Create account Log in

Article Talk Read Edit View history Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Français
日本語
Edit links

Knuth–Bendix completion algorithm

From Wikipedia, the free encyclopedia

The **Knuth–Bendix completion algorithm** (named after Donald Knuth and Peter Bendix[1]) is a semi-decision[2][3] algorithm for transforming a set of equations (over terms) into a confluent term rewriting system. When the algorithm succeeds, it effectively solves the word problem for the specified algebra.

Buchberger's algorithm for computing Gröbner bases is a very similar algorithm. Although developed independently, it may also be seen as the instantiation of Knuth–Bendix algorithm in the theory of polynomial rings.

Contents [hide]

1 Introduction
2 Rules
3 Example
4 String rewriting systems in group theory
4.1 Motivation in group theory
4.2 Description of the algorithm for finitely presented monoids
4.3 Examples
4.3.1 A terminating example
4.3.2 A non-terminating example
5 Generalizations
6 References
7 External links

Introduction [edit]

For a set E of equations, its **deductive closure** $(\overset{*}{\leftrightarrow}_E)$ is the set of all equations that can be derived by applying equations from E in any order. Formally, E is considered a binary relation, $(\rightarrow_E)$ is its rewrite closure, and $(\overset{*}{\leftrightarrow}_E)$ is the equivalence closure of $(\rightarrow_E)$. For a set R of rewrite rules, its **deductive closure** $(\overset{*}{\rightarrow}_R \circ \overset{*}{\leftarrow}_R)$ is the set of all equations than can be confirmed by applying rules from R left-to-right to both sides until they are literally equal. Formally, R is again viewed as binary relation, $(\rightarrow_R)$ is its rewrite closure, $(\leftarrow_R)$ is its converse, and $(\overset{*}{\rightarrow}_R \circ \overset{*}{\leftarrow}_R)$ is the relation composition of their reflexive transitive closures $(\overset{*}{\rightarrow}_R$ and $\overset{*}{\leftarrow}_R)$.

For example, if $E = \{ 1{\cdot}x = x,\ x^{-1}{\cdot}x = 1,\ (x{\cdot}y){\cdot}z = x{\cdot}(y{\cdot}z) \}$ are the group axioms, the derivation chain

$$a^{-1}{\cdot}(a{\cdot}b) \quad \overset{*}{\leftrightarrow}_E \quad (a^{-1}{\cdot}a){\cdot}b \quad \overset{*}{\leftrightarrow}_E \quad 1{\cdot}b \quad \overset{*}{\leftrightarrow}_E \quad b$$

demonstrates that $a^{-1}{\cdot}(a{\cdot}b) \overset{*}{\leftrightarrow}_E b$ is a member of E's deductive closure. If $R = \{ 1{\cdot}x \rightarrow x,\ x^{-1}{\cdot}x \rightarrow 1,\ (x{\cdot}y){\cdot}z \rightarrow x{\cdot}(y{\cdot}z) \}$ is a "rewrite rule" version of E, the derivation chains

$$(a^{-1}{\cdot}a){\cdot}b \quad \overset{*}{\rightarrow}_R \quad 1{\cdot}b \quad \overset{*}{\rightarrow}_R \quad b \qquad \text{and} \qquad b \quad \overset{*}{\leftarrow}_R \quad b{\cdot}1$$

demonstrate that $(a^{-1}{\cdot}a){\cdot}b \overset{*}{\rightarrow}_R \circ \overset{*}{\leftarrow}_R b{\cdot}1$ is a member of R's deductive closure. However, there is no way to derive $a^{-1}{\cdot}(a{\cdot}b) \overset{*}{\rightarrow}_R \circ \overset{*}{\leftarrow}_R b$ similar to above, since a right-to-left application of the rule $(x{\cdot}y){\cdot}z \rightarrow x{\cdot}(y{\cdot}z)$ is not allowed.

The Knuth–Bendix algorithm takes a set E of equations between terms, and a reduction ordering ($>$) on the set of all terms, and attempts to construct a confluent and terminating term rewriting system R that has the same deductive closure as E. While proving consequences from E often requires human intuition, proving consequences from R does not. For more details, see Confluence (abstract rewriting)#Motivating examples, which gives an example proof from group theory, performed both using E and using R.

Rules [edit]

Given a set E of equations between terms, the following inference rules can be used to transform it into an equivalent convergent term rewrite system (if possible):[4][5] They are based on a user-given reduction ordering ($>$) on the set of all terms; it is lifted to a well-founded ordering ($\triangleright$) on the set of rewrite rules by defining $(s \rightarrow t)$

▹ $(l \to r)$ if

- $s >_e l$, or
- s and l are literally similar and $t > r$.

Delete	$\langle E \cup \{s = s\}$	$, R \rangle$	$\vdash$	$\langle E$	$, R \rangle$	
Compose	$\langle E$	$, R \cup \{s \to t\} \rangle$	$\vdash$	$\langle E$	$, R \cup \{s \to u\} \rangle$	if $t \to_R u$
Simplify	$\langle E \cup \{s = t\}$	$, R \rangle$	$\vdash$	$\langle E \cup \{s = u\}$	$, R \rangle$	if $t \to_R u$
Orient	$\langle E \cup \{s = t\}$	$, R \rangle$	$\vdash$	$\langle E$	$, R \cup \{s \to t\} \rangle$	if $s > t$
Collapse	$\langle E$	$, R \cup \{s \to t\} \rangle$	$\vdash$	$\langle E \cup \{u = t\}$	$, R \rangle$	if $s \to_R u$ by $l \to r$ with $(s \to t)$ ▹ $(l \to r)$
Deduce	$\langle E$	$, R \rangle$	$\vdash$	$\langle E \cup \{s = t\}$	$, R \rangle$	if (s,t) is a critical pair of R

Example [edit]

The following example run, obtained from the E theorem prover, computes a completion of the (additive) group axioms as in Knuth, Bendix (1970). It starts with the three initial equations for the group (neutral element 0, inverse elements, associativity), using `f(X,Y)` for $X+Y$, and `i(X)` for $-X$. The 10 equations marked with "final" represent the resulting convergent rewrite system. "pm" is short for "paramodulation", implementing *deduce*. Critical pair computation is an instance of paramodulation for equational unit clauses. "rw" is rewriting, implementing *compose*, *collapse*, and *simplify*. Orienting of equations is done implicitly and not recorded.

```
      1 :  : [++equal(f(X1,0), X1)] : initial("GROUP.lop", at_line_9_column_1)
      2 :  : [++equal(f(X1,i(X1)), 0)] : initial("GROUP.lop", at_line_12_column_1)
      3 :  : [++equal(f(f(X1,X2),X3), f(X1,f(X2,X3)))] : initial("GROUP.lop",
at_line_15_column_1)
      5 :  : [++equal(f(X1,X2), f(X1,f(0,X2)))] : pm(3,1)
      6 :  : [++equal(f(X1,f(X2,i(f(X1,X2)))), 0)] : pm(2,3)
      7 :  : [++equal(f(0,X2), f(X1,f(i(X1),X2)))] : pm(3,2)
     27 :  : [++equal(f(X1,0), f(0,i(i(X1))))] : pm(7,2)
     36 :  : [++equal(X1, f(0,i(i(X1))))] : rw(27,1)
     46 :  : [++equal(f(X1,X2), f(X1,i(i(X2))))] : pm(5,36)
     52 :  : [++equal(f(0,X1), X1)] : rw(36,46)
     60 :  : [++equal(i(0), 0)] : pm(2,52)
     63 :  : [++equal(i(i(X1)), f(0,X1))] : pm(46,52)
     64 :  : [++equal(f(X1,f(i(X1),X2)), X2)] : rw(7,52)
     67 :  : [++equal(i(i(X1)), X1)] : rw(63,52)
     74 :  : [++equal(f(i(X1),X1), 0)] : pm(2,67)
     79 :  : [++equal(f(0,X2), f(i(X1),f(X1,X2)))] : pm(3,74)
     83 :  : [++equal(X2, f(i(X1),f(X1,X2)))] : rw(79,52)
    134 :  : [++equal(f(i(X1),0), f(X2,i(f(X1,X2))))] : pm(83,6)
    151 :  : [++equal(i(X1), f(X2,i(f(X1,X2))))] : rw(134,1)
    165 :  : [++equal(f(i(X1),i(X2)), i(f(X2,X1)))] : pm(83,151)
    239 :  : [++equal(f(X1,0), X1)] : 1 : 'final'
    240 :  : [++equal(f(X1,i(X1)), 0)] : 2 : 'final'
    241 :  : [++equal(f(f(X1,X2),X3), f(X1,f(X2,X3)))] : 3 : 'final'
    242 :  : [++equal(f(0,X1), X1)] : 52 : 'final'
    243 :  : [++equal(i(0), 0)] : 60 : 'final'
    244 :  : [++equal(i(i(X1)), X1)] : 67 : 'final'
    245 :  : [++equal(f(i(X1),X1), 0)] : 74 : 'final'
    246 :  : [++equal(f(X1,f(i(X1),X2)), X2)] : 64 : 'final'
    247 :  : [++equal(f(i(X1),f(X1,X2)), X2)] : 83 : 'final'
    248 :  : [++equal(i(f(X1,X2)), f(i(X2),i(X1)))] : 165 : 'final'
```

See also Word problem (mathematics) for another presentation of this example.

String rewriting systems in group theory [edit]

An important case in computational group theory are string rewriting systems which can be used to give canonical labels to elements or cosets of a finitely presented group as products of the generators. This special case is the focus of this section.

Motivation in group theory [edit]

The critical pair lemma states that a term rewriting system is locally confluent (or weakly confluent) if and only if all its critical pairs are convergent. Furthermore, we have Newman's lemma which states that if an (abstract) rewriting system is strongly normalizing and weakly confluent, then the rewriting system is confluent. So, if we can add rules to the term rewriting system in order to force all critical pairs to be convergent while maintaining the strong normalizing property, then this will force the resultant rewriting system to be confluent.

Consider a finitely presented monoid $M = \langle X \mid R \rangle$ where X is a finite set of generators and R is a set of defining relations on X. Let X^* be the set of all words in X (i.e. the free monoid generated by X). Since the relations R generate an equivalence relation on X*, one can consider elements of M to be the equivalence classes of X^* under R. For each class *{w_1, w_2, ... }* it is desirable to choose a standard representative w_k. This representative is called the **canonical** or **normal form** for each word w_k in the class. If there is a computable method to determine for each w_k its normal form w_i then the word problem is easily solved. A confluent rewriting system allows one to do precisely this.

Although the choice of a canonical form can theoretically be made in an arbitrary fashion this approach is generally not computable. (Consider that an equivalence relation on a language can produce an infinite number of infinite classes.) If the language is well-ordered then the order < gives a consistent method for defining minimal representatives, however computing these representatives may still not be possible. In particular, if a rewriting system is used to calculate minimal representatives then the order < should also have the property:

A < B → XAY < XBY for all words A,B,X,Y

This property is called **translation invariance**. An order that is both translation-invariant and a well-order is called a **reduction order**.

From the presentation of the monoid it is possible to define a rewriting system given by the relations R. If A x B is in R then either A < B in which case B → A is a rule in the rewriting system, otherwise A > B and A → B. Since < is a reduction order a given word W can be reduced W > W_1 > ... > W_n where W_n is irreducible under the rewriting system. However, depending on the rules that are applied at each $W_i \to W_{i+1}$ it is possible to end up with two different irreducible reductions $W_n \neq W'_m$ of W. However, if the rewriting system given by the relations is converted to a confluent rewriting system via the Knuth–Bendix algorithm, then all reductions are guaranteed to produce the same irreducible word, namely the normal form for that word.

Description of the algorithm for finitely presented monoids [edit]

Suppose we are given a presentation $\langle X \mid R \rangle$, where X is a set of generators and R is a set of relations giving the rewriting system. Suppose further that we have a reduction ordering $<$ among the words generated by X(e.g., shortlex order). For each relation $P_i = Q_i$ in R, suppose $Q_i < P_i$. Thus we begin with the set of reductions $P_i \to Q_i$.

First, if any relation $P_i = Q_i$ can be reduced, replace P_i and Q_i with the reductions.

Next, we add more reductions (that is, rewriting rules) to eliminate possible exceptions of confluence. Suppose that P_i and P_j, where $i \neq j$, overlap.

1. Case 1: either the prefix of P_i equals the suffix of P_j, or vice versa. In the former case, we can write $P_i = BC$ and $P_j = AB$; in the latter case, $P_i = AB$ and $P_j = BC$.
2. Case 2: either P_i is completely contained (surrounded) in P_j, or vice versa. In the former case, we can write $P_i = B$ and $P_j = ABC$; in the latter case, $P_i = ABC$ and $P_j = B$.

Reduce the word ABC using P_i first, then using P_j first. Call the results r_1, r_2, respectively. If $r_1 \neq r_2$, then we have an instance where confluence could fail. Hence, add the reduction $\max r_1, r_2 \to \min r_1, r_2$ to R.

After adding a rule to R, remove any rules in R that might have reducible left sides.

Repeat the procedure until all overlapping left sides have been checked.

Examples [edit]

A terminating example [edit]

Consider the monoid: $\langle x, y \mid x^3 = y^3 = (xy)^3 = 1 \rangle$. We use the shortlex order. This is an infinite monoid but nevertheless, the Knuth–Bendix algorithm is able to solve the word problem.

Our beginning three reductions are therefore (1) $x^3 \to 1$, (2) $y^3 \to 1$, and (3) $(xy)^3 \to 1$.

Consider the word x^3yxyxy. Reducing using (1), we get $yxyxy$. Reducing using (3), we get x^2. Hence, we get $yxyxy = x^2$, giving the reduction rule (4) $yxyxy \to x^2$.

Similarly, using $xyxyxy^3$ and reducing using (2) and (3), we get $xyxyx = y^2$. Hence the reduction (5) $xyxyx \to y^2$.

Both of these rules obsolete (3), so we remove it.

Next, consider x^3yxyx by overlapping (1) and (5). Reducing we get $yxyx = x^2y^2$, so we add the rule (6) $yxyx \to x^2y^2$. Considering $xyxyx^3$ by overlapping (1) and (5), we get $xyxy = y^2x^2$, so we add the rule (7) $y^2x^2 \to xyxy$. These obsolete rules (4) and (5), so we remove them.

Now, we are left with the rewriting system

- (1) $x^3 \to 1$
- (2) $y^3 \to 1$
- (6) $yxyx \to x^2y^2$
- (7) $y^2x^2 \to xyxy$

Checking the overlaps of these rules, we find no potential failures of confluence. Therefore, we have a confluent rewriting system, and the algorithm terminates successfully.

A non-terminating example [edit]

The order of the generators may crucially affect whether the Knuth–Bendix completion terminates. As an example, consider the free Abelian group by the monoid presentation:

$$\langle x, y, x^{-1}, y^{-1} \,|\, xy = yx, xx^{-1} = x^{-1}x = yy^{-1} = y^{-1}y = 1 \rangle.$$

The Knuth–Bendix completion with respect to lexicographic order $x < x^{-1} < y < y^{-1}$ finishes with a convergent system, however considering the length-lexicographic order $x < y < x^{-1} < y^{-1}$ it does not finish for there are no finite convergent systems compatible with this latter order.[6]

Generalizations [edit]

If Knuth–Bendix does not succeed, it will either run forever, or fail when it encounters an unorientable equation (i.e. an equation that it cannot turn into a rewrite rule). The enhanced completion without failure will not fail on unorientable equations and provides a semi-decision procedure for the word problem.

The notion of logged rewriting discussed in the paper by Heyworth and Wensley listed below allows some recording or logging of the rewriting process as it proceeds. This is useful for computing identities among relations for presentations of groups.

References [edit]

1. ^ D. Knuth, "The Genesis of Attribute Grammars"
2. ^ Jacob T. Schwartz; Domenico Cantone; Eugenio G. Omodeo; Martin Davis (2011). *Computational Logic and Set Theory: Applying Formalized Logic to Analysis*. Springer Science & Business Media. p. 200. ISBN 978-0-85729-808-9.
3. ^ Hsiang, J.; Rusinowitch, M. (1987). "On word problems in equational theories". *Automata, Languages and Programming*. Lecture Notes in Computer Science **267**. p. 54. doi:10.1007/3-540-18088-5_6. ISBN 978-3-540-18088-3., p. 55
4. ^ Bachmair, L., Dershowitz, N., Hsiang, J. (Jun 1986). "Orderings for Equational Proofs". *Proc. IEEE Symposium on Logic in Computer Science*. pp. 346–357.
5. ^ N. Dershowitz, J.-P. Jouannaud (1990). Jan van Leeuwen, ed. *Rewrite Systems*. Handbook of Theoretical Computer Science **B**. Elsevier. pp. 243–320. Here: sect.8.1, p.293
6. ^ V. Diekert, A.J. Duncan, A.G. Myasnikov (2011). "Geodesic Rewriting Systems and Pregroups". In Oleg Bogopolski, Inna Bumagin, Olga Kharlampovich, Enric Ventura. *Combinatorial and Geometric Group Theory: Dortmund and Ottawa-Montreal conferences*. Springer Science & Business Media. p. 62. ISBN 978-3-7643-9911-5.

- D. Knuth, P. Bendix (1970). J. Leech, ed. *Simple Word Problems in Universal Algebras*. Pergamon Press. pp. 263–297.
- Gérard Huet (1981). "A Complete Proof of Correctness of the Knuth-Bendix Completion Algorithm" (PDF). *J. Comput. System Sci.* **23** (1): 11–21. doi:10.1016/0022-0000(81)90002-7.
- C. Sims. 'Computations with finitely presented groups.' Cambridge, 1994.
- Anne Heyworth and C.D. Wensley. "Logged rewriting and identities among relators." *Groups St. Andrews 2001 in Oxford. Vol. I,* 256–276, London Math. Soc. Lecture Note Ser., 304, Cambridge Univ. Press,

Cambridge, 2003.

External links [edit]

- Weisstein, Eric W., "Knuth–Bendix Completion Algorithm", *MathWorld*.

v · t · e	Donald Knuth [hide]
Publications	*The Art of Computer Programming* · "The Complexity of Songs" · *Computers and Typesetting* · *Concrete Mathematics* · *Surreal Numbers* · *Things a Computer Scientist Rarely Talks About* · *Selected papers series*
Software	TeX · METAFONT · MIXAL (MIX · MMIX · GNU MDK)
Fonts	AMS Euler · Computer Modern · Concrete Roman
Literate programming	WEB · CWEB
Algorithms	Knuth's Algorithm X · **Knuth–Bendix completion algorithm** · Knuth–Morris–Pratt algorithm · Knuth shuffle · Robinson–Schensted–Knuth correspondence · Trabb Pardo–Knuth algorithm · Generalization of Dijkstra's algorithm · Knuth's Simpath algorithm
Other	Dancing Links · Knuth reward check · Knuth Prize · Man or boy test · Quater-imaginary base · -yllion · Potrzebie system of weights and measures

Categories: Computational group theory | Donald Knuth | Combinatorics on words | Rewriting systems