The Andrews-Curtis conjecture, term rewriting and first-order proofs

A. Lisitsa

Department of Computer Science, University of Liverpool, Liverpool, UK
A.Lisitsa@liverpool.ac.uk

Abstract. The Andrews-Curtis conjecture (ACC) remains one of the outstanding open problems in combinatorial group theory. In short, it states that every balanced presentation of the trivial group can be transformed into a trivial presentation by a sequence of simple transformations. It is generally believed that the conjecture may be false and there are several series of potential counterexamples for which required simplifications are not known. Finding simplifications poses a challenge for any computational approach - the search space is unbounded and the lower bound on the length of simplification sequences is known to be at least superexponential. Various specialised search algorithms have been used to eliminate some of the potential counterexamples. In this paper we present an alternative approach based on automated reasoning. We formulate a term rewriting system ACT for AC-transformations, and its translation(s) into the first-order logic. The problem of finding AC-simplifications is reduced to the problem of proving first-order formulae, which is then tackled by the available automated theorem provers. We report on the experiments demonstrating the efficiency of the proposed method by finding required simplifications for several new open cases.

1 Introduction

The topic of this paper can be described by two expressions: *applied automated reasoning* and *experimental mathematics*. We show how automated first-order theorem proving and disproving can be used to explore the Andrews-Curtis conjecture (ACC) [2]. This conjecture remains one of the outstanding open problems in combinatorial group theory. In short, it states that every balanced presentation of the trivial group can be transformed into a trivial presentation by a sequence of simple transformations. It is generally believed that the conjecture may be false and there are several series of potential counterexamples for which required simplifications are not known.

For a group presentation $\langle x_1, \ldots, x_n; r_1, \ldots r_m \rangle$ with generators x_i, and relators r_j, consider the following transformations.

AC1 Replace some r_i by r_i^{-1}.
AC2 Replace some r_i by $r_i \cdot r_j$, $j \neq i$.

AC3 Replace some r_i by $w \cdot r_i \cdot w^{-1}$ where w is any word in the generators.
AC4 Re-order the relators.
AC5 Introduce a new generator y and relator y or delete a generator y and relator y.

We notice that AC4 rule is redundant in a sense that its effect can be achieved by an application of a sequence of AC1 and AC2 rules. Indeed, for any two relators r_i and r_j their transposition $\ldots r_i \ldots r_j \ldots \mapsto \ldots r_j, \ldots r_i \ldots$ is the result of the application of the sequence of rules $\mathrm{AC2}_{ij}$ $\mathrm{AC1}_i$ $\mathrm{AC2}_{ji}$ $\mathrm{AC1}_j$ $\mathrm{AC2}_{ij}$ $\mathrm{AC1}_i$. As any permutation is a composition of transpositions the statement follows.

Two presentations g and g' are called *Andrews-Curtis equivalent (AC-equivalent)* if one of them can be obtained from the other by applying a finite sequence of transformations of the types (AC1) - (AC4). Two presentations are *stably AC-equivalent* if one of them can be obtained from the other by applying a finite sequence of transformations of the types (AC1) - (AC5).

A group presentation $g = \langle x_1, \ldots, x_n; r_1, \ldots r_m \rangle$ is called *balanced* if $n = m$, that is a number of generators is the same as a number of relators. Such n we call a *dimension* of g and denote by $Dim(g)$.

Conjecture 1 (Andrews-Curtis [2]).
If $\langle x_1, \ldots, x_n; r_1, \ldots r_n \rangle$ is a balanced presentation of the trivial group it is AC-equivalent to the trivial presentation $\langle x_1, \ldots, x_n; x_1, \ldots x_n \rangle$.

The *weak form* of the conjecture states that every balanced presentation for a trivial group is stably AC-equivalent (i.e. transformations AC5 are allowed) to the trivial presentation.

In what follows we will assume that we are dealing with the strong form of the conjecture unless stated otherwise.

Both variants of the conjecture remain open and challenging problems. According to [4] the prevalent opinion is that the conjecture is false, but no counterexamples have been found so far. There are, however, potential counterexamples and even infinite series of potential counterexamples, which provide an opportunity to use a computational approach to explore the conjecture. Notice, that if the statement of the conjecture holds for a particular presentation this fact can be established, at least in principle, by enumeration and application of all possible sequences of transformations until the trivial presentation is obtained. Then, in principle, one may attack potential counterexamples for AC-conjecture by the automated search of the AC-sequences leading to the trivial presentations (AC-simplifying sequences). Such a search is a computationally difficult and the search space grows exponentially with the length of the sequences. As it was noticed in [18], neither total enumeration, nor random search can be effectively applied here. More efficient search procedure using *genetic algorithms* has been proposed in [18] and it was used to show that a well-known potential counterexample $\langle x, y | xyxy^{-1}x^{-1}y^{-1}, x^2y^{-3} \rangle$ is, in fact, AC-equivalent to the trivial presentation, and by that it is not a counterexample. Further exploration and improvement of genetic approach can be found in [20] and [13] where many new simplifications are presented as well.

In [12] it was shown that a systematic breadth-first search of the tree of equivalent presentations is a viable alternative to genetic algorithms of [18] which allowed to show, in particular, that the potential counterexample

$$\langle x, y | xyxy^{-1}x^{-1}y^{-1}, x^3y^{-4} \rangle$$

is unique up to the AC-equivalence among all balanced presentations of trivial groups with two generators up to the length 13. This counterexample (AK-3) is one of the infinite series of presentations proposed by Akbulut and Kirby [1] and is the smallest for which it is not known whether it is AC-equivalent to trivial presentation. The paper [16] discusses the implementation aspects of the breadth-first search for AC-simplifications on high-performance computer platform using disk-based hash tables. The approach is illustrated by successful search of AC-simplifications for some known non-trivial cases. In [11] an alternative approach for refuting the potential counterexamples based on the methods from computational group theory was proposed. In this approach AC-simplifications are extracted from the results produced by Todd-Coxeter coset enumeration algorithm, by application of ad hoc techniques. The approach has been used to find some non-trivial AC-simplifications.

Lower bound on the length of simplifications is known to be superexponential [7,14]. So the failure to deal AK-3 example by any known computational approach should not be overestimated, we are still exploring very small part of the huge search space.

In this paper we propose an alternative approach for testing the groups presentations as to whether they satisfy the Andrews-Curtis conjecture which is based on use of term-rewriting systems and first-order logic. We formulate the term rewriting system ACT for AC-transformations, and its translations into the first-order logic. The problem of finding AC-simplifications is reduced to the problem of proving first-order formulas, which is then tackled by the available automated theorem provers. We show that the approach is competitive by demonstrating simplifications for a few open cases. An abstract with an announcement of the proposed method and simplifications of known cases has appeared in [15].

2 ACT Term Rewriting Systems

Let T_G be the equational theory of groups defined by the the following equations in a vocabulary $(\cdot, r, e)$:

- $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
- $x \cdot e = x$
- $x \cdot r(x) = e$

For each $n \geq 2$ we formulate a term rewriting system modulo T_G, which captures AC-transformations of presentations of dimension n. We start with dimension $n = 2$.

For an alphabet $A = \{a_1, a_2\}$ a term rewriting system ACT_2 consists the following rules:

R1L $f(x, y) \to f(r(x), y))$
R1R $f(x, y) \to f(x, r(y))$
R2L $f(x, y) \to f(x \cdot y, y)$
R2R $f(x, y) \to f(x, y \cdot x)$
R3L$_i$ $f(x, y) \to f((a_i \cdot x) \cdot r(a_i), y)$ for $a_i \in A, i = 1, 2$
R3R$_i$ $f(x, y) \to f(x, (a_i \cdot y) \cdot r(a_i))$ for $a_i \in A, i = 1, 2$

The term rewriting system ACT_2 gives rise to the rewrite relation $\to_{ACT}$ on the set of all terms defined in the standard way [3]. For terms t_1, t_2 in groups vocabulary we write $t_1 =_G t_2$ if equality $t_1 = t_2$ is derivable in T_G. We extend $=_G$ homomorphically by defining $f(t_1, t_2) =_G f(s_1, s_2)$ iff $t_1 =_G s_1$ and $t_2 =_G s_2$. Denote by $[t]_G$ the equivalence class of t wrt $=_G$, that is $[t]_G = \{t' \mid t =_G t'\}$.

Then rewrite relation $\to_{ACT/G}$ for ACT modulo theory T_G is defined [3] as follows: $t \to_{ACT/G} s$ iff there exist $t' \in [t]_G$ and $s' \in [s]_G$ such that $t' \to_{ACT} s'$.

Claim (on formalization). The notion of rewrite relation $\to_{ACT/G}$ captures adequately the notion of AC-rewriting, as defined in Section 1 that is for presentations p_1 and p_2 we have $p_1 \to^*_{AC} p_2$ iff $t_{p_1} \to^*_{ACT/G}$. Here t_p denotes a term encoding of a presentation p, that is for $p = \langle a_1, a_2 \mid t_1.t_2\rangle$ we have $t_p = f(t_1, t_2)$.

The term rewriting system ACT_2 can be simplified without changing the transitive closure of the rewriting relation. Reduced term rewriting system $rACT_2$ consists of the following rules:

R1L $f(x, y) \to f(r(x), y))$
R2L $f(x, y) \to f(x \cdot y, y)$
R2R $f(x, y) \to f(x, y \cdot x)$
R3L$_i$ $f(x, y) \to f((a_i \cdot x) \cdot r(a_i), y)$ for $a_i \in A, i = 1, 2$

Proposition 1. *Term rewriting systems ACT_2 and $rACT_2$ considered modulo T_G are equivalent, that is $\to^*_{ACT_2/G}$ and $\to^*_{rACT_2/G}$ coincide.*

Proposition 2. *For ground t_1 and t_2 we have $t_1 \to^*_{ACT_2/G} t_2 \Leftrightarrow t_2 \to^*_{ACT_2/G} t_1$, that is $\to^*_{ACT_2/G}$ is symmetric.*

Now we present two variants of translations of ACT_2 into first-order logic with an intention to use automated theorem proving to show AC-equivalence.

2.1 Equational Translation

Denote by E_{ACT_2} an equational theory $T_G \cup rACT^=$ where $rACT^=$ includes the following axioms (equality variants of the above rewriting rules):

E-R1L $f(x, y) = f(r(x), y))$
E-R2L $f(x, y) = f(x \cdot y, y)$
E-R2R $f(x, y) = f(x, y \cdot x)$
E-R3L$_i$ $f(x, y) = f((a_i \cdot x) \cdot r(a_i), y)$ for $a_i \in A, i = 1, 2$

Proposition 3. *For ground terms t_1 and t_2 $t_1 \to^*_{ACT_2/G} t_2$ iff $E_{ACT_2} \vdash t_1 = t_2$*

Proof (sketch) By Proposition 2 $t_1 \to^*_{ACT_2/G} t_2 \Leftrightarrow t_2 \leftrightarrow^*_{ACT_2/G} t_1$. By Birkhoff's theorem [5,19,8] the latter condition is equivalent to $E_{ACT_2} \models t_1 = t_2$ and therefore $E_{ACT_2} \vdash t_1 = t_2$.

In a variant of the equational translation the axioms $\mathbf{E-R3L_i}$ are replaced by "non-ground" axiom $\mathbf{E-RLZ}: f(x,y) = f((z \cdot x) \cdot r(z), y)$ and the corresponding analogue of Proposition 3 holds true.

2.2 Implicational Translation

Denote by I_{ACT_2} the first-order theory $T_G \cup rACT_2^{\to}$ where $rACT_2^{\to}$ includes the following axioms:

I-R1L $R(f(x,y)) \to R(f(r(x),y)))$
I-R2L $R(f(x,y)) \to R(f(x \cdot y, y))$
I-R2R $R(f(x,y)) \to R(f(x, y \cdot x))$
I-R3L$_i$ $R(f(x,y)) \to R(f((a_i \cdot x) \cdot r(a_i), y))$ for $a_i \in A, i = 1, 2$

Proposition 4. *For ground terms t_1 and t_2 $t_1 \to^*_{ACT_2/G} t_2$ iff $I_{ACT_2} \vdash R(t_1) \to R(t_2)$*

Similarly to the case of equational translation "non-ground" axiom **I-R3Z**: $R(f(x,y)) \to R(f((z \cdot x) \cdot r(z), y))$ can be used instead of **I-R3L**$_i$ with a corresponding analogue of Proposition 4 holding true.

2.3 Higher Dimensions

For dimensions $n > 2$ the rewriting systems ACT_n, their reduced versions $rACT_n$, their equational and implicational translations can be formulated such that the analogues of Propositions 3 and 4 hold true. To cut a long story short we show here only an equational translation $rACT_3^{=}$ ("non-ground" variant):

$$
\begin{array}{ll}
f(x,y,z) = f(r(x),y,z) & f(x,y,z) = f(x,r(y),z) \\
f(x,y,z) = f(x,y,r(z)) & f(x,y,z) = f(x \cdot y,y,z) \\
f(x,y,z) = f(x \cdot z,y,z) & f(x,y,z) = f(x,y \cdot x,z) \\
f(x,y,z) = f(x,y \cdot z,z) & f(x,y,z) = f(x,y,z \cdot x) \\
f(x,y,z) = f(x,y,z \cdot y) & f(x,y,z) = f((v \cdot x) \cdot r(v),y,z) \\
f(x,y,z) = f(x,(v \cdot y) \cdot r(v),z) & f(x,y,z) = f(x,y,(v \cdot z) \cdot r(v)).
\end{array}
$$

3 Automated Proving and Disproving for ACC Exploration

Propositions 3 and 4 (and their analogues) suggest a way of using automated reasoning for exploration of ACC. For any concrete pair of presentations p_1 and p_2, to establish whether they are AC-equivalent one can formulate a theorem proving/disproving tasks of the form $E_{ACT_n} \vdash t_{p_1} = t_{p_2}$, or $I_{ACT_n} \vdash R(t_{p_1}) \to R(t_{p_2})$ ($E_{ACT_n} \not\vdash t_{p_1} = t_{p_2}$, or $I_{ACT_n} \not\vdash R(t_{p_1}) \to R(t_{p_2})$).

Unfortunately disproving by finite countermodel model finding has its fundamental limitations in the context of ACC. Based on the results of [6] it cannot be used to disprove ACC. At the same time one can get some non-trivial results on necessity of some of the rules for simplification, both in solved cases and non-solved cases. For example we have:

Proposition 5. *To simplify AK-3 (if at all it is possible) one really needs conjugation with both generators a and b.*

We have used finite model builder Mace4 [17] to build countermodels of sizes 12 and 6 respectively for the cases where either of the conjugation rules was missing.

3.1 Theorem Proving for Simplification

Known Cases We have applied automated theorem proving using Prover9 prover[17] to confirm that all cases eliminated as potential counterexamples in [16,12,18,11,13] can be eliminated by our method too.

New Cases Using automated theorem proving we were able to eliminate the following potential counterexamples for ACC, which are all *irreducible cyclically presented groups* [10] whose status was open to the best of our knowledge [13,10,9]. We use notation of [9] to refer to these examples. We also follow the standard convention to use capital letters $A, B, C\dots$ to denote inverse of $a, b, c, \dots$ respectively.

Dim = 2
- **T14** $\langle a, b \mid ababABB, babaBAA\rangle$
- **T28** $\langle a, b \mid aabbbbABBBB, bbaaaaBAAAA\rangle$
- **T36** $\langle a, b \mid aababAABB, bbabaBBAA\rangle$
- **T62** $\langle a, b \mid aaabbAbABBB, bbbaaBaBAAA\rangle$
- **T74** $\langle a, b \mid aabaabAAABB, bbabbaBBBAA\rangle$

Dim = 3
- **T16** $\langle a, b, c \mid ABCacbb, BCAbacc, CABcbaa\rangle$
- **T21** $\langle a, b, c \mid ABCabac, BCAbcba, CABcacb\rangle$
- **T48** $\langle a, b, c \mid aacbcABCC, bbacaBCAA, ccbabCABB\rangle$
- **T88** $\langle a, b, c \mid aacbAbCAB, bbacBcABC, ccbaCaBCA\rangle$
- **T89** $\langle a, b, c \mid aacbcACAB, bbacBABC, ccbaCBCA\rangle$

Dim = 4
- **T96** $\langle a, b, c, d \mid adCADbc, baDBAcd, cbACBda, dcBDCab\rangle$
- **T97** $\langle a, b, c, d \mid adCAbDc, baDBcAd, cbACdBa, dcBDaCb\rangle$

We were able to prove corresponding formulas in both equational and (variants of) implicational translations. The proofs for implicational translations are more transparent and more amenable for simplifying transformations extractions. The proofs generated by Prover9 for implicational translations are essentially sequences of atomic formulas of the from $R(r_1, r_2)$ (for Dim =2) which encompass simplification sequences of presentations $\langle a, b \mid r_1,\ r_2\rangle$. All such atomic formulas produced with the references to the applied clauses which encode particular rules from (AC1)-(AC3). In the Appendix we show a simplification extracted manually from the proof for T16 (Dim = 3) presentation.

4 Conclusion

As it was noticed in [18] neither total enumeration, nor random search can be effectively applied to disproving the Andrews-Curtis conjecture. We have shown in this paper that systematic, goal-oriented search implemented in automated theorem proving procedures provides an interesting and viable alternative.

Furthermore, although finite model finding can not be used directly to disprove AC-conjecture, it can be a tool for establishing non-derivability for sybsystems of transformations.

We have published all computer-generated proofs online[1].

References

1. Selman Akbulut and Robion Kirby. A potential smooth counterexample in dimension 4 to the Poincare conjecture, the Schoenflies conjecture, and the Andrews-Curtis conjecture. *Topology*, 24(4):375–390, 1985.
2. J. Andrews and M.L. Curtis. Free groups and handlebodies. *Proc. Amer. Math. Soc.*, 16:192–195, 1965.
3. Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, New York, NY, USA, 1998.
4. Gilbert Baumslag, Alexei G. Myasnikov, and Vladimir Shpilrain. *Open problems in combinatorial group theory. Second edition*, volume 296, pages 1–38. Amer. Math. Soc., Providence, RI, 2002.
5. Garrett Birkhoff. On the structure of abstract algebras. In *Mathematical proceedings of the Cambridge philosophical society*, volume 31, pages 433–454. Cambridge Univ Press, 1935.
6. Alexandre V. Borovik, Alexander Lubotzky, and Alexei G. Myasnikov. *The Finitary Andrews-Curtis Conjecture*, pages 15–30. Birkhäuser Basel, Basel, 2005.
7. M. R. Bridson. The complexity of balanced presentations and the Andrews-Curtis conjecture. *ArXiv e-prints*, April 2015.
8. Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In Jan van Leeuwen, editor, *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*, pages 243–320. Elsevier, Amsterdam, 1990.
9. Martin Edjvet and Jerry Swan. Irreducible cyclically presented groups. `https://www.maths.nottingham.ac.uk/personal/pmzme/Irreducible-Cyclically-Presented-Groups.pdf`, 2005–2010.

[1] https://zenodo.org/record/1248986
DOI: 10.5281/zenodo.1248986

10. Martin Edjvet and Jerry Swan. On irreducible cyclic presentations of the trivial group. *Experimental Mathematics*, 23(2):181–189, 2014.
11. George Havas and Colin Ramsay. Andrews-Curtis and Todd-Coxeter proof words. Technical report, in Oxford. Vol. I, London Math. Soc. Lecture Note Ser, 2001.
12. George Havas and Colin Ramsay. Breadth-first search and the Andrews-Curtis conjecture. *International Journal of Algebra and Computation*, 13(01):61–68, 2003.
13. Krzysztof Krawiec and Jerry Swan. Ac-trivialization proofs eliminating some potential counterexamples to the andrews-curtis conjecture. `www.cs.put.poznan.pl/kkrawiec/wiki/uploads/Site/ACsequences.pdf`, 2015.
14. B. Lishak. Balanced finite presentations of the trivial group. *ArXiv e-prints*, April 2015.
15. Alexei Lisitsa. First-order theorem proving in the exploration of Andrews-Curtis conjecture. *TinyToCS*, 2, 2013.
16. Stephen B. McCaul and R. Sean Bowman. Fast searching for Andrews-Curtis trivializations. *Experimental Mathematics*, 2006:193–197, 2006.
17. W. McCune. Prover9 and mace4. `http://www.cs.unm.edu/~mccune/prover9/`, 2005–2010.
18. Alexei D. Miasnikov. Genetic algorithms and the Andrews-Curtis conjecture. *International Journal of Algebra and Computation*, 09(06):671–686, 1999.
19. David A. Plaisted. In Dov M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming (Vol. 1)*, chapter Equational Reasoning and Term Rewriting Systems, pages 274–364. Oxford University Press, Inc., New York, NY, USA, 1993.
20. Jerry Swan, Gabriela Ochoa, Graham Kendall, and Martin Edjvet. Fitness landscapes and the andrews-curtis conjecture. *IJAC*, 22(2), 2012.

Appendix

4.1 Technical details

We used Prover9 and Mace4 version 0.5 (December 2007) [17] and one of two system configurations:

A) AMD A6-3410MX APU 1.60Ghz, RAM 4 GB, Windows 7 Enterprise

B) Intel(R) Core(TM) i7-4790 CPU 3.60Ghz, RAM 32 GB, Windows 7 Enterprise

Table 1. Time to prove simplifications for system configuration B)

	T14	**T28**	**T36**	**T62**	**T74**	**T16**	**T21**	**T48**	**T88**	**T89**	**T96**	**97**
Dim	2	2	2	2	2	3	3	3	3	3	4	4
Equational	6.02s	6.50s	7.18s	24.34s	57.17s	12.87s	11.98s	34.63s	57.69s	17.50s	114.05s	115.10s
Implicational	1.57s	2.46s	1.34s	22.50s	6.29s	1.61s	1.45s	2.17s	1.97s	2.14s	102.34s	89.65s
Implicational GC	t/o	t/o	t/o	t/o	t/o	3.76s	1.61s	t/o	0.86s	0.75s	t/o	t/o

“t/o” stands for timeout in 200s; “GC” means encoding with ground conjugation rules; all other encodings are with non-ground conjugation rules.

4.2 AC-trivialization for T16

Initial presentation:

$\langle a, b, c \mid ABCacbb, BCAbacc, CABcbaa\rangle$

Simplification:

$\langle ABCacbb, BCAbacc, CABcbaa\rangle$

$\xrightarrow{x,y,z\to x,y,azA} \langle ABCacbb, BCAbacc, aCABcba\rangle$

$\xrightarrow{x,y,z\to x,y,zx} \langle ABCacbb, BCAbacc, aCABacbb\rangle$

$\xrightarrow{x,y,z\to x,y,bzB} \langle ABCacbb, BCAbacc, baCABacb\rangle$

$\xrightarrow{x,y,z\to x,y,zy} \langle ABCacbb, BCAbacc, bac\rangle$

$\xrightarrow{x,y,z\to x,y,czC} \langle ABCacbb, BCAbacc, cba\rangle$

$\xrightarrow{x,y,z\to x',y,z} \langle BBCAcba, BCAbacc, cba\rangle$

$\xrightarrow{x,y,z\to x,y,z'} \langle BBCAcba, BCAbacc, ABC\rangle$

$\xrightarrow{x,y,z\to xz,y,z} \langle BBCA, BCAbacc, ABC\rangle$

$\xrightarrow{x,y,z\to x',y,z} \langle acbb, BCAbacc, ABC\rangle \xrightarrow{x,y,z\to x,y,z'} \langle acbb, BCAbacc, cba\rangle$

$\xrightarrow{x,y,z\to x,y,azA} \langle acbb, BCAbacc, acb\rangle \xrightarrow{x,y,z\to x,y,z'} \langle acbb, BCAbacc, BCA\rangle$

$\xrightarrow{x,y,z\to x,y,zx} \langle acbb, BCAbacc, b\rangle \xrightarrow{x,y,z\to x,y,z'} \langle acbb, BCAbacc, B\rangle$

$\xrightarrow{x,y,z\to xz,y,z} \langle acb, BCAbacc, B\rangle \xrightarrow{x,y,z\to xz,y,z} \langle ac, BCAbacc, B\rangle$

$\xrightarrow{x,y,z\to x,y',z} \langle ac, CCABacb, B\rangle \xrightarrow{x,y,z\to x,yz,z} \langle ac, CCABac, B\rangle$

$\xrightarrow{x,y,z\to x,y',z} \langle ac, CAbacc, B\rangle \xrightarrow{x,y,z\to x,y,z'} \langle ac, CAbacc, b\rangle$

$\xrightarrow{x,y,z\to x',y,z} \langle CA, CAbacc, b\rangle$

$\xrightarrow{x,y,z\to x,yx,z} \langle CA, CAbacA, b\rangle \xrightarrow{x,y,z\to x,y',z} \langle CA, aCABac, b\rangle$

$\xrightarrow{x,y,z\to x,yx,z} \langle CA, aCAB, b\rangle \xrightarrow{x,y,z\to x,yz,z} \langle CA, aCA, b\rangle$

$\xrightarrow{x,y,z\to x',y,z} \langle ac, aCA, b\rangle \xrightarrow{x,y,z\to x,yx,z} \langle ac, a, b\rangle$

$\xrightarrow{x,y,z\to x,y',z} \langle ac, A, b\rangle \xrightarrow{x,y,z\to x,yx,z} \langle ac, c, b\rangle$

$\xrightarrow{x,y,z\to x,y',z} \langle ac, C, b\rangle \xrightarrow{x,y,z\to xy,y,z} \langle a, C, b\rangle$

$\xrightarrow{x,y,z\to x,yz,z} \langle a, Cb, b\rangle \xrightarrow{x,y,z\to x,y',z} \langle a, Bc, b\rangle$

$\xrightarrow{x,y,z\to x,y,zy} \langle a, Bc, c\rangle \xrightarrow{x,y,z\to x,y,z'} \langle a, Bc, C\rangle$

$\xrightarrow{x,y,z\to x,yz,z} \langle a, B, C\rangle \xrightarrow{x,y,z\to x,y,z'} \langle a, B, c\rangle$

$\xrightarrow{x,y,z\to x,y',z} \langle a, b, c\rangle$