-2009-11A compiled and run on a 2018 MacBook Pro with macOS Catalina 10.15.4 with a 2.2 GHz 6-core Intel Core i7 processor and 32GB of memory (but Prover9 only used 1 core and memory was not an issue).

[38] The main operations applied in the proofs here are paramodulation, hyperresolution, and rewriting. See the literature or the Prover9 documentation for more on these. Positions in expressions are represented as lists of positive natural numbers; as equality ($=$) is taken as a binary function symbol, positions in paramodulation of two equations start with 1 (usually; the lhs) or 2 (the rhs). E.g., in this proof the identity $(x/1)/x = 1$ on the line numbered 24 is obtained by unifying the lhs of that at line numbered 4 with the subterm at position 1.2, i.e. the subterm x/y, in the lhs of the identity at line numbered 3.

◀

Proof of Lem. 22(4). It is shown meet distributes over join in CRAs.

```
============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.09 (+ 0.00) seconds.
% Length of proof is 38.
% Level of proof is 7.
% Maximum clause weight is 27.
% Given clauses 43.

1 x ^ (y v z) = (x ^ y) v (x ^ z) # label(non_clause) # label(goal).  [goal].
2 x / 1 = x.  [assumption].
4 x / x = 1.  [assumption].
5 (x / y) / (z / y) = (x / z) / (y / z).  [assumption].
6 (x / y) / x = 1.  [assumption].
7 x ^ y = x / (x / y).  [assumption].
9 x ^ y = y ^ x.  [assumption].
10 x / (x / y) = y / (y / x).  [copy(9),rewrite([7(1),7(3)])].
13 (x ^ y) / z = (x / z) ^ (y / z).  [assumption].
14 (x / (x / y)) / z = (x / z) / ((x / z) / (y / z)).  [copy(13),rewrite([7(1),7(6)])].
15 x v y = x * (y / x).  [assumption].
16 x v x = x.  [assumption].
17 x * 1 = x.  [copy(16),rewrite([15(1),4(1)])].
18 x v y = y v x.  [assumption].
19 x * (y / x) = y * (x / y).  [copy(18),rewrite([15(1),15(3)])].
22 (x * y) / z = (x / z) * (y / (z / x)).  [assumption].
23 (x / y) * (z / (y / x)) = (x * z) / y.  [copy(22),flip(a)].
24 x / (y * z) = (x / y) / z.  [assumption].
25 (x / y) / z = x / (y * z).  [copy(24),flip(a)].
26 (x / y) / z = (x / z) / y.  [assumption].
27 (c1 ^ c2) v (c1 ^ c3) != c1 ^ (c2 v c3).  [deny(1)].
28 (c1 / (c1 / c2)) * ((c1 / (c1 / c3)) / (c1 / (c1 / c2))) != c1 / ((c1 / c2) / (c3 / c2)).  [copy(27),rewrite([7(3),7(8),15(11),15(21),7(24),25(25,R)])].
32 (x / (y / z)) / (y / (y / z)) = x / y.  [para(6(a,1),5(a,1,2)),rewrite([2(3)]),flip(a)].
33 ((x / y) / (z / y)) / (x / z) = 1.  [para(5(a,1),6(a,1,1))].
34 (x / y) / ((x / z) / (y / z)) = (x / y) / ((x / y) / (z / y)).  [para(5(a,1),7(a,2,2)),rewrite([7(3)]),flip(a)].
37 (x / (y / z)) / (z / (z / y)) = x / y.  [para(10(a,1),5(a,1,2)),rewrite([6(8),2(8)])].
39 x / (x / (x / y)) = x / y.  [para(6(a,1),10(a,1,2)),rewrite([2(3)]),flip(a)].
93 (x * y) / y = x.  [para(10(a,1),15(a,2,2)),rewrite([15(2),19(4),6(2),17(2),23(4)]),flip(a)].
121 (x / (x / y)) * z = (x * z) / (x / y).  [para(6(a,1),23(a,1,2,2)),rewrite([2(4)])].
136 (c1 * ((c1 / (c1 / c3)) / (c1 / (c1 / c2)))) / (c1 / c2) != c1 / ((c1 / c2) / (c3 / c2)).  [back_rewrite(28),rewrite([121(17)])].
203 (x / y) / ((x / y) / (z / y)) = (x / y) / (x / z).  [para(26(a,1),14(a,1)),flip(a)].
247 (x / y) / ((x / z) / (y / z)) = (x / y) / (x / z).  [back_rewrite(34),rewrite([203(10)])].
352 (c1 * ((c1 / c2) / (c1 / c3))) / (c1 / c2) != c1 / ((c1 / c2) / (c3 / c2)).  [para(26(a,1),136(a,1,1,2)),rewrite([39(8)])].
353 (c1 * ((c1 / c2) / (c1 / c3))) / (c1 / c2) != c1 / ((c1 / c3) / (c2 / c3)).  [para(5(a,1),352(a,2,2))].
661 (x * (y / z)) / y = x / (y / (y / z)).  [para(93(a,1),32(a,1,1)),flip(a)].
675 c1 / ((c1 / c3) / (c2 / c3)) != c1 / ((c1 / c2) / ((c1 / c2) / (c1 / c3))).  [back_rewrite(353),rewrite([661(13)]),flip(a)].
1150 x / ((y / z) / ((y / z) / (y / u))) = x / ((y / u) / (z / u)).  [para(33(a,1),37(a,1,1,2)),rewrite([2(2),247(6)])].
1151 $F.  [resolve(1150,a,675,a(flip))].

============================== end of proof ==========================
```

◀

Proof of Lem. 34. We first show the equivalences in $f \cdot g \equiv f' \cdot g \equiv f' \cdot g'$ in turn. In each case we only show the CRA equation arising for the left components, from which the CRA equation for the right components follows by symmetry.

```
============================== PROOF =================================

% Proof 1 at 181.68 (+ 1.21) seconds.
% Length of proof is 71.
% Level of proof is 14.
% Maximum clause weight is 29.000.
% Given clauses 1229.

1 (a * (e / b)) / (f * (b / e)) = (c * (e / d)) / (f * (d / e)) # label(non_clause) # label(goal).  [goal].
2 x / 1 = x.  [assumption].
3 x / x = 1.  [assumption].
4 1 / x = 1.  [assumption].
5 (x / y) / (z / y) = (x / z) / (y / z).  [assumption].
6 (x / y) / x = 1.  [assumption].
7 x / (x / y) = y / (y / x).  [assumption].
8 (x * y) / x = y.  [assumption].
9 x / (x * y) = 1.  [assumption].
10 a / b = c / d.  [assumption].
11 c / d = a / b.  [copy(10),flip(a)].
12 b / a = d / c.  [assumption].
13 d / c = b / a.  [copy(12),flip(a)].
14 (c * (e / d)) / (f * (d / e)) != (a * (e / b)) / (f * (b / e)).  [deny(1)].
15 ((x / y) / (z / y)) / (u / (y / z)) = ((x / z) / u) / ((y / z) / u).  [para(5(a,1),5(a,1,1))].
16 (x / (y / z)) / ((u / z) / (y / z)) = (x / (u / z)) / ((y / u) / (z / u)).  [para(5(a,1),5(a,1,2)),flip(a)].
17 ((x / y) / z) / (x / z) = 1.  [para(6(a,1),5(a,1,1)),rewrite([4(3)]),flip(a)].
18 (x / (y / z)) / (y / (y / z)) = x / y.  [para(6(a,1),5(a,1,2)),rewrite([2(3)]),flip(a)].
20 (x / (x / y)) / (z / (y / x)) = (y / z) / ((y / x) / z).  [para(7(a,1),5(a,1,1))].
21 (x / (y / z)) / (z / (z / y)) = x / y.  [para(7(a,1),5(a,1,2)),rewrite([6(8),2(8)])].
26 x / (y / (y / x)) = x / y.  [para(7(a,1),7(a,1,2)),rewrite([6(6),2(6)])].
28 ((x * y) / z) / (x / z) = y / (z / x).  [para(8(a,1),5(a,1,1)),flip(a)].
29 (x / y) / z = x / (y * z).  [para(8(a,1),5(a,1,2)),rewrite([9(6),2(6)])].
30 (x * y) / y = x.  [para(8(a,1),7(a,1,2)),rewrite([9(4),2(4)])].
31 ((c * (e / d)) / f) / (d / e) != ((a * (e / b)) / f) / (b / e).  [back_rewrite(14),rewrite([29(11,R),29(22,R)])].
34 (a / b) / c = 1.  [para(11(a,1),6(a,1,1))].
35 d / (b / a) = c / (a / b).  [para(11(a,1),7(a,1,2)),rewrite([13(9)]),flip(a)].
37 (b / a) / d = 1.  [para(13(a,1),6(a,1,1))].
55 (x / y) / z = (x / z) / y.  [para(6(a,1),15(a,2,2)),rewrite([21(6),2(6)])].
91 ((x * y) / z) / (y / z) = x / (z / y).  [para(30(a,1),5(a,1,1)),flip(a)].
92 (x / y) / z = x / (z * y).  [para(30(a,1),5(a,1,2)),rewrite([29(6,R),6(6),2(6)])].
98 (x / (a / b)) / (c / (a / b)) = x / c.  [para(34(a,1),5(a,1,2)),rewrite([2(4)]),flip(a)].
```

```
131 (x / ((y * z) / u)) / ((y / u) / ((y * z) / u)) = (x / (y / u)) / (z / (u / y)).  [para(8(a,1),16(a,2,2,1))].
202 (x / b) / c = (x / a) / d.  [para(37(a,1),15(a,2,2)),rewrite([35(11),98(12),2(10)])].
206 ((x / y) / z) / x = 1.  [para(6(a,1),17(a,1,2)),rewrite([2(5)])].
222 (x / y) / ((x * z) / y) = 1.  [para(30(a,1),17(a,1,1,1))].
234 (x / (y / z)) / (u / (z / y)) = x / ((y * u) / z).  [back_rewrite(131),rewrite([222(7),2(5)]),flip(a)].
240 (x / y) / (x / (y / z)) = 1.  [para(7(a,1),206(a,1,1)),rewrite([55(4)])].
281 (x * (y / z)) / y = x / (y / (y / z)).  [para(30(a,1),18(a,1,1)),flip(a)].
413 (x / y) / (z / y) = x / (z * (y / z)).  [para(29(a,1),5(a,1)),flip(a)].
416 (x / (y * z)) / (x / y) = 1.  [para(29(a,1),6(a,1,1))].
418 (x * y) / ((x * y) / z) = z / ((z / x) / y).  [para(29(a,2),7(a,1,2)),flip(a)].
420 (x * y) / (x * z) = y / z.  [para(8(a,1),29(a,1,1)),flip(a)].
463 (x / (y * z)) / (x / z) = 1.  [para(29(a,1),17(a,1,1))].
502 d / b = c / a.  [para(35(a,1),18(a,1,1)),rewrite([7(10),18(11)]),flip(a)].
505 (x / d) / (b / d) = (x / b) / (c / a).  [para(502(a,1),5(a,1,2)),flip(a)].
588 b / d = a / c.  [para(37(a,1),20(a,2,2)),rewrite([35(10),98(11),2(8)]),flip(a)].
673 (x / d) / (a / c) = (x / b) / (c / a).  [back_rewrite(505),rewrite([588(5)])].
776 ((x * y) / z) / x = y / z.  [para(8(a,1),55(a,1,1)),flip(a)].
1074 (x * y) / (z * y) = x / z.  [para(30(a,1),92(a,1,1)),flip(a)].
1376 (b * x) / (a * d) = x / c.  [para(8(a,1),202(a,1,1)),rewrite([29(8)]),flip(a)].
1532 (a / c) / ((b * x) / d) = 1.  [para(588(a,1),222(a,1,1))].
1593 x / ((x * y) / (y / z)) = 1.  [para(30(a,1),240(a,1,1))].
2040 x / ((x * (y * z)) / y) = 1.  [para(30(a,1),416(a,1,1))].
10409 (x / c) / ((b * x) / d) = 1.  [para(1376(a,1),463(a,1,1))].
13429 x / ((x * y) / (z / (z / y))) = 1.  [para(7(a,1),1593(a,1,2,2))].
20049 x / ((b * (c * x)) / d) = 1.  [para(8(a,1),10409(a,1,1))].
22693 (d * x) / (b * (c * x)) = 1.  [para(20049(a,1),28(a,2)),rewrite([29(13,R),502(10),29(13,R),6(12),4(9),2(9)])].
23285 (x * y) / ((b * (c * y)) / d) = x / (((b * (c * y)) / d) / y).  [para(20049(a,1),91(a,1,2)),rewrite([2(10)])].
23312 d * x = (b * (c * x)) / (((b * (c * x)) / d) / x).  [para(22693(a,1),7(a,1,2)),rewrite([2(4),29(13,R)])].
68851 x / ((y / z) * (u / (z / y))) = x / ((y * u) / z).  [para(234(a,1),29(a,1)),flip(a)].
116832 x / ((b * y) / d) = x / ((a * y) / c).  [para(1532(a,1),413(a,1,2)),rewrite([2(7),673(16),8(12),68851(14)])].
116842 x / ((y * (z * u)) / z) = x / (y * u).  [para(2040(a,1),413(a,1,2)),rewrite([2(6),92(8),420(8),8(6)])].
117453 x / (((b * (c * y)) / d) / y) = x / a.  [back_rewrite(23285),rewrite([116832(8),116842(8),1074(4)]),flip(a)].
118025 d * x = (b * (c * x)) / a.  [back_rewrite(23312),rewrite([117453(14)])].
118050 (b * (c * x)) / (a * x) = d.  [para(118025(a,1),30(a,1,1)),rewrite([29(7)])].
118354 (b * (c * x)) / d = a * x.  [para(118050(a,1),7(a,1,2)),rewrite([29(15,R),29(15,R),202(14),8(12),6(11),2(10)])].
118800 (c * x) / d = (a * x) / b.  [para(118354(a,1),776(a,1,1)),flip(a)].
122513 (x * (y / x)) / (x / y) = y.  [para(3(a,1),418(a,2,2)),rewrite([281(5),26(5),2(6)])].
123399 x * (y / z) = (x * y) / (z / (z / y)).  [para(13429(a,1),122513(a,1,2)),rewrite([776(5),26(3),2(4)])].
123605 $F.  [back_rewrite(31),rewrite([123399(5),55(11),55(15),18(15),55(7),118800(5),123399(12),55(18),55(22),18(22),55(14)]),xx(a)].

============================== end of proof ==========================
```

and

```
============================== PROOF =================================

% Proof 1 at 168.96 (+ 1.07) seconds.
% Length of proof is 111.
% Level of proof is 20.
% Maximum clause weight is 43.000.
% Given clauses 1433.

1 (e * (a / f)) / (b * (f / a)) = (e * (c / f)) / (d * (f / c)) # label(non_clause) # label(goal).  [goal].
2 x / 1 = x.  [assumption].
3 x / x = 1.  [assumption].
4 1 / x = 1.  [assumption].
5 (x / y) / (z / y) = (x / z) / (y / z).  [assumption].
6 (x / y) / x = 1.  [assumption].
7 x / (x / y) = y / (y / x).  [assumption].
8 (x * y) / x = y.  [assumption].
9 x / (x * y) = 1.  [assumption].
10 a / b = c / d.  [assumption].
11 c / d = a / b.  [copy(10),flip(a)].
12 b / a = d / c.  [assumption].
13 d / c = b / a.  [copy(12),flip(a)].
14 (e * (c / f)) / (d * (f / c)) != (e * (a / f)) / (b * (f / a)).  [deny(1)].
15 ((x / y) / (z / y)) / (u / (y / z)) = ((x / z) / u) / ((y / z) / u).  [para(5(a,1),5(a,1,1))].
16 (x / (y / z)) / ((u / z) / (y / z)) = (x / (u / z)) / ((y / u) / (z / u)).  [para(5(a,1),5(a,1,2)),flip(a)].
17 ((x / y) / z) / (x / z) = 1.  [para(6(a,1),5(a,1,1)),rewrite([4(3)]),flip(a)].
18 (x / (y / z)) / (y / (y / z)) = x / y.  [para(6(a,1),5(a,1,2)),rewrite([2(3)]),flip(a)].
20 (x / (x / y)) / (z / (y / x)) = (y / z) / ((y / x) / z).  [para(7(a,1),5(a,1,1))].
21 (x / (y / z)) / (z / (z / y)) = x / y.  [para(7(a,1),5(a,1,2)),rewrite([6(8),2(8)])].
25 x / (x / (x / y)) = x / y.  [para(6(a,1),7(a,1,2)),rewrite([2(3)]),flip(a)].
26 x / (y / (y / x)) = x / y.  [para(7(a,1),7(a,1,2)),rewrite([6(6),2(6)])].
28 ((x * y) / z) / (x / z) = y / (z / x).  [para(8(a,1),5(a,1,1)),flip(a)].
29 (x / y) / z = x / (y * z).  [para(8(a,1),5(a,1,2)),rewrite([9(6),2(6)])].
30 (x * y) / y = x.  [para(8(a,1),7(a,1,2)),rewrite([9(4),2(4)])].
33 (a / b) / c = 1.  [para(11(a,1),6(a,1,1))].
34 d / (b / a) = c / (a / b).  [para(11(a,1),7(a,1,2)),rewrite([13(9)]),flip(a)].
36 (b / a) / d = 1.  [para(13(a,1),6(a,1,1))].
54 (x / y) / z = (x / z) / y.  [para(6(a,1),15(a,2,2)),rewrite([21(6),2(6)])].
64 ((x / (y / z)) / u) / ((z / (z / y)) / u) = (x / y) / (u / (y / (y / z))).  [para(7(a,1),15(a,2,2,1)),rewrite([6(3),2(3)]),flip(a)].
65 (((x * y) / z) / u) / ((x / z) / u) = (y / (z / x)) / (u / (x / z)).  [para(8(a,1),15(a,1,1,1)),flip(a)].
90 ((x * y) / z) / (y / z) = x / (z / y).  [para(30(a,1),5(a,1,1)),flip(a)].
91 (x / y) / z = x / (z * y).  [para(30(a,1),5(a,1,2)),rewrite([29(6,R),6(6),2(6)])].
97 (x / (a / b)) / (c / (a / b)) = x / c.  [para(33(a,1),5(a,1,2)),rewrite([2(4)]),flip(a)].
130 (x / ((y * z) / u)) / ((y / u) / ((y * z) / u)) = (x / (y / u)) / (z / (u / y)).  [para(8(a,1),16(a,2,2,1))].
173 (x / ((y * z) / u)) / ((z / u) / ((y * z) / u)) = (x / (z / u)) / (y / (u / z)).  [para(30(a,1),16(a,2,2,1))].
201 (x / b) / c = (x / a) / d.  [para(36(a,1),15(a,2,2)),rewrite([34(11),97(12),2(10)])].
205 ((x / y) / z) / x = 1.  [para(6(a,1),17(a,1,2)),rewrite([2(5)])].
209 (x / y) / ((z * x) / y) = 1.  [para(8(a,1),17(a,1,1,1))].
221 (x / y) / ((x * z) / y) = 1.  [para(30(a,1),17(a,1,1,1))].
232 (x / (y / z)) / (u / (z / y)) = x / ((u * y) / z).  [back_rewrite(173),rewrite([209(7),2(5)]),flip(a)].
233 (x / (y / z)) / (u / (z / y)) = x / ((y * u) / z).  [back_rewrite(130),rewrite([221(7),2(5)]),flip(a)].
239 (x / y) / (x / (y / z)) = 1.  [para(7(a,1),205(a,1,1)),rewrite([54(4)])].
240 (x / y) / (z * x) = 1.  [para(8(a,1),205(a,1,1,1))].
271 ((x / y) * z) / x = z / (x / (x / y)).  [para(8(a,1),18(a,1,1)),flip(a)].
280 (x * (y / z)) / y = x / (y / (y / z)).  [para(30(a,1),18(a,1,1)),flip(a)].
295 (x / (y / z)) / ((u * y) / (y / z)) = x / (u * y).  [para(240(a,1),5(a,1,2)),rewrite([2(4)]),flip(a)].
298 (x / (x / y)) / (z * y) = 1.  [para(7(a,1),240(a,1,1))].
412 (x / y) / (z / y) = x / (z * (y / z)).  [para(29(a,1),5(a,1)),flip(a)].
415 (x / (y * z)) / (x / y) = 1.  [para(29(a,1),6(a,1,1))].
417 (x * y) / ((x * y) / z) = z / ((z / x) / y).  [para(29(a,2),7(a,1,2)),flip(a)].
419 (x * y) / (x * z) = y / z.  [para(8(a,1),29(a,1,1)),flip(a)].
```

```
440 (x * y) / (y * z) = x / z.  [para(30(a,1),29(a,1,1)),flip(a)].
462 (x / (y * z)) / (x / z) = 1.  [para(29(a,1),17(a,1,1))].
501 d / b = c / a.  [para(34(a,1),18(a,1,1)),rewrite([7(10),18(11)]),flip(a)].
504 (x / d) / (b / d) = (x / b) / (c / a).  [para(501(a,1),5(a,1,2)),flip(a)].
506 d / (c / a) = b / (b / d).  [para(501(a,1),7(a,1,2))].
514 (x / (b / (b / d))) / (c / a) = x / d.  [para(501(a,1),18(a,1,1,2)),rewrite([501(8),506(9),54(10)])].
587 b / d = a / c.  [para(36(a,1),20(a,2,2)),rewrite([34(10),97(11),2(8)]),flip(a)].
668 (x / (b / (a / c))) / (c / a) = x / d.  [back_rewrite(514),rewrite([587(4)])].
672 (x / d) / (a / c) = (x / b) / (c / a).  [back_rewrite(504),rewrite([587(5)])].
687 ((a / c) / x) / (b / x) = 1.  [para(587(a,1),17(a,1,1,1))].
775 ((x * y) / z) / x = y / z.  [para(8(a,1),54(a,1,1)),flip(a)].
789 ((x * y) / z) / y = x / z.  [para(30(a,1),54(a,1,1)),flip(a)].
1051 ((x * y) * z) / (y * x) = z.  [para(91(a,2),8(a,1)),rewrite([29(4)])].
1052 (x * y) / (z * x) = y / z.  [para(8(a,1),91(a,1,1)),flip(a)].
1073 (x * y) / (z * y) = x / z.  [para(30(a,1),91(a,1,1)),flip(a)].
1372 ((x / b) / y) / (c / y) = ((x / a) / d) / (y / c).  [para(201(a,1),5(a,1,1)),flip(a)].
1375 (b * x) / (a * d) = x / c.  [para(8(a,1),201(a,1,1)),rewrite([29(8)]),flip(a)].
1531 (a / c) / ((b * x) / d) = 1.  [para(587(a,1),221(a,1,1))].
1592 x / ((x * y) / (y / z)) = 1.  [para(30(a,1),239(a,1,1))].
1746 (x / (y / (y / z))) / ((u * z) / (y / (y / z))) = x / (u * z).  [para(298(a,1),5(a,1,2)),rewrite([2(4)]),flip(a)].
2039 x / ((x * (y * z)) / y) = 1.  [para(30(a,1),415(a,1,1))].
5345 (c * x) / (a * d) = x / b.  [para(775(a,1),201(a,1)),rewrite([29(8)]),flip(a)].
8867 x * y = y * x.  [para(1051(a,1),7(a,1,2)),rewrite([30(3),29(6,R),440(5),3(3),4(4),2(4)])].
10408 (x / c) / ((b * x) / d) = 1.  [para(1375(a,1),462(a,1,1))].
13428 x / ((x * y) / (z / (z / y))) = 1.  [para(7(a,1),1592(a,1,2,2))].
14308 (x / (y / (a / c))) / (b / ((a / c) / y)) = (x / y) / (b / (a / c)).  [para(687(a,1),64(a,1,2)),rewrite([2(14),21(25)])].
14799 (x / (b / y)) / (c / (y / b)) = (x / (a / y)) / (d / (y / a)).  [para(201(a,1),65(a,1,1)),rewrite([201(9),65(10)]),flip(a)].
17820 (x / b) / ((c * x) / d) = 1.  [para(5345(a,1),462(a,1,1))].
20048 x / ((b * (c * x)) / d) = 1.  [para(8(a,1),10408(a,1,1))].
21589 x / ((c * (b * x)) / d) = 1.  [para(8(a,1),17820(a,1,1))].
22692 (d * x) / (b * (c * x)) = 1.  [para(20048(a,1),28(a,2)),rewrite([29(13,R),501(10),29(13,R),6(12),4(9),2(9)])].
23284 (x * y) / ((b * (c * y)) / d) = x / (((b * (c * y)) / d) / y).  [para(20048(a,1),90(a,1,2)),rewrite([2(10)])].
23294 (x * d) / (c * (b * x)) = 1.  [para(21589(a,1),90(a,2)),rewrite([29(13,R),13(10),29(13,R),6(12),4(9),2(9)])].
23311 d * x = (b * (c * x)) / (((b * (c * x)) / d) / x).  [para(22692(a,1),7(a,1,2)),rewrite([2(4),29(13,R)])].
23677 (e * (c / f)) / ((b * (c * (f / c))) / (((b * (c * (f / c))) / d) / (f / c))) != (e * (a / f)) / (b * (f / a)).
  [back_rewrite(14),rewrite([23311(10)])].
23855 (c * (b * x)) / ((c * (b * x)) / (x * d)) = x * d.  [para(23294(a,1),7(a,1,2)),rewrite([2(4)]),flip(a)].
68550 x / ((y / z) * (u / (z / y))) = x / ((y * u) / z).  [para(233(a,1),29(a,1)),flip(a)].
68755 ((e * (c / f)) / ((c * (f / c)) / (a / b))) / (b / (((b * (c * (f / c))) / d) / (f / c))) != (e * (a / f)) / (b * (f / a)).
  [para(233(a,2),23677(a,1)),rewrite([54(41),54(37),8(35),789(37),11(29),54(31)])].
69343 (((e * (c / f)) / (c / (a / b))) / (f / c)) / (b / (((b * (c * (f / c))) / d) / (f / c))) != (e * (a / f)) / (b * (f / a)).
  [para(233(a,2),68755(a,1,1)),rewrite([201(19),3(17),4(17),2(16)])].
100851 x / ((b * y) / d) = x / ((a * y) / c).  [para(1531(a,1),412(a,1,2)),rewrite([2(7),672(16),8(12),68550(14)])].
100852 x / ((y * (z * u)) / z) = x / (y * u).  [para(2039(a,1),412(a,1,2)),rewrite([2(6),91(8),419(8),8(6)])].
100971 x / (((b * (c * y)) / d) / y) = x / a.  [back_rewrite(23284),rewrite([100851(8),100852(8),1073(4)]),flip(a)].
101129 (e * (c / f)) / ((b * (c * (f / c))) / a) != (e * (a / f)) / (b * (f / a)).
  [back_rewrite(69343),rewrite([100971(30),54(19),54(15),14799(15),3(8),2(7),3(9),2(8),29(11),23311(10),100971(26)])].
101286 d * x = (b * (c * x)) / a.  [back_rewrite(23311),rewrite([100971(14)])].
101312 (b * (c * x)) / (a * x) = d.  [para(101286(a,1),30(a,1,1)),rewrite([29(7)])].
101351 (b * (c * x)) / d = a * x.  [para(101312(a,1),7(a,1,2)),rewrite([29(15,R),29(15,R),201(14),8(12),6(11),2(10)])].
101507 (c * x) / d = (a * x) / b.  [para(101351(a,1),775(a,1,1)),flip(a)].
101823 (c * x) / (y * d) = ((a * x) / b) / y.  [para(101507(a,1),91(a,1,1)),flip(a)].
101858 (c * (b * x)) / a = x * d.  [back_rewrite(23855),rewrite([101823(11),29(11),30(11)])].
102173 (c * (b * x)) / (a * x) = d.  [para(101858(a,2),8(a,1,1)),rewrite([29(7)])].
102222 (x * (y / x)) / (x / y) = y.  [para(3(a,1),417(a,2,2)),rewrite([280(5),26(5),2(6)])].
102636 x * (y / z) = (x * y) / (z / (z / y)).  [para(13428(a,1),102222(a,1,2)),rewrite([775(5),26(3),2(4)])].
102638 (x / y) * z = (x * z) / (x / (x / y)).  [para(271(a,1),102222(a,1,1,2)),rewrite([102636(4),29(11,R),54(12),18(12)]),flip(a)].
102639 (b * (c * (a * x))) / a = b * (c * x).
  [para(101312(a,1),102222(a,1,1,2)),rewrite([8867(4),101286(4),29(15,R),29(15,R),201(14),8(12),6(11),2(10)])].
102644 c * (b * x) = b * (c * x).
  [para(102173(a,1),102222(a,1,1,2)),rewrite([8867(4),101286(4),102639(8),29(11,R),29(11,R),54(10),201(10),8(8),6(7),2(6)]),flip(a)].
102735 ((c * e) / (c / (c / f))) / ((b * (c * f)) / ((a * c) / (c / f))) != ((a * e) / b) / f.
  [back_rewrite(101129),rewrite([102636(5),8867(3),7(8),102636(15),102636(20),29(28,R),6(26),
   4(22),2(21),54(22),29(22),102636(21),6(25),2(22),102636(28),8867(26),7(31),102636(37),1746(42),29(30,R)])].
104297 (c * e) / ((b * (c * (c * f))) / (a * c)) != ((a * e) / b) / f.
  [para(29(a,1),102735(a,1)),rewrite([102636(22),102638(14),102644(10),25(17),29(34,R),54(30),
   29(34,R),54(32),201(28),8(26),11(24),54(30),1372(30),3(24),4(24),4(26),2(23),295(22)])].
104300 ((c * e) / (b / ((a / c) / f))) / ((c * f) / a) != ((a * e) / b) / f.
  [para(232(a,2),104297(a,1)),rewrite([1052(12),29(19,R),30(15),29(15,R),54(17)])].
104304 $F.  [para(233(a,2),104300(a,1)),rewrite([54(15),54(21),14308(21),54(9),54(15),668(15),54(7),101507(5)]),xx(a)].

============================== end of proof ==========================
```

Commutativity of product is shown by:

```
============================== PROOF =================================

% Proof 1 at 36.46 (+ 0.25) seconds.
% Length of proof is 51.
% Level of proof is 15.
% Maximum clause weight is 27.000.
% Given clauses 384.

1 (x * (z / y)) / (u * (y / z)) = (z * (x / u)) / (y * (u / x)) # label(non_clause) # label(goal).  [goal].
2 x / 1 = x.  [assumption].
3 x / x = 1.  [assumption].
4 1 / x = 1.  [assumption].
5 (x / y) / (z / y) = (x / z) / (y / z).  [assumption].
6 (x / y) / x = 1.  [assumption].
7 x / (x / y) = y / (y / x).  [assumption].
8 (x * y) / x = y.  [assumption].
9 x / (x * y) = 1.  [assumption].
10 (c2 * (c1 / c4)) / (c3 * (c4 / c1)) != (c1 * (c2 / c3)) / (c4 * (c3 / c2)).  [deny(1)].
11 (c1 * (c2 / c3)) / (c4 * (c3 / c2)) != (c2 * (c1 / c4)) / (c3 * (c4 / c1)).  [copy(10),flip(a)].
12 ((x / y) / (z / y)) / (u / (y / z)) = ((x / z) / u) / ((y / z) / u).  [para(5(a,1),5(a,1,1))].
15 (x / (y / z)) / (y / (y / z)) = x / y.  [para(6(a,1),5(a,1,2)),rewrite([2(3)]),flip(a)].
16 ((x / y) / (z / y)) / (x / z) = 1.  [para(5(a,1),6(a,1,1))].
18 (x / (y / z)) / (z / (z / y)) = x / y.  [para(7(a,1),5(a,1,2)),rewrite([6(8),2(8)])].
23 x / (y / (y / x)) = x / y.  [para(7(a,1),7(a,1,2)),rewrite([6(6),2(6)])].
26 (x / y) / z = x / (y * z).  [para(8(a,1),5(a,1,2)),rewrite([9(6),2(6)])].
27 (x * y) / y = x.  [para(8(a,1),7(a,1,2)),rewrite([9(4),2(4)])].
28 ((c2 * (c1 / c4)) / c3) / (c4 / c1) != ((c1 * (c2 / c3)) / c4) / (c3 / c2).
  [back_rewrite(11),rewrite([26(11,R),26(22,R)]),flip(a)].
46 (x / y) / z = (x / z) / y.  [para(6(a,1),12(a,2,2)),rewrite([18(6),2(6)])].
```

```
80 (x / y) / z = x / (z * y).  [para(27(a,1),5(a,1,2)),rewrite([26(6,R),6(6),2(6)])].
84 ((x / y) / (z / y)) / u = (x / z) / (u * (y / z)).  [para(27(a,1),12(a,1,2)),rewrite([26(12,R),6(12),2(10)])].
268 ((x / y) * z) / x = z / (x / (x / y)).  [para(8(a,1),15(a,1,1)),flip(a)].
316 (x * y) / (x * z) = y / z.  [para(8(a,1),26(a,1,1)),flip(a)].
337 (x * y) / (y * z) = x / z.  [para(27(a,1),26(a,1,1)),flip(a)].
387 ((x * y) / z) / x = y / z.  [para(8(a,1),46(a,1,1)),flip(a)].
399 ((x * y) / z) / y = x / z.  [para(27(a,1),46(a,1,1)),flip(a)].
444 ((x * y) * z) / (y * x) = z.  [para(80(a,2),8(a,1)),rewrite([26(4)])].
445 (x * y) / (z * x) = y / z.  [para(8(a,1),80(a,1,1)),flip(a)].
558 (x / (y * (z / y))) / (x / z) = 1.  [para(26(a,1),16(a,1,1))].
1405 (((x * y) / z) / u) / (x / u) = (y / z) / (u / x).  [para(387(a,1),5(a,1,1)),flip(a)].
1491 x * y = y * x.  [para(444(a,1),7(a,1,2)),rewrite([27(3),26(6,R),337(5),3(3),4(4),2(4)])].
9149 (x * y) / (z * ((y * x) / z)) = 1.  [para(444(a,1),558(a,1,2)),rewrite([399(7)])].
24090 (x * y) / ((y * z) * (x / z)) = 1.  [para(316(a,1),9149(a,1,2,2))].
24092 (x * y) / ((z * y) * (x / z)) = 1.  [para(445(a,1),9149(a,1,2,2))].
28378 (x / (y / (y / z))) / (u * (z / y)) = (x / z) / u.  [para(23(a,1),84(a,2,2,2)),rewrite([46(4),3(4),2(3)]),flip(a)].
34509 ((x * y) * z) / ((z * x) * y) = 1.  [para(8(a,1),24090(a,1,2,2))].
34562 ((x * y) * z) / ((y * z) * x) = 1.  [para(27(a,1),24092(a,1,2,2))].
36164 (x * y) * z = (z * x) * y.  [para(34509(a,1),7(a,1,2)),rewrite([2(4),34562(9),2(6)])].
36219 ((x * y) * z) / y = z * x.  [para(36164(a,2),27(a,1,1))].
36252 (x * y) * z = x * (y * z).  [para(36164(a,1),1491(a,1))].
36609 (x * (y * z)) / y = z * x.  [back_rewrite(36219),rewrite([36252(2)])].
37712 x * (y * z) = y * (x * z).  [para(1491(a,1),36252(a,1,1)),rewrite([36252(2)])].
37886 (((c2 / c3) * (x * c1)) / (c3 / c2)) / (c4 * x) != ((c2 * (c1 / c4)) / c3) / (c4 / c1).
  [para(36609(a,2),28(a,2,1,1)),rewrite([80(20),46(24)]),flip(a)].
37913 (x * (y * z)) / y = x * z.  [para(36609(a,2),1491(a,1))].
38342 ((c2 / c3) * (x * c1)) / (c4 * ((c3 / c2) * x)) != ((c2 * (c1 / c4)) / c3) / (c4 / c1).  [para(26(a,1),37886(a,1)),rewrite([37712(12)])].
39055 ((c2 / c3) * (x * c1)) / (c4 * (x * (c3 / c2))) != ((c2 * (c1 / c4)) / c3) / (c4 / c1).  [para(1491(a,1),38342(a,1,2,2))].
39728 (x / y) * z = (x * z) / (x / (x / y)).  [para(268(a,1),37913(a,1)),flip(a)].
39756 (((c2 * (x * c1)) / c3) / c4) / x != ((c2 * (c1 / c4)) / c3) / (c4 / c1).
  [back_rewrite(39055),rewrite([39728(6),26(17,R),46(12),28378(17),46(8)])].
40853 (((c1 * (c2 * x)) / c3) / c4) / x != ((c2 * (c1 / c4)) / c3) / (c4 / c1).
  [para(1491(a,1),39756(a,1,1,1,1,2)),rewrite([37712(4)])].
40854 $F.  [resolve(40853,a,1405,a)].

============================== end of proof ==========================
```

◀

Proof of $\equiv = \equiv'$ as claimed in Remark 36. We first prove that $\frac{a_1}{b_1} \equiv \frac{a_2}{b_2}$ entails $\frac{a_1}{b_1} \equiv' \frac{a_2}{b_2}$, and then show the reverse implication.

Unfolding the definition of $\equiv$ yields that for the former it suffices to show that $\frac{a}{b} \equiv' \frac{a/b}{b/a}$ and that $\equiv'$ is an equivalence relation. The first follows immediately from (5) and (6), whereas for the second only transitivity is non-trivial. It boils down to showing that $\frac{a_1}{b_1} \equiv' \frac{a_2}{b_2}$ and $\frac{a_2}{b_2} \equiv' \frac{a_3}{b_3}$ entail $\frac{a_1}{b_1} \equiv' \frac{a_3}{b_3}$. Unfolding the definition, we must show $a_1/a_2 = b_1/b_2$, $a_2/a_1 = b_2/b_1$, $a_2/a_3 = b_2/b_3$, and $a_3/a_2 = b_3/b_2$, then $a_1/a_3 = b_1/b_3$ and $a_3/a_1 = b_3/b_1$. The Prover9 proof below shows the first of these two, with the other following by symmetry.

```
============================== PROOF =================================

% Proof 1 at 118.06 (+ 1.00) seconds.
% Length of proof is 65.
% Level of proof is 13.
% Maximum clause weight is 49.000.
% Given clauses 1126.

1 a1 / a3 = b1 / b3 # label(non_clause) # label(goal).  [goal].
2 x / 1 = x.  [assumption].
3 x / x = 1.  [assumption].
4 1 / x = 1.  [assumption].
5 (x / y) / (z / y) = (x / z) / (y / z).  [assumption].
6 (x / y) / x = 1.  [assumption].
7 x / (x / y) = y / (y / x).  [assumption].
8 a1 / a2 = b1 / b2.  [assumption].
9 b1 / b2 = a1 / a2.  [copy(8),flip(a)].
10 a2 / a3 = b2 / b3.  [assumption].
11 b2 / b3 = a2 / a3.  [copy(10),flip(a)].
12 a2 / a1 = b2 / b1.  [assumption].
13 b2 / b1 = a2 / a1.  [copy(12),flip(a)].
14 a3 / a2 = b3 / b2.  [assumption].
15 b3 / b2 = a3 / a2.  [copy(14),flip(a)].
16 b1 / b3 != a1 / a3.  [deny(1)].
17 ((x / y) / (z / y)) / (u / (y / z)) = ((x / z) / u) / ((y / z) / u).  [para(5(a,1),5(a,1,1))].
18 (x / (y / z)) / ((u / z) / (y / z)) = (x / (u / z)) / ((y / u) / (z / u)).  [para(5(a,1),5(a,1,2)),flip(a)].
19 ((x / y) / z) / (x / z) = 1.  [para(6(a,1),5(a,1,1)),rewrite([4(3)]),flip(a)].
20 (x / (y / z)) / (y / (y / z)) = x / y.  [para(6(a,1),5(a,1,2)),rewrite([2(3)]),flip(a)].
22 (x / (x / y)) / (z / (y / x)) = (y / z) / ((y / x) / z).  [para(7(a,1),5(a,1,1))].
23 (x / (y / z)) / (z / (z / y)) = x / y.  [para(7(a,1),5(a,1,2)),rewrite([6(8),2(8)])].
27 x / (x / (x / y)) = x / y.  [para(6(a,1),7(a,1,2)),rewrite([2(3)]),flip(a)].
28 x / (y / (y / x)) = x / y.  [para(7(a,1),7(a,1,2)),rewrite([6(6),2(6)])].
31 (a1 / a2) / b1 = 1.  [para(9(a,1),6(a,1,1))].
32 b1 / (a1 / a2) = b2 / (a2 / a1).  [para(9(a,1),7(a,1,2)),rewrite([13(9)])].
36 b3 / (a3 / a2) = b2 / (a2 / a3).  [para(11(a,1),7(a,1,2)),rewrite([15(9)]),flip(a)].
40 (a3 / a2) / b3 = 1.  [para(15(a,1),6(a,1,1))].
58 (x / y) / z = (x / z) / y.  [para(6(a,1),17(a,2,2)),rewrite([23(6),2(6)])].
78 ((x / (y / z)) / ((u / y) / (z / y))) / ((w / (y / z)) / ((u / y) / (z / y))) =
  ((x / w) / ((y / z) / w)) / (((u / z) / w) / ((y / z) / w)).  [para(17(a,1),17(a,1,2)),flip(a)].
107 ((x / a3) / (a2 / a3)) / (b2 / (a2 / a3)) = (x / a2) / b3.  [para(40(a,1),17(a,2,2)),rewrite([36(11),2(18)])].
142 (x / (y / (z / u))) / ((z / (z / u)) / (y / (z / u))) = (x / (u / (u / z))) / (y / z).  [para(7(a,1),18(a,2,1,2)),rewrite([6(15),2(15)])].
145 (x / (y / (y / z))) / (z / y) = x / z.  [para(7(a,1),18(a,2,2)),rewrite([20(8),20(12)])].
214 ((x / y) / z) / x = 1.  [para(6(a,1),19(a,1,2)),rewrite([2(5)])].
233 ((a3 / a2) / x) / b3 = 1.  [para(40(a,1),19(a,1,2)),rewrite([2(8)])].
263 (x / (y / ((z / u) / w))) / ((z / ((z / u) / w)) / (y / ((z / u) / w))) = (x / (z / ((z / u) / w))) / (y / z).  [para(214(a,1),18(a,2,2,2)),rewrite([2(19)])].
284 ((x / ((a3 / a2) / y)) / z) / ((b3 / ((a3 / a2) / y)) / z) = (x / b3) / (z / (b3 / ((a3 / a2) / y))).  [para(233(a,1),17(a,1,1,2)),rewrite([2(4)]),flip(a)].
```

```
293 (x / (a2 / a3)) / (b2 / (a2 / a3)) = x / b2.  [para(11(a,1),20(a,1,1,2)),rewrite([11(8)])].
294 (x / (a2 / a1)) / (b2 / (a2 / a1)) = x / b2.  [para(13(a,1),20(a,1,1,2)),rewrite([13(8)])].
314 ((x / y) / z) / (x / (y / u)) = 1.  [para(20(a,1),214(a,1,1,1))].
316 (x / b2) / a3 = (x / a2) / b3.  [back_rewrite(107),rewrite([293(12),58(4)])].
443 (x / y) / (y / x) = x / y.  [para(28(a,1),20(a,1,1)),rewrite([27(4)])].
453 b1 / a1 = b2 / a2.  [para(32(a,1),20(a,1,1)),rewrite([7(10),20(11)]),flip(a)].
458 b1 / (b2 / a2) = a1 / (a1 / b1).  [para(453(a,1),7(a,1,2))].
511 a1 / b1 = a2 / b2.  [para(31(a,1),22(a,2,2)),rewrite([32(10),294(11),2(8)]),flip(a)].
514 a3 / b3 = a2 / b2.  [para(40(a,1),22(a,2,2)),rewrite([36(10),293(11),2(8)]),flip(a)].
622 b1 / (b2 / a2) = a1 / (a2 / b2).  [back_rewrite(458),rewrite([511(9)])].
706 b3 / a3 = b2 / a2.  [para(36(a,1),20(a,1,1)),rewrite([7(10),20(11)]),flip(a)].
717 b3 / (b2 / a2) = a3 / (a2 / b2).  [para(706(a,1),7(a,1,2)),rewrite([514(9)])].
794 (x / y) / ((x / z) / y) = z / (z / (x / y)).  [para(58(a,1),7(a,1,2))].
798 (b3 / x) / b2 = (a3 / a2) / x.  [para(15(a,1),58(a,1,1)),flip(a)].
841 (x / (x / y)) / (z / (y / x)) = x / (x / (y / z)).  [para(58(a,1),22(a,2)),rewrite([58(9),794(9)])].
1595 (x / (y / b2)) / (a3 / (y / b2)) = (x / a3) / ((y / a2) / b3).  [para(316(a,1),5(a,1,2)),flip(a)].
2372 (x / y) / (z / (z / (y / x))) = x / y.  [para(23(a,1),443(a,1,1)),rewrite([841(6),23(10)])].
2792 (b1 / x) / (b2 / a2) = (a1 / (a2 / b2)) / x.  [para(622(a,1),58(a,1,1)),flip(a)].
12559 (x / ((y / z) / u)) / ((y / (z / w)) / ((y / z) / u)) = x / (y / (z / w)).  [para(314(a,1),5(a,1,2)),rewrite([2(5)]),flip(a)].
27006 (x / y) / ((y / x) / z) = x / y.  [para(443(a,1),145(a,2)),rewrite([2372(5)])].
34571 ((x / y) / z) / ((y / x) / u) = (x / y) / z.  [para(27006(a,1),58(a,1,1)),flip(a)].
34654 (x / (x / y)) / (z / (x / y)) = y / (y / (x / z)).  [para(27006(a,1),78(a,1,1)),rewrite([6(6),4(7),2(6),794(9),19(14),2(10)])].
34669 (x / y) / (z / (z / (u / (y / x)))) = (x / y) / (z / (z / u)).  [para(27006(a,1),142(a,1,1)),rewrite([34654(7),34571(13)])].
48623 x / (y / ((y / z) / x)) = x / y.  [para(28(a,1),263(a,1,1)),rewrite([794(10),34669(8),23(5),6(7),2(7)]),flip(a)].
49660 b2 / (b3 / ((a3 / a2) / x)) = a2 / a3.  [para(11(a,1),48623(a,2)),rewrite([798(6)])].
50238 (b2 / x) / (b3 / ((a3 / a2) / y)) = (a2 / a3) / x.  [para(49660(a,1),58(a,1,1)),flip(a)].
68845 b1 / b3 = a1 / a3.  [para(2792(a,1),284(a,1,1)),rewrite([58(20),717(15),12559(21),1595(11),3(6),4(6),2(5),50238(16),6(11),2(8)]),flip(a)].
68846 $F.  [resolve(68845,a,16,a)].

============================== end of proof ==========================
```

For the reverse implication we must show that $\frac{a_1}{b_1} \equiv' \frac{a_2}{b_2}$ entails $\frac{a_1}{b_1} \equiv \frac{a_2}{b_2}$. Unfolding definitions, we must show that if $a_1/a_2 = b_1/b_2$ and $a_2/a_1 = b_2/b_1$, then $a_1/b_1 = a_2/b_2$ and $b_1/a_1 = b_2/a_2$. The Prover9 proof below shows the first of these two, with the other following by symmetry.

```
============================== PROOF =================================

% Proof 1 at 18.28 (+ 0.10) seconds.
% Length of proof is 22.
% Level of proof is 7.
% Maximum clause weight is 25.000.
% Given clauses 265.

1 a1 / b1 = a2 / b2 # label(non_clause) # label(goal).  [goal].
2 x / 1 = x.  [assumption].
3 x / x = 1.  [assumption].
4 1 / x = 1.  [assumption].
5 (x / y) / (z / y) = (x / z) / (y / z).  [assumption].
6 (x / y) / x = 1.  [assumption].
7 x / (x / y) = y / (y / x).  [assumption].
8 a1 / a2 = b1 / b2.  [assumption].
9 b1 / b2 = a1 / a2.  [copy(8),flip(a)].
10 a2 / a1 = b2 / b1.  [assumption].
11 b2 / b1 = a2 / a1.  [copy(10),flip(a)].
12 a2 / b2 != a1 / b1.  [deny(1)].
13 ((x / y) / (z / y)) / (u / (y / z)) = ((x / z) / u) / ((y / z) / u).  [para(5(a,1),5(a,1,1))].
27 (a1 / a2) / b1 = 1.  [para(9(a,1),6(a,1,1))].
28 b2 / (a2 / a1) = b1 / (a1 / a2).  [para(9(a,1),7(a,1,2)),rewrite([11(9)]),flip(a)].
30 (a2 / a1) / b2 = 1.  [para(11(a,1),6(a,1,1))].
80 (x / (a1 / a2)) / (b1 / (a1 / a2)) = x / b1.  [para(27(a,1),5(a,1,2)),rewrite([2(4)]),flip(a)].
87 (x / (a2 / a1)) / (b1 / (a1 / a2)) = x / b2.  [para(30(a,1),5(a,1,2)),rewrite([2(4),28(11)]),flip(a)].
92 (x / a2) / b1 = (x / a1) / b2.  [para(30(a,1),13(a,2,2)),rewrite([28(11),80(12),2(10)])].
544 (x / (y / a2)) / (b1 / (y / a2)) = (x / b1) / ((y / a1) / b2).  [para(92(a,1),5(a,1,2)),flip(a)].
21342 a2 / b2 = a1 / b1.  [para(7(a,1),87(a,1,1)),rewrite([544(11),3(6),4(6),2(5)]),flip(a)].
21343 $F.  [resolve(21342,a,12,a)].

============================== end of proof ==========================
```

◀

Proof of $\leqslant$-orderedness in Lem. 37. We first give the proof for the numerators, then that for the denominators.

```
============================== PROOF =================================

% Proof 1 at 325.13 (+ 2.60) seconds.
% Length of proof is 148.
% Level of proof is 14.
% Maximum clause weight is 51.000.
% Given clauses 2447.

1 (a * (e / b)) / (f * (b / e)) = ((a * (e / b)) / (f * (b / e))) ^ ((c * (e / d)) / (f * (d / e)))
  # label(non_clause) # label(goal).  [goal].
2 x / 1 = x.  [assumption].
3 x / x = 1.  [assumption].
4 1 / x = 1.  [assumption].
5 (x / y) / (z / y) = (x / z) / (y / z).  [assumption].
6 (x / y) / x = 1.  [assumption].
7 x / (x / y) = y / (y / x).  [assumption].
8 x ^ y = x / (x / y).  [assumption].
9 x v y = x * (y / x).  [assumption].
10 (x * y) / x = y.  [assumption].
11 x / (x * y) = 1.  [assumption].
12 a ^ b = 1.  [assumption].
13 a / (a / b) = 1.  [copy(12),rewrite([8(3)])].
14 c ^ d = 1.  [assumption].
15 c / (c / d) = 1.  [copy(14),rewrite([8(3)])].
```

```
16 e ^ f = 1.  [assumption].
17 e / (e / f) = 1.  [copy(16),rewrite([8(3)])].
18 a ^ c = a.  [assumption].
19 a / (a / c) = a.  [copy(18),rewrite([8(3)])].
20 b v d = b.  [assumption].
21 b * (d / b) = b.  [copy(20),rewrite([9(3)])].
22 ((a * (e / b)) / (f * (b / e))) ^ ((c * (e / d)) / (f * (d / e))) != (a * (e / b)) / (f * (b / e)).
  [deny(1)].
23 ((a * (e / b)) / (f * (b / e))) / (((a * (e / b)) / (f * (b / e))) / ((c * (e / d)) / (f * (d / e))))
  != (a * (e / b)) / (f * (b / e)).  [copy(22),rewrite([8(23)])].
24 ((x / y) / (z / y)) / (u / (y / z)) = ((x / z) / u) / ((y / z) / u).  [para(5(a,1),5(a,1,1))].
25 (x / (y / z)) / ((u / z) / (y / z)) = (x / (u / z)) / ((y / u) / (z / u)).  [para(5(a,1),5(a,1,2)),flip(a)].
26 ((x / y) / z) / (x / z) = 1.  [para(6(a,1),5(a,1,1)),rewrite([4(3)]),flip(a)].
27 (x / (y / z)) / (y / (y / z)) = x / y.  [para(6(a,1),5(a,1,2)),rewrite([2(3)]),flip(a)].
29 (x / (x / y)) / (z / (y / x)) = (y / z) / ((y / x) / z).  [para(7(a,1),5(a,1,1))].
30 (x / (y / z)) / (z / (z / y)) = x / y.  [para(7(a,1),5(a,1,2)),rewrite([6(8),2(8)])].
31 (x / (x / y)) / (y / (x / y)) = y / (y / (x / y)).  [para(7(a,1),5(a,1)),flip(a)].
32 (x / y) / ((x / z) / (y / z)) = (z / y) / ((z / y) / (x / y)).  [para(5(a,1),7(a,1,2))].
34 x / (x / (x / y)) = x / y.  [para(6(a,1),7(a,1,2)),rewrite([2(3)]),flip(a)].
35 x / (y / (y / x)) = x / y.  [para(7(a,1),7(a,1,2)),rewrite([6(6),2(6)])].
40 ((x * y) / z) / (x / z) = y / (z / x).  [para(10(a,1),5(a,1,1)),flip(a)].
41 (x / y) / z = x / (y * z).  [para(10(a,1),5(a,1,2)),rewrite([11(6),2(6)])].
42 (x * y) / y = x.  [para(10(a,1),7(a,1,2)),rewrite([11(4),2(4)])].
43 (((a * (e / b)) / f) / (b / e)) / ((((a * (e / b)) / f) / (b / e)) / (((c * (e / d)) / f) / (d / e)))
  != ((a * (e / b)) / f) / (b / e).  [back_rewrite(23),rewrite([41(11,R),41(22,R),41(33,R),41(46,R)])].
46 a / b = a.  [para(13(a,1),7(a,1,2)),rewrite([2(3),6(9),2(6)]),flip(a)].
49 c / d = c.  [para(15(a,1),7(a,1,2)),rewrite([2(3),6(9),2(6)]),flip(a)].
52 e / f = e.  [para(17(a,1),7(a,1,2)),rewrite([2(3),6(9),2(6)]),flip(a)].
55 a / c = 1.  [para(19(a,1),7(a,1,2)),rewrite([3(3),6(9),2(6)]),flip(a)].
56 d / b = 1.  [para(21(a,1),10(a,1,1)),rewrite([3(3)]),flip(a)].
75 (x / y) / z = (x / z) / y.  [para(6(a,1),24(a,2,2)),rewrite([30(6),2(6)])].
92 ((x / y) / z) / (u / z) = (x / (y * u)) / (z / u).  [para(10(a,1),24(a,1,2,2)),rewrite([41(4,R),3(3),4(4),2(4),10(8)]),flip(a)].
230 (x / d) / (b / d) = x / b.  [para(56(a,1),5(a,1,2)),rewrite([2(4)]),flip(a)].
231 b / (b / d) = d.  [para(56(a,1),7(a,1,2)),rewrite([2(3)]),flip(a)].
234 (x / (d / y)) / ((b / y) / (d / y)) = x / (b / y).  [para(56(a,1),25(a,2,2,1)),rewrite([4(16),2(14)])].
238 (x / y) / z = x / (z * y).  [para(42(a,1),5(a,1,2)),rewrite([41(6,R),6(6),2(6)])].
245 (x / ((y * z) / u)) / ((z / u) / ((y * z) / u)) = (x / (z / u)) / (y / (u / z)).  [para(42(a,1),25(a,2,2,1))].
252 (x / y) / ((z * x) / y) = 1.  [para(10(a,1),26(a,1,1,1))].
271 (a / x) / c = 1.  [para(55(a,1),26(a,1,2)),rewrite([2(6)])].
272 (d / x) / b = 1.  [para(56(a,1),26(a,1,2)),rewrite([2(6)])].
273 (x / y) / ((x * z) / y) = 1.  [para(42(a,1),26(a,1,1,1))].
275 (x / (y / z)) / (u / (z / y)) = x / ((u * y) / z).  [back_rewrite(245),rewrite([252(7),2(5)]),flip(a)].
286 (x / c) / (x / a) = 1.  [para(7(a,1),271(a,1,1)),rewrite([75(5)])].
297 (x / b) / (x / d) = 1.  [para(7(a,1),272(a,1,1)),rewrite([75(5)])].
307 (x / y) / (x / (y / z)) = 1.  [para(27(a,1),6(a,1,1))].
436 x / ((x * c) / a) = 1.  [para(42(a,1),286(a,1,1))].
454 x / ((b * x) / d) = 1.  [para(10(a,1),297(a,1,1))].
460 a / (a / d) = 1.  [para(46(a,1),297(a,1,1))].
465 (x * d) / (x * b) = 1.  [para(42(a,1),297(a,1,2)),rewrite([238(5)])].
470 a / d = a.  [para(460(a,1),7(a,1,2)),rewrite([2(3),6(9),2(6)]),flip(a)].
693 x / (b / a) = x / b.  [para(46(a,1),30(a,1,2,2)),rewrite([3(7),2(6)])].
694 x / (d / c) = x / d.  [para(49(a,1),30(a,1,2,2)),rewrite([3(7),2(6)])].
792 (x / y) / ((y * c) / (y * a)) = x / ((y * c) / a).  [para(436(a,1),5(a,1,2)),rewrite([2(7),238(11)]),flip(a)].
843 (x / y) / (y / x) = x / y.  [para(31(a,1),29(a,2,2)),rewrite([6(3),2(3),27(6),27(10)])].
881 (x / y) / ((b * x) / d) = 1.  [para(454(a,1),26(a,1,2)),rewrite([2(8)])].
1099 x * d = (x * b) / (b / d).  [para(465(a,1),7(a,1,2)),rewrite([2(4),41(9,R),10(7)])].
1189 b / a = b.  [para(693(a,1),34(a,1,2)),rewrite([3(4),2(3)]),flip(a)].
1196 (b / x) / (a / x) = b / (x / a).  [para(1189(a,1),5(a,1,1)),flip(a)].
1219 d / c = d.  [para(694(a,1),34(a,1,2)),rewrite([3(4),2(3)]),flip(a)].
1226 (d / x) / (c / x) = d / (x / c).  [para(1219(a,1),5(a,1,1)),flip(a)].
1253 f / e = f.  [para(52(a,1),35(a,1,2,2)),rewrite([3(4),2(3)]),flip(a)].
1268 d / a = d.  [para(470(a,1),35(a,1,2,2)),rewrite([3(4),2(3)]),flip(a)].
1273 (f / x) / (e / x) = f / (x / e).  [para(1253(a,1),5(a,1,1)),flip(a)].
1519 (x * y) / (x * z) = y / z.  [para(10(a,1),40(a,2,2)),rewrite([41(5,R),3(4),4(5),2(5)])].
1543 ((x * y) * z) / (y * x) = z.  [para(42(a,1),40(a,1,2)),rewrite([41(4),41(6,R),6(6),2(6)])].
1544 (x * y) / (z * x) = y / z.  [para(42(a,1),40(a,2,2)),rewrite([41(5,R),6(5),2(5)])].
1552 ((x * y) / z) / x = y / z.  [para(40(a,1),27(a,1,1)),rewrite([30(5)]),flip(a)].
1572 ((x * y) / (x / z)) / (y / (z / x)) = z / (z / (x * y)).  [para(40(a,1),29(a,2,2)),rewrite([273(7),2(5)]),flip(a)].
1587 (a * x) / (x * c) = 1.  [para(436(a,1),40(a,2)),rewrite([41(9,R),271(9),2(7)])].
1601 (x / (y / z)) / ((x / (y / z)) / (u / (z / y))) = (u / (z / y)) / (u / ((z * x) / y)).
  [para(40(a,1),32(a,2,1)),rewrite([273(9),2(7),40(12)]),flip(a)].
1631 (x / y) / (c / a) = x / ((y * c) / a).  [back_rewrite(792),rewrite([1519(6)])].
1643 (x * c) / (c / a) = a * x.  [para(1587(a,1),7(a,1,2)),rewrite([2(4),1544(9)]),flip(a)].
1708 (x * y) / (y * z) = x / z.  [para(42(a,1),41(a,1,1)),flip(a)].
1710 (x / (y * z)) / (x / z) = 1.  [para(41(a,1),26(a,1,1))].
1960 (x / y) / ((x / z) / y) = z / (z / (x / y)).  [para(75(a,1),7(a,1,2))].
1973 (a / x) / b = a / x.  [para(46(a,1),75(a,1,1)),flip(a)].
1974 (c / x) / d = c / x.  [para(49(a,1),75(a,1,1)),flip(a)].
1975 (e / x) / f = e / x.  [para(52(a,1),75(a,1,1)),flip(a)].
1986 ((x * y) / z) / y = x / z.  [para(42(a,1),75(a,1,1)),flip(a)].
1988 (b / x) / (b / d) = d / x.  [para(231(a,1),75(a,1,1)),flip(a)].
2025 (b / x) / a = b / x.  [para(1189(a,1),75(a,1,1)),flip(a)].
2026 (d / x) / c = d / x.  [para(1219(a,1),75(a,1,1)),flip(a)].
2029 (d / x) / a = d / x.  [para(1268(a,1),75(a,1,1)),flip(a)].
2040 x / ((y / z) / u) = x / ((y / u) / z).  [para(75(a,1),40(a,2,2)),rewrite([40(6)])].
2323 b / (a / x) = b.  [para(1973(a,1),35(a,1,2,2)),rewrite([3(6),2(3)]),flip(a)].
2726 b / (x / (x / a)) = b.  [para(7(a,1),2323(a,1,2))].
2732 (b / x) / (a / y) = b / x.  [para(2323(a,1),75(a,1,1)),flip(a)].
2734 b / (x / a) = b / x.  [back_rewrite(1196),rewrite([2732(5)]),flip(a)].
2752 d / (c / x) = d.  [para(1974(a,1),35(a,1,2,2)),rewrite([3(6),2(3)]),flip(a)].
2762 d / (x / (x / c)) = d.  [para(7(a,1),2752(a,1,2))].
2769 (d / x) / (c / y) = d / x.  [para(2752(a,1),75(a,1,1)),flip(a)].
2771 d / (x / c) = d / x.  [back_rewrite(1226),rewrite([2769(5)]),flip(a)].
2790 f / (e / x) = f.  [para(1975(a,1),35(a,1,2,2)),rewrite([3(6),2(3)]),flip(a)].
3174 (f / x) / (e / y) = f / x.  [para(2790(a,1),75(a,1,1)),flip(a)].
3178 f / (x / e) = f / x.  [back_rewrite(1273),rewrite([3174(5)]),flip(a)].
3760 c / (d / x) = c.  [para(2026(a,1),35(a,1,2,2)),rewrite([3(6),2(3)]),flip(a)].
3772 c / (x / (x / d)) = c.  [para(7(a,1),3760(a,1,2))].
3778 (c / x) / (d / y) = c / x.  [para(3760(a,1),75(a,1,1)),flip(a)].
4292 a / (d / x) = a.  [para(2029(a,1),35(a,1,2,2)),rewrite([3(6),2(3)]),flip(a)].
4305 (a / x) / ((d / y) / x) = a / (x / (d / y)).  [para(4292(a,1),5(a,1,1)),flip(a)].
4311 a / ((d / x) / y) = a.  [para(41(a,2),4292(a,1,2))].
4329 b / (x / (y / a)) = b / (x / y).  [para(2726(a,1),25(a,1,1)),rewrite([27(8),1960(13),2732(14)]),flip(a)].
4948 d / (x / (y / c)) = d / (x / y).  [para(2762(a,1),25(a,1,1)),rewrite([27(8),1960(13),2769(14)]),flip(a)].
```

```
7457 c / (x / (y / d)) = c / (x / y).  [para(3772(a,1),25(a,1,1)),rewrite([27(8),1960(13),3778(14)]),flip(a)].
8424 (a / x) / ((d / y) / z) = a / x.  [para(4311(a,1),75(a,1,1)),flip(a)].
8451 a / (x / (d / y)) = a / x.  [back_rewrite(4305),rewrite([8424(6)]),flip(a)].
9223 b / (x / (a * y)) = b / (x / y).  [para(238(a,1),2734(a,1,2))].
9250 d / (x / (c * y)) = d / (x / y).  [para(238(a,1),2771(a,1,2))].
10330 x / ((x * y) / (y / z)) = 1.  [para(42(a,1),307(a,1,1))].
16959 x * y = y * x.  [para(1543(a,1),7(a,1,2)),rewrite([42(3),41(6,R),1708(5),3(3),4(4),2(4)])].
17705 (((a * (e / b)) / f) / (b / e)) / (((((c * (e / b)) / f) / (b / e)) / (c / a)) / (((c * (e / d)) / f) / (d / e)))
  != ((a * (e / b)) / f) / (b / e).  [para(1643(a,2),43(a,1,2,1,1,1)),rewrite([16959(16),75(22),75(26)])].
17833 (a * x) / ((b * (x * c)) / d) = 1.  [para(1643(a,1),881(a,1,1))].
17863 (c * x) / (c / a) = a * x.  [para(16959(a,1),1643(a,1,1))].
18962 x / ((x * (y * z)) / z) = 1.  [para(42(a,1),1710(a,1,1))].
20581 f / ((x * e) / y) = f / (x / y).  [para(1986(a,1),3178(a,1,2)),flip(a)].
20687 d / (b / (b / x)) = d / x.  [para(34(a,1),1988(a,1,1)),rewrite([1988(6)]),flip(a)].
20995 (b / x) / (y / a) = (b / x) / y.  [para(2732(a,1),92(a,2)),rewrite([2025(4),41(8,R)])].
25756 x / ((x * y) / (z / (z / y))) = 1.  [para(7(a,1),10330(a,1,2,2))].
30626 (d / x) / y = d / (b / ((b / x) / y)).  [para(41(a,2),20687(a,1,2,2)),rewrite([41(10,R)]),flip(a)].
43520 b / (x / ((a * y) / z)) = b / (x / (y / z)).  [para(1552(a,1),4329(a,1,2,2)),flip(a)].
47774 x / (b / (d / x)) = x / b.  [para(35(a,1),234(a,1,1)),rewrite([27(11),230(6)]),flip(a)].
48034 (a * x) / (b / (b / x)) = (a * x) / b.  [para(9223(a,1),35(a,1,2))].
48078 (a * x) / (b / (d / x)) = (a * x) / b.  [para(9223(a,1),47774(a,1,2))].
48214 (c * x) / (d / (d / x)) = (c * x) / d.  [para(9250(a,1),35(a,1,2))].
65562 b / ((x * b) / d) = d / x.  [para(231(a,1),275(a,1,1)),rewrite([56(4),2(3)]),flip(a)].
78759 ((a * x) / b) / ((x * c) / d) = 1.  [para(17833(a,1),275(a,2)),rewrite([41(12,R),30626(12),4948(16),20687(14),75(12),48078(7)])].
245607 (x * (y / x)) / (x / y) = y.  [para(3(a,1),1572(a,1,2)),rewrite([2(6),41(7,R),3(7),2(6)])].
246535 (x * (y * z)) / z = x * y.  [para(18962(a,1),245607(a,1,2)),rewrite([238(4),1519(4),42(2),2(3)]),flip(a)].
246678 x * (y / z) = (x * y) / (z / (z / y)).  [para(25756(a,1),245607(a,1,2)),rewrite([1552(5),35(3),2(4)])].
248267 ((((a * e) / b) / f) / (b / e)) / ((((a * e) / b) / f) / ((((c * e) / d) / f) / (d / e)))
  != (((a * e) / b) / f) / (b / e).  [back_rewrite(17705),rewrite([246678(5),48034(9),246678(16),75(22),75(26),27(26),75(18),41(18),
  75(22),17863(18),41(18,R),246678(23),48214(27),246678(36),48034(40)])].
250480 ((a * x) / b) / (x / d) = a.
  [para(78759(a,1),1601(a,2,2)),rewrite([3760(4),3760(5),7457(10),1986(7),46(5),7(5),55(4),2(3),2(10)]),flip(a)].
250518 ((a * x) / b) / ((y * x) / d) = a / y.  [para(250480(a,1),275(a,1,1)),rewrite([8451(5)]),flip(a)].
251447 b / ((c * (x * b)) / d) = d / x.
  [para(65562(a,1),1631(a,1,1)),rewrite([30626(6),20995(8),4948(8),20687(6),16959(9),246678(9),
  41(12,R),30626(12),6(14),2(12),56(11),2(10),75(11),41(11),1099(10),43520(16),231(12)]),flip(a)].
252566 ((c * x) / d) / (d / x) = (c * x) / d.  [para(251447(a,1),843(a,1,2)),rewrite([75(8),246535(6),75(15),246535(13)])].
253954 ((((a * e) / b) / f) / (b / e)) / ((((a * e) / b) / f) / (((c * e) / d) / f)) != (((a * e) / b) / f) / (b / e).
  [para(2040(a,1),248267(a,1,2)),rewrite([252566(27)])].
254740 $F.  [para(5(a,1),253954(a,1,2)),rewrite([250518(22),55(14),20581(19),49(16),4(16),2(13)]),xx(a)].

============================== end of proof ==========================
```

and

```
============================== PROOF =================================

% Proof 1 at 90.22 (+ 0.51) seconds.
% Length of proof is 70.
% Level of proof is 9.
% Maximum clause weight is 47.000.
% Given clauses 716.

1 (f * (b / e)) / (a * (e / b)) = ((f * (b / e)) / (a * (e / b))) v ((f * (d / e)) / (c * (e / d)))
  # label(non_clause) # label(goal).  [goal].
2 x / 1 = x.  [assumption].
3 x / x = 1.  [assumption].
4 1 / x = 1.  [assumption].
5 (x / y) / (z / y) = (x / z) / (y / z).  [assumption].
6 (x / y) / x = 1.  [assumption].
7 x / (x / y) = y / (y / x).  [assumption].
8 x ^ y = x / (x / y).  [assumption].
9 x v y = x * (y / x).  [assumption].
10 (x * y) / x = y.  [assumption].
11 x / (x * y) = 1.  [assumption].
12 a ^ b = 1.  [assumption].
13 a / (a / b) = 1.  [copy(12),rewrite([8(3)])].
16 e ^ f = 1.  [assumption].
17 e / (e / f) = 1.  [copy(16),rewrite([8(3)])].
18 a ^ c = a.  [assumption].
19 a / (a / c) = a.  [copy(18),rewrite([8(3)])].
20 b v d = b.  [assumption].
21 b * (d / b) = b.  [copy(20),rewrite([9(3)])].
22 ((f * (b / e)) / (a * (e / b))) v ((f * (d / e)) / (c * (e / d))) != (f * (b / e)) / (a * (e / b)).  [deny(1)].
23 ((f * (b / e)) / (a * (e / b))) * (((f * (d / e)) / (c * (e / d))) / ((f * (b / e)) / (a * (e / b))))
  != (f * (b / e)) / (a * (e / b)).  [copy(22),rewrite([9(23)])].
24 ((x / y) / (z / y)) / (u / (y / z)) = ((x / z) / u) / ((y / z) / u).  [para(5(a,1),5(a,1,1))].
25 (x / (y / z)) / ((u / z) / (y / z)) = (x / (u / z)) / ((y / u) / (z / u)).  [para(5(a,1),5(a,1,2)),flip(a)].
26 ((x / y) / z) / (x / z) = 1.  [para(6(a,1),5(a,1,1)),rewrite([4(3)]),flip(a)].
27 (x / (y / z)) / (y / (y / z)) = x / y.  [para(6(a,1),5(a,1,2)),rewrite([2(3)]),flip(a)].
29 (x / (x / y)) / (z / (y / x)) = (y / z) / ((y / x) / z).  [para(7(a,1),5(a,1,1))].
30 (x / (y / z)) / (z / (z / y)) = x / y.  [para(7(a,1),5(a,1,2)),rewrite([6(8),2(8)])].
31 (x / (x / y)) / (y / (x / y)) = y / (y / (x / y)).  [para(7(a,1),5(a,1)),flip(a)].
34 x / (x / (x / y)) = x / y.  [para(6(a,1),7(a,1,2)),rewrite([2(3)]),flip(a)].
35 x / (y / (y / x)) = x / y.  [para(7(a,1),7(a,1,2)),rewrite([6(6),2(6)])].
39 1 * x = x.  [para(10(a,1),2(a,1)),flip(a)].
40 ((x * y) / z) / (x / z) = y / (z / x).  [para(10(a,1),5(a,1,1)),flip(a)].
41 (x / y) / z = x / (y * z).  [para(10(a,1),5(a,1,2)),rewrite([11(6),2(6)])].
42 (x * y) / y = x.  [para(10(a,1),7(a,1,2)),rewrite([11(4),2(4)])].
45 a / b = a.  [para(13(a,1),7(a,1,2)),rewrite([2(3),6(9),2(6)]),flip(a)].
51 e / f = e.  [para(17(a,1),7(a,1,2)),rewrite([2(3),6(9),2(6)]),flip(a)].
54 a / c = 1.  [para(19(a,1),7(a,1,2)),rewrite([3(3),6(9),2(6)]),flip(a)].
55 d / b = 1.  [para(21(a,1),10(a,1,1)),rewrite([3(3)]),flip(a)].
74 (x / y) / z = (x / z) / y.  [para(6(a,1),24(a,2,2)),rewrite([30(6),2(6)])].
233 (x / (d / y)) / ((b / y) / (d / y)) = x / (b / y).  [para(55(a,1),25(a,2,2,1)),rewrite([4(16),2(14)])].
244 (x / ((y * z) / u)) / ((z / u) / ((y * z) / u)) = (x / (z / u)) / (y / (u / z)).  [para(42(a,1),25(a,2,2,1))].
251 (x / y) / ((z * x) / y) = 1.  [para(10(a,1),26(a,1,1,1))].
270 (a / x) / c = 1.  [para(54(a,1),26(a,1,2)),rewrite([2(6)])].
274 (x / (y / z)) / (u / (z / y)) = x / ((u * y) / z).  [back_rewrite(244),rewrite([251(7),2(5)]),flip(a)].
285 (x / c) / (x / a) = 1.  [para(7(a,1),270(a,1,1)),rewrite([74(5)])].
437 ((x / c) / y) / (x / a) = 1.  [para(285(a,1),26(a,1,2)),rewrite([2(8)])].
692 x / (b / a) = x / b.  [para(45(a,1),30(a,1,2,2)),rewrite([3(7),2(6)])].
842 (x / y) / (y / x) = x / y.  [para(31(a,1),29(a,2,2)),rewrite([6(3),2(3),27(6),27(10)])].
1188 b / a = b.  [para(692(a,1),34(a,1,2)),rewrite([3(4),2(3)]),flip(a)].
```

```
1252 f / e = f.  [para(51(a,1),35(a,1,2,2)),rewrite([3(4),2(3)]),flip(a)].
1272 (f / x) / (e / x) = f / (x / e).  [para(1252(a,1),5(a,1,1)),flip(a)].
1542 ((x * y) * z) / (y * x) = z.  [para(42(a,1),40(a,1,2)),rewrite([41(4),41(6,R),6(6),2(6)])].
1707 (x * y) / (y * z) = x / z.  [para(42(a,1),41(a,1,1)),flip(a)].
1959 (x / y) / ((x / z) / y) = z / (z / (x / y)).  [para(74(a,1),7(a,1,2))].
1974 (e / x) / f = e / x.  [para(51(a,1),74(a,1,1)),flip(a)].
1985 ((x * y) / z) / y = x / z.  [para(42(a,1),74(a,1,1)),flip(a)].
2789 f / (e / x) = f.  [para(1974(a,1),35(a,1,2,2)),rewrite([3(6),2(3)]),flip(a)].
3166 (f / x) / ((e / y) / x) = f / (x / (e / y)).  [para(2789(a,1),5(a,1,1)),flip(a)].
3167 f / (x / (x / e)) = f.  [para(7(a,1),2789(a,1,2))].
3172 f / ((e / x) / y) = f.  [para(41(a,2),2789(a,1,2))].
3173 (f / x) / (e / y) = f / x.  [para(2789(a,1),74(a,1,1)),flip(a)].
3177 f / (x / e) = f / x.  [back_rewrite(1272),rewrite([3173(5)]),flip(a)].
5613 f / (x / (y / e)) = f / (x / y).  [para(3167(a,1),25(a,1,1)),rewrite([27(8),1959(13),3173(14)]),flip(a)].
5667 (f / x) / ((e / y) / z) = f / x.  [para(3172(a,1),74(a,1,1)),flip(a)].
5688 f / (x / (e / y)) = f / x.  [back_rewrite(3166),rewrite([5667(6)]),flip(a)].
16958 x * y = y * x.  [para(1542(a,1),7(a,1,2)),rewrite([42(3),41(6,R),1707(5),3(3),4(4),2(4)])].
21396 ((f / x) / y) / (e / z) = (f / x) / y.  [para(41(a,2),3173(a,1,1)),rewrite([41(9,R)])].
22637 f / ((x * (e / y)) / z) = f / (x / z).  [para(1985(a,1),5688(a,1,2)),flip(a)].
48489 (x * (d / y)) / (b / y) = x / ((b / y) / (d / y)).  [para(42(a,1),233(a,1,1)),flip(a)].
66357 $F.  [para(274(a,2),23(a,1,2)),rewrite([41(31,R),74(27),1188(25),842(29),74(26),48489(20),5613(20),74(17),
  3177(18),41(22,R),74(18),21396(22),5613(29),22637(27),45(22),437(22),16958(13),39(13)]),xx(a)].

============================== end of proof ==========================
```

◀

Proof of Thm. 42. It was left open to show that if equations (16)–(20) hold, each CRA equation is satisfied.

- (1) is the same as (17);
- (5) follows from (18) and (16) and using $1/a = 1$, which is easily derived;
- (6) is the same as (19);
- (4) follows from:

```
============================== PROOF =================================

% Proof 1 at 9.37 (+ 0.08) seconds.
% Length of proof is 54.
% Level of proof is 13.
% Maximum clause weight is 23.000.
% Given clauses 182.

1 (x / y) / (z / y) = (x / z) / (y / z) # label(non_clause) # label(goal).  [goal].
2 x / x = 1.  [assumption].
3 x ^ y = x / (x / y).  [assumption].
4 x ^ y = y ^ x.  [assumption].
5 x / (x / y) = y / (y / x).  [copy(4),rewrite([3(1),3(3)])].
6 (x / y) / z = (x / z) / y.  [assumption].
7 x / 1 = x.  [assumption].
8 (x / y) / (y / x) = x / y.  [assumption].
9 (c1 / c3) / (c2 / c3) != (c1 / c2) / (c3 / c2).  [deny(1)].
10 x / (y / (y / x)) = x / (x / (x / y)).  [para(5(a,1),3(a,2,2)),rewrite([3(2)]),flip(a)].
11 (x / y) / x = 1 / y.  [para(2(a,1),6(a,1,1)),flip(a)].
13 (x / y) / ((x / z) / y) = z / (z / (x / y)).  [para(6(a,1),5(a,1,2))].
14 (x / (x / y)) / z = (y / z) / (y / x).  [para(5(a,1),6(a,1,1))].
15 ((x / y) / z) / u = ((x / u) / y) / z.  [para(6(a,1),6(a,1,1)),flip(a)].
20 ((x / y) / z) / (y / x) = (x / y) / z.  [para(8(a,1),6(a,1,1)),flip(a)].
28 (x / y) / (1 / y) = x / (y / (y / x)).  [para(10(a,2),5(a,1)),rewrite([11(6)]),flip(a)].
31 (x / (y / (y / x))) / z = (x / z) / (x / (x / y)).  [para(10(a,2),6(a,1,1))].
39 1 / (x / y) = 1.  [para(5(a,1),11(a,1,1)),rewrite([6(3),2(3)]),flip(a)].
40 ((x / y) / z) / x = (1 / y) / z.  [para(11(a,1),6(a,1,1)),flip(a)].
46 1 / x = 1.  [para(7(a,1),39(a,1,2))].
48 ((x / y) / z) / x = 1.  [back_rewrite(40),rewrite([46(5),46(5)])].
49 x / (y / (y / x)) = x / y.  [back_rewrite(28),rewrite([46(3),7(3)]),flip(a)].
50 (x / y) / x = 1.  [back_rewrite(11),rewrite([46(4)])].
52 (x / y) / (x / (x / z)) = (x / z) / y.  [back_rewrite(31),rewrite([49(3)]),flip(a)].
55 x / (x / (x / y)) = x / y.  [back_rewrite(10),rewrite([49(3)]),flip(a)].
58 (x / (x / y)) / ((y / z) / (y / x)) = z / (z / (y / (y / x))).  [para(5(a,1),13(a,1,1))].
59 (x / y) / ((z / (z / x)) / y) = (x / z) / ((x / z) / (x / y)).  [para(5(a,1),13(a,1,2,1))].
71 x / (x / ((x / y) / z)) = (x / y) / z.  [para(50(a,1),13(a,1,2,1)),rewrite([46(4),7(4)]),flip(a)].
73 (x / y) / (x / (y / z)) = 1.  [para(5(a,1),48(a,1,1)),rewrite([6(4)])].
92 (x / y) / (x / z) = (z / y) / (z / x).  [para(14(a,1),6(a,1))].
94 (x / (x / y)) / z = (y / (y / x)) / z.  [para(14(a,2),6(a,1))].
115 (x / y) / (x / ((y / z) / u)) = 1.  [para(48(a,1),14(a,2,1)),rewrite([6(5),46(10)])].
182 ((x / y) / z) / u = ((x / u) / z) / y.  [para(15(a,2),6(a,1))].
187 ((x / y) / z) / (z / x) = (x / z) / y.  [para(8(a,1),15(a,1,1)),flip(a)].
347 (x / y) / ((y / x) / z) = x / y.  [para(20(a,1),49(a,1,2,2)),rewrite([2(6),7(3)]),flip(a)].
747 (x / y) / (x / (x / z)) = (x / y) / z.  [para(52(a,2),6(a,1))].
838 (x / y) / ((y / z) / x) = x / y.  [para(6(a,1),347(a,1,2))].
926 (x / (y / (y / (z / (z / x))))) / ((z / y) / (z / x)) = x / z.  [para(58(a,1),13(a,1,2)),rewrite([6(9),115(16),7(12)])].
945 (x / (x / y)) / ((x / (x / y)) / z) = y / (y / (x / (x / z))).  [para(55(a,1),58(a,1,2,1)),rewrite([58(6),747(11)]),flip(a)].
1254 (x / (y / z)) / ((z / (z / y)) / (y / x)) = x / y.  [para(14(a,2),59(a,1,2)),rewrite([73(12),7(10)])].
1262 (x / (y / (y / z))) / ((y / (y / x)) / z) = x / y.  [para(55(a,1),59(a,1,2)),rewrite([945(6),55(5),945(15),55(14),73(13),7(10)])].
1267 x / (y / ((y / x) / z)) = x / y.  [para(73(a,1),59(a,1,2)),rewrite([7(6),73(11),7(7)])].
1484 x / (y / ((z / x) / (z / y))) = x / y.  [para(92(a,1),1267(a,1,2,2))].
1548 (x / (x / y)) / ((y / (y / x)) / z) = y / (y / (x / (x / z))).  [para(94(a,1),3(a,2,2)),rewrite([3(3),945(6)]),flip(a)].
1559 x / (y / (y / (z / (z / x)))) = x / (y / (y / z)).  [para(94(a,1),49(a,1,2)),rewrite([1548(6)])].
1630 (x / (y / (y / z))) / ((z / y) / (z / x)) = x / z.  [back_rewrite(926),rewrite([1559(5)])].
2872 (((x / y) / z) / u) / (u / x) = (x / u) / (x / ((x / y) / z)).  [para(71(a,1),187(a,1,1,1))].
2897 ((x / y) / z) / u = (x / y) / (x / ((x / z) / u)).  [para(187(a,1),182(a,1,1)),rewrite([6(8),2872(8)])].
5634 (x / y) / (z / (y / x)) = (x / y) / z.  [para(8(a,1),1484(a,1,2,2,1)),rewrite([49(6)])].
5839 (x / y) / (z / ((y / u) / x)) = (x / y) / z.  [para(838(a,1),5634(a,1,1)),rewrite([2897(5),8(5),747(5),838(9)])].
17428 (x / y) / (z / (y / (y / x))) = (x / y) / (z / y).  [para(1262(a,1),5839(a,1,2)),flip(a)].
19369 (x / (y / (y / z))) / (z / y) = x / z.  [para(1630(a,1),1254(a,1,1)),rewrite([17428(11),2897(10),347(13)]),flip(a)].
19647 (x / y) / (z / y) = (x / z) / (y / z).  [para(19369(a,1),19369(a,1,1)),rewrite([50(6),7(6),6(6)])].
19648 $F.  [resolve(19647,a,9,a)].
```

```
============================== end of proof ==========================
```

For completeness sake we also used Prover9 to reprove the result of [13, 12] that commutative BCK algebras with relative cancellation are equivalent to algebras satisying (16)–(20). We proceeded by first showing that commutative BCK algebras with relative cancellation make each of (16)–(20) hold. To keep proofs, relatively, short we add already derived equations to the assumptions.

- (16) holds for BCI algebras as it is the same as (11);
- (17) holds for BCI algebras:

```
============================== PROOF =================================

% Proof 1 at 0.01 (+ 0.00) seconds.
% Length of proof is 10.
% Level of proof is 3.
% Maximum clause weight is 13.000.
% Given clauses 9.

1 x / 1 = x # label(non_clause) # label(goal).  [goal].
5 (x / (x / y)) / y = 1.  [assumption].
6 x / x = 1.  [assumption].
7 x / y != 1 | y / x != 1 | x = y.  [assumption].
8 x / 1 != 1 | x = 1.  [assumption].
9 x / 1 != 1 | 1 = x.  [copy(8),flip(b)].
11 c1 / 1 != c1.  [deny(1)].
23 (x / 1) / x = 1.  [para(6(a,1),5(a,1,1,2))].
26 x / (x / 1) = 1.  [hyper(9,a,5,a),flip(a)].
48 $F.  [ur(7,b,23,a,c,11,a(flip)),rewrite([26(5)]),xx(a)].

============================== end of proof ==========================
```

- (18) holds for BCI algebras:

```
============================== PROOF =================================

% Proof 1 at 0.05 (+ 0.01) seconds.
% Length of proof is 11.
% Level of proof is 4.
% Maximum clause weight is 17.000.
% Given clauses 34.

1 (x / y) / z = (x / z) / y # label(non_clause) # label(goal).  [goal].
4 ((x / y) / (x / z)) / (z / y) = 1.  [assumption].
5 (x / (x / y)) / y = 1.  [assumption].
7 x / y != 1 | y / x != 1 | x = y.  [assumption].
10 x / 1 = x.  [assumption].
12 (c1 / c3) / c2 != (c1 / c2) / c3.  [deny(1)].
16 (x / (y / z)) / (x / ((u / z) / (u / y))) = 1.  [para(4(a,1),4(a,1,2)),rewrite([10(9)])].
19 (x / y) / (x / (z / (z / y))) = 1.  [para(5(a,1),4(a,1,2)),rewrite([10(7)])].
236 ((x / y) / z) / ((x / z) / y) = 1.  [para(19(a,1),16(a,1,2)),rewrite([10(7)])].
647 (x / y) / z = (x / z) / y.  [hyper(7,a,236,a,b,236,a)].
648 $F.  [resolve(647,a,12,a)].

============================== end of proof ==========================
```

- (19) holds for cBCK algebras as it is the same as (14);
- (20) is the only non-trivial equation; only it requires also relative cancellation to hold. It took Prover9 a bit more than one and a half hour to come up with a proof:

```
============================== PROOF =================================

% Proof 1 at 5810.83 (+ 33.71) seconds.
% Length of proof is 43.
% Level of proof is 10.
% Maximum clause weight is 36.000.
% Given clauses 2350.

1 (x / y) / (y / x) = x / y # label(non_clause) # label(goal).  [goal].
2 x / x = 1.  [assumption].
3 1 / x = 1.  [assumption].
4 x ^ y = x / (x / y).  [assumption].
5 x ^ y = y ^ x.  [assumption].
6 x / (x / y) = y / (y / x).  [copy(5),rewrite([4(1),4(3)])].
7 (x / y) / z = (x / z) / y.  [assumption].
8 x / y != 1 | x / z != 1 | y / x != z / x | y = z.  [assumption].
9 x / 1 = x.  [assumption].
10 (c1 / c2) / (c2 / c1) != c1 / c2.  [deny(1)].
11 x / (y / (y / x)) = x / (x / (x / y)).  [para(6(a,1),4(a,2,2)),rewrite([4(2)]),flip(a)].
12 (x / y) / x = 1.  [para(2(a,1),7(a,1,1)),rewrite([3(2)]),flip(a)].
14 (x / y) / ((x / z) / y) = z / (z / (x / y)).  [para(7(a,1),6(a,1,2))].
15 (x / (x / y)) / z = (y / z) / (y / x).  [para(6(a,1),7(a,1,1))].
16 ((x / y) / z) / u = ((x / u) / y) / z.  [para(7(a,1),7(a,1,1)),flip(a)].
24 x / (y / z) != 1 | x / u != 1 | (y / x) / z != u / x | y / z = u.  [para(7(a,1),8(c,1))].
28 x / (y / (y / x)) = x / y.  [para(11(a,2),6(a,1)),rewrite([12(6),9(6)])].
32 x / (x / (x / (y / z))) = x / (y / z).  [para(7(a,1),11(a,1,2)),rewrite([7(4),28(5)]),flip(a)].
40 x / (x / (x / y)) = x / y.  [para(11(a,1),11(a,2,2,2)),rewrite([7(5),2(5),9(4),28(3),32(5)]),flip(a)].
46 ((x / y) / z) / x = 1.  [para(12(a,1),7(a,1,1)),rewrite([3(2)]),flip(a)].
47 ((x / y) / z) / (x / z) = 1.  [para(7(a,1),12(a,1,1))].
52 x / (x / ((x / y) / z)) = (x / y) / z.  [para(46(a,1),6(a,1,2)),rewrite([9(4)]),flip(a)].
53 ((x / (x / y)) / z) / y = 1.  [para(6(a,1),46(a,1,1,1))].
54 (x / y) / (x / (y / z)) = 1.  [para(6(a,1),46(a,1,1)),rewrite([7(4)])].
90 ((x / (x / y)) / z) / (y / z) = 1.  [para(6(a,1),47(a,1,1,1))].
113 (x / y) / (x / (z / (z / y))) = 1.  [para(6(a,1),53(a,1,1)),rewrite([7(5)])].
124 (x / (x / y)) / (y / ((y / x) / z)) = 1.  [para(6(a,1),54(a,1,1))].
213 (x / (x / (y / z))) / (y / (z / u)) = 1.  [para(54(a,1),15(a,2,1)),rewrite([3(10)])].
531 ((x / (x / (y / z))) / u) / ((y / u) / z) = 1.  [para(7(a,1),90(a,1,2))].
551 ((x / (x / y)) / (y / z)) / (z / (y / x)) = 1.  [para(15(a,2),90(a,1,1))].
557 ((x / y) / (z / y)) / (x / z) = 1.  [para(90(a,1),16(a,2))].
```

```
583 (x / (x / (y / z))) / (y / (u / (u / z))) = 1.  [para(113(a,1),15(a,2,1)),rewrite([3(11)])].
2967 x / (y / ((y / x) / z)) != 1 | x / u != 1 | z / (z / (y / x)) != u / x | y / ((y / x) / z) = u.  [para(6(a,1),24(c,1))].
3237 x / (y / ((y / x) / z)) = x / y.  [para(124(a,1),14(a,1,2)),rewrite([9(6),54(11),9(7)])].
3280 x / y != 1 | x / z != 1 | u / (u / (y / x)) != z / x | y / ((y / x) / u) = z.  [back_rewrite(2967),rewrite([3237(4)])].
5206 ((x / y) / (z / y)) / (x / (z / u)) = 1.  [para(557(a,1),213(a,1,1,2)),rewrite([9(5)])].
21101 (x / (y / (z / u))) / (x / ((y / w) / (z / w))) = 1.  [para(5206(a,1),113(a,1,2,2,2)),rewrite([9(8)])].
27486 (x / (y / z)) / (x / ((u / (u / y)) / (z / (y / u)))) = 1.  [para(551(a,1),531(a,1,2)),rewrite([7(9),9(11)])].
28777 (x / (x / (y / (z / u)))) / (y / ((z / w) / (u / w))) = 1.  [para(557(a,1),583(a,1,2,2,2)),rewrite([9(9)])].
75243 (x / (x / ((y / (y / z)) / (u / z)))) / (z / (u / y)) = 1.  [para(28777(a,1),27486(a,1,2)),rewrite([9(11)])].
81865 x / ((x / y) / (y / x)) = y / ((y / x) / (x / y). [hyper(3280,a,75243,a,b,21101,a,c,7,a),rewrite([2(1),9(2),2(1),9(2),
  2(1),9(2),2(1),9(2),28(3),2(2),9(3),2(2),9(3),40(4),2(5),9(6),2(5),9(6),40(7),2(6),9(7),2(6),9(7),2(6),9(7),28(8)])].
81872 (x / y) / (y / x) = x / y.  [para(81865(a,1),4(a,2,2)),rewrite([4(4),52(5),3237(8)])].
81873 $F.  [resolve(81872,a,10,a)].

============================== end of proof ==========================
```

Finally, we show that (16)–(20) entail each of the conditions of cBCK algebras with relative cancellation. We show the latter in a convenient order.

- (11) holds as it is the same as (16);
- (14) holds as it is the same as (19);
- (13) follows from (16)–(19) by:

```
============================== PROOF =================================

% Proof 1 at 0.01 (+ 0.00) seconds.
% Length of proof is 10.
% Level of proof is 4.
% Maximum clause weight is 11.000.
% Given clauses 9.

1 1 / x = 1 # label(non_clause) # label(goal).  [goal].
2 x / x = 1.  [assumption].
3 x / 1 = x.  [assumption].
4 (x / y) / z = (x / z) / y.  [assumption].
5 x / (x / y) = y / (y / x).  [assumption].
7 1 / c1 != 1.  [deny(1)].
8 (x / y) / x = 1 / y.  [para(2(a,1),4(a,1,1)),flip(a)].
25 1 / (x / y) = 1.  [para(5(a,1),8(a,1,1)),rewrite([4(3),2(3)]),flip(a)].
29 1 / x = 1.  [para(3(a,1),25(a,1,2))].
30 $F.  [resolve(29,a,7,a)].

============================== end of proof ==========================
```

- (12) follows from (17) and (19);
- (10) follows from (16) and (18);
- (9) follows from (16)–(19) by:

```
============================== PROOF =================================

% Proof 1 at 0.01 (+ 0.00) seconds.
% Length of proof is 11.
% Level of proof is 4.
% Maximum clause weight is 15.000.
% Given clauses 19.

1 ((x / y) / (x / z)) / (z / y) = 1 # label(non_clause) # label(goal).  [goal].
2 x / x = 1.  [assumption].
4 (x / y) / z = (x / z) / y.  [assumption].
5 x / (x / y) = y / (y / x).  [assumption].
7 1 / x = 1.  [assumption].
11 ((c1 / c2) / (c1 / c3)) / (c3 / c2) != 1.  [deny(1)].
12 (x / y) / x = 1.  [para(2(a,1),4(a,1,1)),rewrite([7(2)]),flip(a)].
14 (x / (x / y)) / z = (y / z) / (y / x).  [para(5(a,1),4(a,1,1))].
24 ((x / y) / z) / (x / z) = 1.  [para(4(a,1),12(a,1,1))].
165 ((x / y) / (x / z)) / (z / y) = 1.  [para(14(a,1),24(a,1,1))].
166 $F.  [resolve(165,a,11,a)].

============================== end of proof ==========================
```

- (15) is the only non-trivial condition; only it requires also (20) to hold:

```
============================== PROOF =================================

% Proof 1 at 0.20 (+ 0.01) seconds.
% Length of proof is 27.
% Level of proof is 8.
% Maximum clause weight is 17.000.
% Given clauses 122.

1 x / y != 1 | x / z != 1 | y / x != z / x | y = z # label(non_clause) # label(goal).  [goal].
2 x / x = 1.  [assumption].
3 x / 1 = x.  [assumption].
4 (x / y) / z = (x / z) / y.  [assumption].
5 x / (x / y) = y / (y / x).  [assumption].
6 (x / y) / (y / x) = x / y.  [assumption].
7 1 / x = 1.  [assumption].
8 x / y != 1 | y / x != 1 | x = y.  [assumption].
11 ((x / y) / (x / z)) / (z / y) = 1.  [assumption].
12 c1 / c2 = 1.  [deny(1)].
13 c1 / c3 = 1.  [deny(1)].
14 c3 / c1 = c2 / c1.  [deny(1)].
15 c3 != c2.  [deny(1)].
16 (x / y) / x = 1.  [para(2(a,1),4(a,1,1)),rewrite([7(2)]),flip(a)].
18 (x / (x / y)) / z = (y / z) / (y / x).  [para(5(a,1),4(a,1,1))].
19 (x / y) / ((x / z) / y) = z / (z / (x / y)).  [para(4(a,1),5(a,1,2))].
34 ((x / (x / y)) / (y / z)) / (z / (y / x)) = 1.  [para(5(a,1),11(a,1,1,1))].
47 c3 / (c2 / c1) = c1.  [para(13(a,1),5(a,1,2)),rewrite([3(3),14(5)]),flip(a)].
60 (c2 / c1) / c3 = 1.  [para(14(a,1),16(a,1,1))].
71 (c3 / x) / (c2 / c1) = c1 / x.  [para(47(a,1),4(a,1,1)),flip(a)].
265 c1 / (c1 / (c2 / c3)) = c2 / c3.  [para(60(a,1),19(a,1,2)),rewrite([3(5)]),flip(a)].
```

```
2476 c2 / c3 = 1.  [para(71(a,1),34(a,1,1)),rewrite([4(11),265(7),6(7)])].
2595 c3 / c2 != 1.  [ur(8,b,2476,a,c,15,a)].
2600 (c3 / x) / (c3 / c2) = c2 / x.  [para(2476(a,1),18(a,1,1,2)),rewrite([3(3)]),flip(a)].
3220 c1 / (c3 / c2) = c1.  [para(47(a,1),2600(a,1,1)),rewrite([5(10),12(9),3(8)])].
3268 c3 / c2 = 1.  [para(3220(a,1),5(a,1,2)),rewrite([2(3),4(9),14(7),16(9),3(6)]),flip(a)].
3269 $F.  [resolve(3268,a,2595,a)].

============================== end of proof ==========================
```

◀