ELSEVIER

Theoretical Computer Science 207 (1998) 13–23

Theoretical Computer Science

Contributions of Ronald V. Book to the theory of string-rewriting systems [1]

Robert McNaughton *,[2]

Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA

Abstract

Keywords: Church–Rosser property; Monoid presentation; Semi-Thue systems; String rewriting; Thue systems

1. Introduction

Ron Book's interest in string-rewriting systems was stimulated by Maurice Nivat [12], who, in the 1970s, investigated Thue systems [15] and semi-Thue systems for applications to formal languages and algebra. The collection of research problems that Book was to focus on in the 1980s was, to a large extent, an outgrowth of the collection of problems that Nivat and his collaborators had focused on in the 1970s (see Berstell's 1977 paper [1]).

During most of the 1980s Book was intensively interested in research in this area. He is to be lauded for carrying out his research on a broad front, maintaining an interest in several different research questions, developing his own thoughts and paying careful attention to the results of others. He had many research collaborators, including several doctoral students and people who spent some fruitful post-doctoral years at Santa Barbara. He was, in effect, the leader of a group that included all or most of these. Part of our appreciation of the impact that he had on the field of rewriting systems was what these students and post-docs went on to do after they left Santa Barbara. I would like to interject a personal remark at this point and mention how much I have gained from this group. I have profited not only from the clear research

* E-mail: mcnaught@cs.rpi.edu.

[1] This review was written for a celebration of Book's sixtieth birthday, held at the University of Minnesota on April 12, 1997. I reget that I could not complete it in time for it to appear in the Festschrift volume (edited by Ding-Zhu Du and Ker-I Ko).

[2] Supported by Grant No. CCR-9500182 from the National Science Foundation.

PII S0304-3975(98)00053-X

orientation that Book has provided, but also from the contact I have had with him and with those who have acquired this orientation from him.

Book was joined by Friedrich Otto in 1993 in writing a monograph [8] that has a fairly complete account of this area of research, including most of Book's contributions. Because of its importance, I shall often refer to it informally rather than by its location in the list of references at the end of the review.

The plan for the remainder of this review is to look at various research questions as Book and his collaborators originally posed them, and, in some cases, to trace their history. Readers who want more technical detail will find most of what they want in the monograph. This short review is not intended to serve either as a complete survey or as a technical introduction. I regret that time has not permitted even a mention of the work of most of his followers.

2. Thue systems [15] and semi-Thue systems

These are the two basic abstract concepts used in the study of string rewriting, and are presented here briefly. From them a *mixed system* is defined, which is a mixture of a Thue system and a semi-Thue system related in a certain way. The concept of *mixed system*, offered as an explication of the concept of *string rewriting system*, is not found outside this review; it is used to explain ideas in the evolving literature.

A Thue system is an ordered pair (Σ, Q), where Σ is a finite alphabet and Q is a set of unordered pairs of strings over Σ. The set Q, which is usually finite, is called the "set of rules." For $(y_1, y_2) \in Q$ and $x, z \in \Sigma^*$ one writes $xy_1z \leftrightarrow xy_2z$ and $xy_2z \leftrightarrow xy_1z$; thus the rules are symmetric. One writes $x \leftrightarrow^* y$ to assert the existence of a sequence $x_0 = x, x_1, \ldots, x_p = y (p \geqslant 0)$ such that for each $i \leqslant p-1$, $x_i \leftrightarrow x_{i+1}$. When $x \leftrightarrow^* y$ holds one says that x and y are *equivalent*.

A *semi-Thue system* is an ordered pair (Σ, Q) for Σ a finite alphabet and Q a set of *ordered* pairs of strings. Again Q is usually finite, but the rules of Q are not necessarily symmetric. When $(y_1, y_2) \in Q$ and $x, z \in \Sigma^*$, one now writes $xy_1z \rightarrow xy_2z$, but not $xy_2z \rightarrow xy_1z$. And one writes $x \rightarrow^* y$ to assert the existence of a sequence $x_0, x_1, \ldots x_p$ $(p \geqslant 0)$ such that, for each $i \leqslant p-1$, $x_i \rightarrow x_{i+1}$. When $x \rightarrow^* y$ holds one says various things, e.g., "y is derivable from x" and "x reduces to y."

A string-rewriting system frequently involves a semi-Thue system; $x \rightarrow y$ means "x is (or can be) rewritten as y." But rewriting in practice is mostly of two kinds, reduction and generation. If $x \rightarrow y$ is a reduction then y is somehow simpler or smaller than x, e.g., $|y| < |x|$. If it is generative then y is generally more complex or larger than x, e.g., $|y| > |x|$. (The notation $|x|$ means the length of the string x.) In the rewriting literature, $x \rightarrow y$ frequently means "x reduces to y in one step." In Book's papers, it frequently implies that $|y| < |x|$, and sometimes merely that $|y| \leqslant |x|$.

Thue systems are important as presentations of monoids, in which the derivation of an equivalence $x \leftrightarrow^* y$ is a proof that x and y represent the same object in the

monoid. Thus Thue systems are in essence the basis of combinatorial monoid theory (of which combinatorial group theory is a well known subtheory).

A *mixed system* is an ordered triple (Σ, E, R) in which (Σ, E) is a Thue system, (Σ, R) is a semi-Thue system, and (defining $\leftrightarrow$ from E and $\rightarrow$ from R as above) $x \rightarrow y$ implies $x \leftrightarrow^* y$. In a mixed system, $\leftrightarrow^*$ is called the *equivalence relation* and $\rightarrow^*$ the *reduction relation.* The relation $\leftrightarrow$ is called the *equivalence-step relation* and $\rightarrow$ the *reduction-step relation.*

A certain kind of mixed system predominated in Book's papers in the 1980s (although he did not refer to it as a "mixed system"). This system begins as a Thue system (Σ, E) from which the relations $\leftrightarrow$ and $\leftrightarrow^*$ are defined. Then, from it, a semi-Thue system is defined as (Σ, R) where R is the set of all (y_1, y_2) where $|y_2| < |y_1|$ and either $(y_1, y_2) \in E$ or $(y_2, y_1) \in E$, with $\rightarrow$ and $\rightarrow^*$ defined as above. Verbally, $\rightarrow^*$ is the *reduction relation based on length.* The relationship between $\leftrightarrow$ and $\rightarrow$ in this system may be complicated by the presence of length-preserving rules in E, i.e., rules $(y_1, y_2) \in E$ such that $|y_1| = |y_2|$.

Another kind of mixed system begins with what is called an *abstract reduction system* in the recent literature, which is a semi-Thue system (Σ, R) whose $\rightarrow$ (as defined) is to be thought of as a reduction-step relation. Some semi-Thue systems are more appropriate for being thought of as reduction systems than others. For example, it is generally assumed that the reduction-step relation should be *noetherian*; that is, there should be no infinite sequence of strings $x_1, x_2, \ldots$ such that $x_i \rightarrow x_{i+1}$ holds for all i. One implication of this property is that we can never have both $x \rightarrow y$ and $y \rightarrow x$. Another implication is the existence of at least one *irreducible* string, i.e., an $x \in \Sigma^*$ for which there is no y such that $x \rightarrow y$. Therefore, if $\rightarrow$ is appropriate for being a reduction-step relation then the relation $\rightarrow^*$ is a transitive, reflexive, antisymmetric relation, whose converse is a partial well ordering.

We can get a mixed system from such an existing semi-Thue system (Σ, R) by putting $E = \{(x, y) | (x, y) \in R \text{ or } (y, x) \in R\}$. This type of mixed system is, in effect, what was used in a 1988 monograph by Jantzen [10] and in the very first chapter of the 1993 monograph by Book and Otto. More will be said about it in the next section.

In this review, a rewriting system will always be a mixed system. In some cases we can think of equivalence as coming first and reduction as an instrument in proving equivalence. In other cases we can think of reduction as coming first and equivalence as being defined from it.

3. The Church–Rosser property

This was the most eminent of the properties of Thue systems that Book studied. But the concept as Book used it in the early 1980s is not precisely the same as the concept as it usually appears now. The earlier concept is based on the length of strings, the later one is more abstract. In this review both concepts will be subsumed under one.

A mixed system (Σ, E, R) has the *Church–Rosser property* if, for every $x \in \Sigma^*$, there is a sequence $x_0, x_1, \ldots, x_n$, $n \geqslant 0$, such that (1) $x_0 = x$, (2) $x_i \to x_{i+1}$ for each i, and (3) x_n is the unique irreducible string equivalent to x. The string x_n is thus a *canonical form* of x. The sequence $x_0, x_1, \ldots, x_n$ we call a *reduction sequence* for x. (The term "Church–Rosser" honors work by Alonzo Church and Barkley Rosser [9] on the lambda calculus.)

Note that if we have $x = x_0 \to x_1 \to \cdots \to x_i$, and both $x_i \to x_{i+1}$ and $x_i \to x'_{i+1}$ hold, then we can take either x_{i+1} or x'_{i+1} as the $(i+1)^{st}$ string in a reduction sequence for x. The two strings x_{i+1} and x'_{i+1}, being equivalent to x, have the same canonical form as x, which implies that there are reduction sequences to it both from x_{i+1} and from x'_{i+1}. Consequently, whenever we are constructing a reduction sequence from a given string, we can always take as the next string any string to which the last string reduces. Thus a reduction sequence from any x can be obtained in a straightforward way, with no need for back-tracking.

A mixed system (Σ, E, R) has the *length-based Church–Rosser property*, if it has the Church–Rosser property and reduction is always accompanied by a decrease in length, viz., $(x, y) \in R$ implies $|y| < |x|$, and hence $x \to y$ implies $|y| < |x|$. In such a system, the reduction of a string to its canonical form is expeditious for two reasons: not only is backtracking avoidable in obtaining the reduction sequence, as already noted, but each new string in the sequence is shorter in length than its predecessor. Thus the reduction of a string to its canonical form in a system with the length-based Church–Rosser property can be done in linear time (as will be demonstrated in detail in the next section). Book made much of this fact.

Book was also impressed with the fact that it was computationally simple to tell whether a given Thue system (Σ, Q) for finite Q has the length-based Church–Rosser property. With O'Dunlaing [7] he noted that the decision procedure for this problem, discovered by Nivat and Cochet [12], could be made to run in polynomial time.

When length-preserving rules play an important rôle, it may be appropriate to consider a property that is considerably weaker than the length-based Church–Rosser property. A rewriting system is *preperfect* if it satisfies two conditions:

(a) For every $x \in \Sigma^*$, there is a sequence $x_0 = x, x_1, \ldots, x_n$ such that $x_i \leftrightarrow x_{i+1}$ and $|x_{i+1}| \leqslant |x_i|$ for every i, and x_n has minimal length of all strings equivalent to x (but may not be uniquely so).

(b) For $x, y \in \Sigma^*$, if $|x| = |y|$, $x \leftrightarrow^* y$ and x and y have minimal length in their equivalence class, then there exist $n \geqslant 0$ and a sequence of equal-length strings $x_0 = x, x_1, \ldots, x_n = y$ such that $x_i \leftrightarrow x_{i+1}$ for each i.

This definition varies from the one Book gives, but is equivalent to it. Note that there is no reference in the definition to a reduction relation. Book and his collaborators did not investigate or use the preperfect property as much as they did the length-based Church–Rosser property.

There is a similarity and a difference between preperfectness and the length-based Church–Rosser property. Systems of both kinds offer the computational advantage that any given string can be reduced to an equivalent string of minimal length. However,

the reduction is more expeditious if the system has the length-based Church–Rosser property. In general, the algorithm to reduce a given string to its minimal length in preperfect systems is computationally more complex.

The procedure to reduce a string to an equivalent string of minimal length in a system is useful in the solution of the *string equivalence problem* for that system, i.e.: given strings w_1 and w_2, is $w_1 \leftrightarrow^* w_2$? (This problem is also known as the *word problem* for the monoid presented by the Thue system.) To decide whether w_1 and w_2 are equivalent, one simply reduces the strings to their canonical forms and tests for equality.

The string equivalence problem for Church–Rosser systems and preperfect systems is therefore solvable. However, the algorithm is more complex for preperfect systems than for Church–Rosser systems. As we have noted, even the procedure for obtaining a minimal-length string equivalent to a given string is more complex for preperfect systems. For Book's discussion of alternatives to the length-based Church–Rosser property for systems with viable reduction procedures, the reader is referred to pp. 65–66 of [6].

In cases where a system does not have the length-based Church–Rosser property, it is sometimes possible to revise the system so that it has some other Church–Rosser property. Usually this would require finding another reduction relation not based wholly on length. One idea along these lines is to refine the "shorter than" relation over strings to include alphabetic comparisons. Assuming the alphabet Σ is ordered, we can define $x < y$ for $x, y \in \Sigma^*$ to mean that either $|x| < |y|$ or $|x| = |y|$ and x precedes y in alphabetic order. Then, following [11], we can define a mixed system (Σ, E, R) to be *lexicographically confluent* if (1) $x < y$ for all $(y,x) \in R$ and (2) (Σ, E, R) has the Church–Rosser property. The value of this idea rests on the fact that $<$ is a *complete ordering* of Σ^*: we always have either $x < y$ or $y < x$, for distinct strings x and y.

Thus there are variants to the length-based Church–Rosser property, which is the reason the definition offered at the beginning of this section does not involve length at all. That definition is a generalization of the length-based concept, which Book most often used in the 1980s. However, in their 1993 monograph, Book and Otto have put the abstract concept in the very first chapter, not discussing the length-based concept until Chapter 3 (where, however, it is studied quite thoroughly).

In the mathematical sciences, abstract concepts are often preferred to concrete concepts because they are more general. Let us make some observations along this line about Church–Rosser rewriting systems before closing this section.

If a mixed system $S_1 = (\Sigma, E, R)$ has the Church–Rosser property then it is possible to define

$$E' = \{(x, y) | (x, y) \in R \text{ or } (y,x) \in R\}$$

whereupon the system $S_2 = (\Sigma, E', R)$ is equivalent to S_1; that is to say, $x \leftrightarrow^* y$ holds in S_1 if and only if $x \leftrightarrow^* y$ holds in S_2. (The proof is left to the reader.) This shows that any mixed system with the Church–Rosser property could have started out as an

abstract reduction system, and explains in part the motivation behind the use of abstract reduction systems in recent rewriting theory.

Theorists who prefer to work with abstract reduction systems like to focus on the *confluence* property of such systems. Where (Σ, R) is an abstract reduction system and E is defined from R as E' was defined in the preceding paragraph, then (Σ, R) has the confluence property if and only if (Σ, E, R) has the Church–Rosser property (see, e.g., Lemma 1.1.7 of [8]). For the purposes of this paragraph this result can serve as a definition of "confluence." Because of the closeness in meaning of "confluence" and "Church–Rosser," the former term is not discussed in this review outside this paragraph, even though it is at present the more popular term.

In the next section we shall return to the more concrete Church–Rosser concept of the early 1980s in order to describe one of Book's most important ideas.

4. Linear-time string reduction

Perhaps the most impressive of Book's results about rewriting systems from an applications point of view is that systems with the length-based Church–Rosser property have a highly efficient method of reduction of a string to a canonical form. In [2] he shows how to construct, for any such system, an automaton with two pushdown stores that can reduce any string over the alphabet to its canonical form in time that is linear in the length of the string. This method of reduction will now be described in detail, although the treatment will be discursive rather than technical. We assume that we have a mixed system (Σ, E, R) with the length-based Church–Rosser property.

To execute the first reduction step of a given string, we must find a factor of that string that is the left member of a rule of R; such a factor let us call a "handle." There may be several handles in the string, so we must decide both how we should begin our search for handles and which handle should be the first to be rewritten. We might locate all the handles, and chose to reduce according to which rule yields the greatest reduction in length. But it turns out that it would be quite uneeonomical of time to locate all possible handles before each new step in the reduction. It could in many cases result in a reduction with a small number of reduction steps, but each step would require much time in deciding which is the optimal handle to rewrite.

Let us give up on this idea. Instead, let us reduce as soon as we find the first handle. Arbitrarily, we can search from left to right, and rewrite the first handle we find. Having completed the reduction step we can then do the same thing to the new shorter string, and so on. In this manner we shall at each new step be reducing the string that results from the previous step by rewriting its leftmost handle. Eventually we shall come to a string without a handle, at which point the reduction is complete: the final string is an irreducible equivalent of the original string. And, since the system has the Church–Rosser property, it is the only irreducible equivalent string.

But there is another point of efficiency to be gained. Suppose in a given step of the procedure that we have reduced w_1xw_2 to w_1yw_2, where $(x, y) \in R$, and where w_1

is long. In order to find the leftmost handle in w_1yw_2, we do not have to begin our search at the left end of w_1. We can be sure from what has happened so far that w_1 has no handle. (We omit the proof of this fact, which is by mathematical induction on the number of reduction steps that have taken place.)

More precisely, let h be the length of the longest left side of a rule of R minus 1. If $|w_1| > h$ then, taking $w_1 = w_{12}w_{13}$ where $|w_{13}| = h$, we can confine our search to $w_{13}yw_2$, ignoring w_{12} completely for this step. If $|w_1| \leqslant h$ then, of course, we must begin our search at the left end of w_1.

This completes our description of the algorithm, from which it can be proved that it will always result in the unique irreducible string equivalent to the original, provided that the system has the Church–Rosser property. Everything that has been said so far about the algorithm holds even if the Church–Rosser property is not the length-based property. However, the analysis that follows, showing that it is a linear-time algorithm, requires the length-based property. If the system is not Church–Rosser at all, an equivalent irreducible string will be found, but there is no guarantee that it will be unique or have minimal length.

In order to analyze the algorithm it is convenient to modify the notion of "step." Let us stipulate that the algorithm begins at time 0 with a pointer at the leftmost character of the input string. Thereafter, the string will be modified and the pointer will be moved. At any time t, when t steps have been executed, let $w_1(t)w_2(t)$ be the string, $w_2(t)$ being the suffix that begins with the character that has the pointer. Thus at time 0, $w_1(0)$ is null and $w_2(0)$ is the entire input string.

The strings $w_1(t+1)$ and $w_2(t+1)$ are obtained from $w_1(t)$ and $w_2(t)$ as follows: Between time t and time $t+1$, the rules of R are considered in order, selecting the first one whose left member is a prefix of $w_2(t)$. (For the analysis we need not specify how the rules of R are to be ordered, although some orderings might have small gains in efficiency over others.) If such a rule is found, that handle is rewritten according to that rule and the pointer is moved h places to the left on the string, or, to the beginning of the string if that is not possible. Thus if $w_1(t) = z_1z_2$ and $w_2(t) = x_1x_2$, where $|z_2| = \min(|w_1(t)|, h)$ and the rule is (x_1, y), then $w_1(t+1) = z_1$ and $w_2(t+1) = z_2yx_2$. If this action occurs the step is called a *step of type 1*. If there is no rule whose left member is a prefix of $w_2(t)$ then the pointer is moved one place to the right; if this action occurs the step is called a *step of type 2*.

The algorithm ends when $|w_2(t)|$ is smaller than the length of the shortest left member of a rule. Note that the amount of time for each step is limited by a constant depending only on the system itself. Thus it can be proved that the execution time for the algorithm is bounded by a linear function of the length g of the original string, by proving that the number of steps is so bounded. Accordingly, let k_1 (k_2) be the number of steps of type 1 (type 2) in the execution of the algorithm.

Since $|y| < |x|$ for all $(x, y) \in R$, the length of the string, which never increases, is diminished at least by 1 for each step of type 1. Consequently, $k_1 < g$.

The pointer, during the course of the computation, moves across almost the entire string. During a step of type 1 it moves left at most h characters, h being a constant

for the system. During a step of type 2 it moves right one character. Where $r =$ the total net movement rightward in the course of the algorithm, we have $k_2 - hk_1 \leqslant r < g$, and hence $k_2 < g + hk_1 < (1+h)g$. This gives us an upper bound on the total number of steps:

$$k_1 + k_2 < k_1 + (1+h)g < (2+h)g.$$

And so we are able to conclude that the computation time for the algorithm is bounded by a linear function of g.

This algorithm would be easily implemented as a computer program, which, if care is taken in the writing, runs in linear time. In [2], Book chose to implement the algorithm as a pushdown automaton with two pushdown stores (see also the proof of Theorem 2.2.9 in [8]).

5. Monoid presentation

As mentioned in Section 2, a Thue system (Σ, E) in which E (as well as Σ) is finite can be regarded as a finite presentation of a monoid, where Σ is the set of generators and E is the set of relators. (The relators in a monoid presentation are unlike the relators in a group presentation, in that they cannot always be reduced to the form (w, e), where w is a word over Σ and e is the null word representing the monoid or group identity.) Thus various questions about monoids can be identified with questions about Thue systems. Book sought results about combinatorial monoid theory that could be obtained by a study of rewriting systems.

A good example is the string equivalence problem for Thue systems, discussed in Section 3. It is well known that this problem, whose domain covers all Thue systems, is undecidable. An important subproblem of the string equivalence problem is the *nullifiability* problem: given a Thue system $T = (\Sigma, E)$ and $w \in \Sigma^*$, does $w \leftrightarrow^* e$ hold in T? (The symbol e represents the null string, which represents the monoid identity.) This problem is also undecidable.

There are many problems about Thue systems that are unsolvable when the domain is the class of all Thue systems. One of Book's research objectives has been to find interesting subclasses of the class of all Thue systems for which these problems are decidable. He achieved certain results along these lines in the early 1980s, on which Otto made improvements in 1986 ([13, 14]).

Two such problems are: (1) The *free-monoid problem*: does a given Thue system represent a free monoid (or, if you prefer, a monoid isomorphic to a free monoid)? (2) The *group problem*: does a given Thue system represent a group (or a monoid isomorphic to a group)?

Of course, every free monoid can be represented in a way that makes it apparent that it is a free monoid: if it has n generators, take (Σ, E) where $\Sigma = \{a_1, \ldots, a_n\}$

and E is the empty set. The same holds for groups: if the group has n generators take $\Sigma = \{a_1, a_1', \ldots, a_n, a_n'\}$ and $E = E_1 \cup E_2$ where

$$E_1 = \{(a_1a_1', e), (a_1'a_1, e), \ldots, (a_na_n', e), (a_n'a_n, e)\}$$

and E_2 is the set of group relators expressed appropriately. The free-monoid problem and the group problem are undecidable for the class of all Thue systems because the free-monoid structure and the group structure can be disguised.

Book used the Church–Rosser property and another property, known as the *monadic* property, to define subclasses of the class of Thue systems for which the free-monoid problem and the group problem are solvable. A Thue system (Σ, E) is *monadic* if, for every rule $(u, v) \in E$, $|v| \leqslant 1$ and $|u| > |v|$. The utility of this concept was that it provided access to the theory of context-free grammars and the theory of regular grammars, which have decidability results that can sometimes be applied to monadic Thue systems.

Although Book often had in mind the length-based Church–Rosser property, the results discussed in this section are valid for the more general Church–Rosser property.

Book was able to prove in 1983 that the free-monoid problem was decidable for the class of all monadic Church–Rosser Thue systems with the cancellative property. (A Thue system has the *cancellative property* if, for all $x, y, z \in \Sigma^*$, $xz \leftrightarrow^* yz$ implies $x \leftrightarrow^* y$, and $zx \leftrightarrow^* zy$ implies $x \leftrightarrow^* y$.) This result, although not stated in [4], follows by methods used in that paper (see p. 172 of [8]). Otto's improvement on this result [13] implies that the free-monoid problem is decidable for the class of Church–Rosser Thue systems (Σ, E) where E is finite.

In 1982 Book proved [3] that the group problem is decidable for the class of monadic Thue systems with the Church–Rosser property (cancellativity was not needed). Otto's improvement [14] implies that this result (as in the case of the free-monoid problem) holds when the class of Thue systems is the class of Church–Rosser Thue systems (Σ, E) with finite E.

The last chapter of the monograph by Book and Otto gives a complete and well written technical exposition of the problems discussed in this section. The end of that chapter surveys a number of other algebraic problems about Thue systems: the conjugacy problem, the cancellativity problem, and the problem of the existence of a nontrivial idempotent, which are not discussed here.

6. Another of Book's results about monoids [5]

This last section will consider another problem about the monoids represented by Thue systems. An element of such a monoid can be thought of as an equivalence class of strings. Since the equivalence classes can be multiplied to get other equivalence classes, they are called *congruence classes*. A congruence class can be identified by

any of its members; the notation $[x]$, which for any $x \in \Sigma^*$ represents the set of all strings congruent to x, can be used conveniently to represent the elements of the monoid.

In the monoid of every Thue system, $[e]$ is the monoid identity (e being the null string). A concept of interest to Book was the *group of units* of a monoid, i.e., the largest subgroup of the monoid whose identity is the identity of the monoid. The elements of this subgroup are called the *units* of the monoid. A unit can be identified as the congruence class of any element that has both a left inverse and a right inverse with respect to the monoid identity.

Book was interested in the various properties of monoids that could be discerned from their groups of units, including the question about whether or not a monoid presentation had the length-based Church–Rosser property. He managed to give a complete solution to this problem for monoids defined by a Thue system (Σ, E) in which E has just one rule of the form $(w, e), w \in \Sigma\Sigma^*$. His result broke down into four cases depending on the string w. He used the following concepts from a field of study known as "the combinatorics on words": The *root* of the string w is the shortest string x such that $w = x^k$ for some positive integer k. If w is its own root ($k = 1$) then w is *primitive*. If there are nonnull strings u, v, z such that $w = uz = zv$ then z is an *overlap* of w. Book's result [5] (see also pp. 62–63 of [6]) about a Thue system $T = (\Sigma, \{(w, e)\})$, the monoid M_T presented by T and the group U_T of units of M_T states:

(a) If w is primitive and has no overlap then U_T is trivial (meaning that $[e]$ is the only member of U_T), and T has the length-based Church–Rosser property.

(b) If the root of w is x, $w = x^k$ for some $k \geqslant 2$, and x has no overlap then U_T is a nontrivial finite cyclic group of order k, and T has the length-based Church–Rosser property.

(c) If w is primitive and has overlap then U_T is infinite and T does not have the length-based Church–Rosser property.

(d) If w is not primitive and its root has overlap then T does not have the length-based Church–Rosser property and U_T is infinite with a nontrivial finite cyclic subgroup.

Example for Case (b). Let $T = (\{a, b\}, \{(ababab, e)\})$. Then $w = (ab)^3$ has root ab, which has no overlap. Using well known methods (see, e.g., [7]), it is easy to see that T has the length-based Church–Rosser property with the one reduction rule $(ababab, e)$. It is not difficult to see that $[ab], [abab], [e] \in U_T$ and that no two of these three are equal. It is somewhat more difficult to verify that these three are the only elements of U_T, and that, therefore, U_T is a cyclic group of order 3. (One way of carrying through this verification is to prove that any reduced string that is not e or ab or $abab$ has one of the following forms: (1) $(ab)^i bu$ $(0 \leqslant i \leqslant 2, u \in \Sigma^*)$, in which case it has no right inverse; (2) $ua(ab)^i (0 \leqslant i \leqslant 2, u \in \Sigma^*)$, in which case it has no left inverse; or (3) $(ab)^i aaubb(ab)^j (0 \leqslant i \leqslant 2, 0 \leqslant j \leqslant 2, u \in \Sigma^*)$, in which case it has neither a left inverse nor a right inverse.)

Example for Case (c). Let $T = (\{a,b\}, \{(abbab, e)\})$. Then $w = abbab$ is primitive and has the overlap ab. First note that ab and b commute:

$$abb \leftrightarrow abbabbab \leftrightarrow bab$$

Hence $babab$, $abbab$, $ababb$ and e are all equivalent in T. From this we see that $[b]$ has the two-sided inverse $[abab]$ in M_T. Thus $[b] \in U_T$, and $[b^i] \in U_T$ for all $i \geqslant 0$.

Observe that, for $x_1, x_2 \in \Sigma^*$, if $x_1 \leftrightarrow^* x_2$ then, for some integer k, $|x_1| - |x_2| = 5k$, $|x_1|_a - |x_2|_a = 2k$ and $|x_1|_b - |x_2|_b = 3k$. (The notation $|x_1|_a$ means the number of occurrences of the letter a in the string x_1, etc.) From this it follows that, for $i \neq j$, b^i and b^j are not equivalent, and hence $[b^i] \neq [b^j]$ in U_T; thus U_T is infinite. It also follows that abb and bab, which are equivalent in T, are not equivalent to any shorter string; thus T does not have the length-based Church–Rosser property.

The reader is warned that the the proofs of Cases (b) and (c) of Book's theorem are considerably more involved than the proofs sketched above for the two examples.

Book was interested in the question about what could be done and what could not be done by Church–Rosser rewriting systems. The theorem examined in this section turns out to be helpful in answering this question in certain cases. For Book's detailed discussion of this matter, see pp. 62–64 of [6].

This ends my brief and incomplete review of Ron Book's work on rewriting systems during the 1980s. Since I have not attempted to cover all of his accomplishments in the area, and have not given an account of the results of his disciples, I can claim to have described only a small part of his impact on the theory of rewriting systems.

References

[1] J. Berstel, Congruences plus que parfaites et langages algébriques, Seminaire d'Informatique Théorique, Institute de Programmation, 1976–1977, pp. 123–147.
[2] R. Book, Confluent and other types of Thue systems, J. Assoc. Comput. Mach. 29 (1982) 171–182.
[3] R. Book, When is a monoid a group? The Church–Rosser case is tractable, Theoret. Comput. Sci. 18 (1982) 325–331.
[4] R. Book, Decidable sentences of Church–Rosser congruences, Theoret. Comput. Sci. 24 (1983) 301–312.
[5] R. Book, Homogeneous Thue systems and the Church–Rosser property, Discrete Math. 48 (1984) 137–145.
[6] R. Book, Thue systems as rewriting systems, J. Symbol. Comput. 3 (1987) 39–68.
[7] R. Book, C. Ó'Dunlaing, Testing for the Church–Rosser property, Theoret. Comput. Sci. 16 (1981) 223–229.
[8] R. Book, F. Otto, String-rewriting systems, Springer, Berlin, 1993.
[9] A. Church, J.B. Rosser, Some properties of conversion, Trans. Am. Math. Soc. 39 (1939) 472–482.
[10] M. Jantzen, Confluent string rewriting, EATCS Monograph No. 14, Springer, Berlin, 1988.
[11] D. Kapur, P. Narendran, The Knuth–Bendix completion procedure and Thue systems, SIAM J. Comput. 14 (1985) 1052-1072.
[12] M. Nivat, M. Benois, Congruences parfaites et quasi-parfaites, Seminaire Dubreil 25 (1971–1972) 7–01–09.
[13] F. Otto, Church–Rosser Thue systems that present free monoids, SIAM J. Comput. 15 (1986) 786–792.
[14] F. Otto, On deciding whether a monoid is a free monoid or is a group, Acta Inform. 23 (1986) 99–110.
[15] A. Thue, Probleme über Veränderungen von Zeichenreihen nach gegebenen Regeln, Skr. Vid. Kristiania, I Mat. Natuv. Klasse, No. 10, 1914 34 pp.