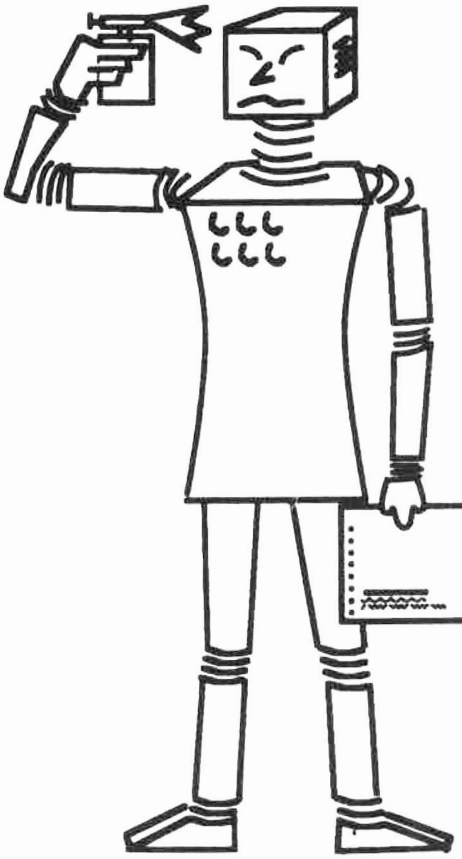
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern

SEKI - REPORT



Conditional semi-Thue Systems for Presenting Monoids

Thomas Deiß

SEKI Report SR-92-12

Conditional semi-Thue Systems for Presenting Monoids[1]

Thomas Deiß
Fachbereich Informatik, Universität Kaiserslautern
Erwin Schrödinger Straße
W-6750 Kaiserslautern
deiss@informatik.uni-kl.de

[1] A short version of this paper was presented at STACS'92, [Dei92]

Abstract

There are well known examples of monoids in literature which do not admit a finite and canonical presentation by a semi-Thue system over a fixed alphabet, not even over an arbitrary alphabet. We introduce conditional Thue and semi-Thue systems similar to conditional term rewriting systems as defined by Kaplan. Using these conditional semi-Thue systems we give finite and canonical presentations of the examples mentioned above. Furthermore we show, that each finitely generated monoid with decidable word problem is embeddable in a monoid which has a finite canonical conditional presentation.

1 Introduction

Thue and semi-Thue systems [Boo85] can be used to examine questions concerning monoids and groups. A Thue system R over an alphabet Σ induces a congruence $\leftrightarrow_R^*$ on Σ^*, the congruence classes modulo $\to_R$ form the monoid M_R. A monoid M is finitely presented by (Σ, R) if M is isomorphic to M_R and both Σ and R are finite, it is finitely generated, if only Σ is finite. If R viewed as a semi-Thue system induces a canonical, i.e. confluent and noetherian, relation, it can be used to decide the word problem of M: Two strings u and v are congruent if and only if they have the same common irreducible descendant.

It is easy to see that a monoid with an undecidable word problem cannot admit a finite and canonical presentation. In addition it has been shown by Narendran and Squier that there exist finitely presented monoids with decidable word problem which do not have a finite and canonical presentation using a fixed alphabet, see e.g. [KN85], resp. using an arbitrary but finite alphabet [Squ87].

To overcome this gap between decidability of the word problem and the existence of finite and canonical presentations we introduce conditional Thue and semi-Thue systems. They are defined similar to conditional term rewriting systems, see e.g. [Kap84, Kap87, JW86, Gan87]. We show, that using conditional semi-Thue systems we get finite and canonical presentations of the examples of Narendran and Squier. Furthermore we are able to strengthen a result of Bauer [Bau81]: Each finitely generated monoid with decidable word problem can be embedded in a monoid, presented by a finite, canonical, and conditional semi-Thue system.

Different conditional string rewriting systems have been used already by Siekmann and Szabo [SS82] to give a finite and canonical presentation of idempotent monoids. They use variables within their rules and a different system to evaluate the premises of a conditional rule and therefore are not a conditional semi-Thue system according to our definition.

In section 2 we summarize results about unconditional Thue and semi-Thue systems. The basic definitions of conditional systems are presented in section 3, followed by results concerning confluence and equivalence in section 4. The examples of Narendran and Squier are examined in section 5. The final section 6 contains the embeddability theorem.

2 Unconditional Thue and semi-Thue systems

We give the basic definitions concerning Thue and semi-Thue systems. Furthermore we state some previous results about decidability of the word problem and the existence of finite canonical presentations according to [SO87].

A *Thue system* R over an alphabet Σ is a set of equations over Σ^*. Throughout this paper we assume Σ to be finite. The *Thue congruence* $\Leftrightarrow_R^*$ is the reflexive transitive closure of the relation $\Leftrightarrow_R$. Thereby, $u \Leftrightarrow_R v$ if there exist $x, y \in \Sigma^*$, such that $u = xly, v = xry$ and $l = r$ or $r = l \in R$. The congruence class $[u]_R$ of a word $u \in \Sigma^*$ is defined as $[u]_R = \{v \in \Sigma^* \mid u \Leftrightarrow_R^* v\}$. The congruence classes form the monoid $M_R = \Sigma^* / \Leftrightarrow_R^*$ under the operation $\circ : [u]_R \circ [v]_R = [uv]_R$. A monoid is presented by (Σ, R) if M is isomorphic to M_R. M is *finitely generated* if Σ is finite, it is *finitely presented* if both Σ and R are finite.

Semi-Thue systems differ from Thue systems in the application of the equations : $u \to_R v$ if there exist $x, y \in \Sigma^*$ such that $u = xly, v = xry$ and $l \to r \in R$. The elements of R are called rules. $\to_R^*$ denotes the reflexive, transitive and $\leftrightarrow_R^*$ the reflexive, transitive, and symmetric closure of $\to_R$.

A word $u \in \Sigma^*$ is *irreducible* modulo R if there is no $v \in \Sigma^*$ such that $u \to_R^+ v$. $IRR(R)$ is the set of all irreducible words modulo R. $\to_R$ is *noetherian* if there is no infinite sequence of words $w_1 \rightarrow_R w_2 \rightarrow_R \ldots$. Two words w_1, w_2 are *joinable* if they have a common descendant: $w_1 \downarrow w_2$ resp. $w_1 \underset{w}{\downarrow} w_2$ if we are interested in the common descendant w.

$\rightarrow_R$ is *confluent* if for all $u, u_1, u_2, u_1 \leftarrow_R^* u \rightarrow_R^* u_2$ implies $u_1 \downarrow u_2$, it is *locally confluent* if $u_1 \leftarrow_R u \rightarrow_R u_2$ implies $u_1 \downarrow u_2$. The *Church-Rosser* property is equivalent to confluence: for all $u_1, u_2 \in \Sigma^*, u_1 \leftrightarrow_R^* u_2$ implies $u_1 \downarrow u_2$.

If $\rightarrow_R$ is noetherian and locally confluent, then it is also confluent [New42]. A semi-Thue system R is *canonical* if $\rightarrow_R$ is confluent and noetherian, then each word $u \in \Sigma^*$ has an unique irreducible normalform $\overline{u}$.

If we view a system R as a Thue system as well as a semi-Thue system, they define the same monoid M_R, i.e. $\Leftrightarrow_R^* = \leftrightarrow_R^*$. Therefore the word problem of a monoid, which has a finite and canonical presentation (Σ, R) is decidable: for $u, v \in \Sigma^*, u \Leftrightarrow_R^* v$ if and only if $\overline{u} = \overline{v}$.

In general it is undecidable whether a given semi-Thue system R is (locally) confluent or noetherian. But is is sufficient to show that for all rules $l \rightarrow r \in R, l > r$ with respect to a well-founded partial order on Σ^* which is compatible with concatenation. Then R is noetherian. The simplest example of such an ordering is ordering by length, others can be found in [Esc86].

Now given a finite, noetherian semi-Thue system it is decidable whether it is locally confluent, and thereby confluent. Let $u \rightarrow v, u' \rightarrow v'$ be two rules in R. If u is a substring of u', i.e. $u' = xuy, x, y \in \Sigma^*$, then u' is an *overlap* and $v' = xvy$ is a *critical pair* of the rules. If u and u' overlap, i.e. $u = xy$ and $u' = yz, x, y, z \in \Sigma^+$, then xyz is an overlap and $vz = xv'$ is a critical pair. We have that R is locally confluent if and only if all critical pairs of R are joinable.

Two string rewriting systems R_1, R_2 are *equivalent* if they present the same monoid using the same alphabet, i.e. $\leftrightarrow_{R_1}^* = \leftrightarrow_{R_2}^*$. If both systems are finite and canonical, equivalence is decidable. They are equivalent if for all $u_1 \rightarrow v_1 \in R$ we have $u_1 \leftrightarrow_{R_2}^* v_1$ and vice versa.

We have seen, that the word problem of a monoid M with a finite and canonical presentation is decidable. Unfortunately, there are finitely presented monoids with decidable word problem which do not admit a finite and canonical presentation. Kapur and Narendran [KN85] showed this for systems over a fixed alphabet, the general result has been proved by Squier [Squ87, Squ88].

Lemma 1 [KN85]. *Let $\Sigma = \{a, b\}, R_1 = \{aba \rightarrow bab\}, R_2 = \{aba \rightarrow ba\}$ and M_{R_1} resp. M_{R_2} be the corresponding monoids. Then both monoids are finitely presented and have decidable word problem, but do not admit finite and canonical equivalent presentations.*

Lemma 2 [Squ87, Squ88]. *There exist finitely presented monoids $S_k, k \geq 1$ with decidable word problem which have no finite canonical presentation.*

In his dissertation [Bau81] Bauer extended semi-Thue systems to n-level rewriting systems, see also [Bau85]. A *n-level rewriting system* R over Σ is a n-tuple $R = (R_1, R_2, \ldots, R_n)$ of semi-Thue systems over Σ. Any admissible sequence of applications of rules starting with $w \in \Sigma^*$ must be the beginning of a sequence $w \rightarrow_{R_1}^* w_1 \rightarrow_{R_2}^* w_2 \rightarrow_{R_3}^* \ldots \rightarrow_{R_n}^* w_n$ where w_i is irreducible modulo R_i. R is canonical if each $w \in \Sigma^*$ has an unique normalform using finite admissible reductions. Then Bauer can show

Lemma 3 [Bau81, Bau85]. *Each finitely presented monoid with decidable word problem has a presentation by a finite canonical 2-level system.*

In his dissertation Bauer used yet another approach to close the gap between decidability of the word problem and the existence of finite canonical presentations. This approach applies also to finitely generated monoids.

Lemma 4 cf. [Bau81]. *Let (Σ, S) be a presentation of a finitely generated monoid M with decidable word problem. Then M can be embedded identically in a monoid M', which is finitely presented by (Σ', S'), such that*

- *$\rightarrow^*_{S'}$ is noetherian.*
- *each word $w \in \Sigma^*$ has an unique irreducible normalform modulo $\rightarrow_{S'}$.*

We will generalize this partial result in section 6 such that we have a canonical finite conditional representation of M'.

3 Conditional systems

We use conditional Thue and semi-Thue systems similar to conditional term rewriting systems as defined e.g. by Kaplan [Kap84]. Therefore the induced congruences are more difficult to handle as for unconditional systems. For example, the congruences are not decidable in general. These problems can be solved by introducing reductive systems analogously to simplifying and reductive conditional term rewriting systems [Kap87, JW86]. At last we state a limitation on the expressiveness of conditional semi-Thue systems when regarding cancellative monoids.

A *conditional Thue system* R is a set of conditional equations. Each equation consists of a *conclusion* $u_0 = v_0$ and a finite set $\{u_i = v_i | 1 \leq i \leq n\}$ of *premises*, all u_i, v_i are strings over an alphabet Σ. We write

$$\overset{n}{\underset{i=1}{\forall}} u_i = v_i :: u_0 = v_0$$

The relation $\Leftrightarrow_R$ is defined as follows: $u \Leftrightarrow_R v$ if and only if there exist $x, y \in \Sigma^*$ and an equation $\forall_{i=1}^n u_i = v_i :: u_0 = v_0$ in R such that $u = xu_0y$ and $v = xv_0y$ or $u = xv_0y$ and $v = xu_0y$ and for all $1 \leq i \leq n$ we have $xu_iy \Leftrightarrow^*_R xv_iy$. $\Leftrightarrow^*_R$ is the *Thue congruence* induced by R. x and y are the left resp. right *context* of the occurrence of u_0 resp. v_0 in u. In this case R is called a *left-right* conditional Thue system. If only the right context y is used in evaluating the premises, i.e. $u_iy \Leftrightarrow^*_R v_iy$, we call R a *right* conditional system.

A *solution* of a conditional equation modulo a Thue system R is a context, such that the premises of the equation within this context are congruent modulo R. x_y is a *minimal solution* modulo a left-right conditional Thue system if there is no suffix x' of x and no prefix y' of y such that x'_y' is a solution. For a right conditional system, a right context y is a minimal solution if it has no prefix y' which is a solution itself. The set of minimal solutions of a conditional equation e modulo a conditional Thue system R is denoted by $sol_R(e)$.

To define conditional *semi-Thue* systems we restrict the application of the equations. Let $\forall_{i=1}^n u_i = v_i :: u_0 \rightarrow v_0$ be a rule of a conditional semi-Thue system R. u_0 is the left-hand side of this rule, v_0 is the right-hand side. Now $u \rightarrow_R v$ if and only if $u = xu_0y$ and $v = xv_0y$, where $x, y \in \Sigma^*$, and xu_iy and xv_iy have a common descendant modulo $\rightarrow_R$ for $1 \leq i \leq n$. As for conditional Thue systems we distinguish left-right and right conditional semi-Thue systems. Solutions modulo a conditional semi-Thue system R are defined in the same manner as for conditional Thue systems. For a minimal solution x_y resp. y we require in addition to the nonexistence of subsolutions, that x and y, resp. (y) are irreducible modulo R.

Notice that the premises must have a common descendant in the context of rule application instead of being congruent as it was in the definition of conditional Thue systems. This causes the first difference to unconditional systems: The Thue congruence and the symmetric, transitive, and reflexive closure of the reduction relation need not coincide anymore. To recover this property we need in addition confluence of R.

Lemma 5 cf. [Kap84, theorem 3.2.]
a) There exists a conditional Thue system R with $\Leftrightarrow_R^* \neq \leftrightarrow_R^*$.
b) If R is confluent, we have $\Leftrightarrow_R^* = \leftrightarrow_R^*$.
proof: The original proofs carry over directly to conditional semi-Thue systems. □

Both $\Leftrightarrow_R^*$ and $\leftrightarrow_R^*$ are compatible with concatenation, hence the congruence classes modulo $\Leftrightarrow_R^*$ resp. $\leftrightarrow_R^*$ form a monoid. As a direct consequence of the lemma above these monoids are the same if R is confluent.

For a finite unconditional system R the relations $\Leftrightarrow_R$ and $\rightarrow_R$ are decidable. This changes, too, when considering conditional systems, cf. [Kap84, theorem 3.3.].

Lemma 6 *There exists a finite conditional Thue system R such that $\Leftrightarrow_R$ and $\rightarrow_R$ are undecidable.*
proof: We use an argumentation similar to that one in the original theorem, but instead of Hilbert's tenth problem we use the encoding of a Turing machine with undecidable halting problem to evaluate the premises. Then the conditional equation can be applied if the premise reduces to an encoding of a final configuration of the Turing machine, but this problem is undecidable. □

Similar to the case of conditional term rewriting systems there is a sufficient criterion such that the reduction relation becomes decidable: We call a conditional semi-Thue system *reductive* if for all rules in R the strings in the premises and the right-hand side are smaller than the left-hand side wrt. to a well-founded ordering which is compatible with concatenation. In analogy to [Kap87, theorem 1.6.] we have

Lemma 7 *Let R be a finite reductive conditional semi-Thue system, then $\rightarrow_R$ is noetherian and decidable.*

The results of the lemmatas 1 and 3 can be combined: If R is finite, confluent, and reductive, then we have $\Leftrightarrow_R^* = \leftrightarrow_R^*$ and $\Leftrightarrow_R^*$ resp. $\leftrightarrow_R^*$ are decidable. Hence R can be used to decide the word problem by means of string rewriting.

We expect that conditional systems have a greater expressiveness than unconditional ones. But cancellative monoids and groups show a limit of conditional systems. A monoid presented by (Σ, R) is cancellative if for all $x, y, u, v \in \Sigma^*, xuy \Leftrightarrow_R^* xvy$ implies $u \Leftrightarrow_R^* v$.

Theorem 1 *A group or cancellative monoid M has a finite canonical left-right conditional presentation iff it has a finite canonical unconditional one.*
proof:

if: If M has a finite canonical unconditional presentation it trivially has a conditional one.

only if: Let (Σ, R) be a finite canonical left-right conditional presentation of M and let $(r)\ \forall_{i=1}^{n} u_i = v_i :: u_0 \rightarrow v_0$ be a conditional rule of R. By definition, r can be applied on a word xu_0y iff x_y is a solution of r modulo R. Since M is cancellative and R is canonical x_y is a solution if and only if λ_λ is a solution, i.e. r can be applied if and only if $u_i \downarrow v_i$ for $1 \leq i \leq n$. Suppose this is true, then we can apply r independent of its context and we may replace r in R by the unconditional rule $u_0 \rightarrow v_0$. If there is at least one premise, such that $u_i \not\downarrow v_i$, then r cannot be applied to any word and we may cancel r in R. These replacements and deletions do not alter the property that R is canonical. Hence we are able to construct a finite canonical unconditional presentation of M. □

$$\begin{tikzcd}
xu_0y \arrow[r, "R_1"] \arrow[dr, dotted, leftrightarrow, "*", "\text{ind.hyp.}\ R_2"'] & xv_0y \arrow[d, "*"', "R_1 \vee R_2"] \\
 & \overline{x}v_0\overline{y} \arrow[u, dotted, bend right, "*"', "\text{ind.hyp.}"']
\end{tikzcd}$$

Figure 1: equivalence

An example of a cancellative monoid is the monoid presented by $\{aba \to bab\}$ [KN85], which does not have a finite and canonical presentation over the alphabet $\Sigma = \{a, b\}$, see lemma 1. An analogous theorem can be proven for right-cancellative monoids and right-conditional systems.

4 Equivalence and Confluence

We examine sufficient criteria to decide equivalence and local confluence of conditional semi-Thue systems. In general, none of these criteria gives us a finite test to decide these questions: we have to consider the (infinite) set of minimal solutions. For deciding local confluence of left-right conditional semi-Thue systems the situation is even worse: we have an infinite number of critical pairs.

4.1 Equivalence

To decide equivalence of two unconditional systems it suffices to show that the rules of each system are congruent modulo the other one. For conditional systems we have to take into account the solutions of the conditional rules. If we compare two finite canonical conditional semi-Thue systems we may restrict the test to minimal solutions. We state this lemma for left-right conditional systems, but it is also valid for right conditional ones.

Lemma 8 *Let R_1, R_2 be two finite canonical conditional semi-Thue systems over an alphabet Σ. If we have for all rules $(r)\forall_{i=1}^{n} u_i = v_i :: u_0 \to v_0$ in R_1, and minimal solutions $x\_y \in sol_{R_1}(r)$ that $xu_0y \leftrightarrow^*_{R_2} xv_0y$ and vice versa for all rules in R_2, then R_1 and R_2 are equivalent.*
proof: We show that $u \leftrightarrow^*_{R_1} v$ implies $u \leftrightarrow^*_{R_2} v$, the case $u \leftrightarrow^*_{R_2} v$ implies $u \leftrightarrow^*_{R_1} v$ is analogous. Since $\to_{R_1}$ is canonical, $\to_{R_1}$ is a well-founded partial ordering on Σ^* which is compatible with concatenation. Let $\twoheadrightarrow_{R_1}$ denote the lexicographic extension of $\to_{R_1}$ to tuples of words. The proof is by noetherian induction using $\twoheadrightarrow_{R_1}$. It suffices to show that $u \to_{R_1} v$ implies $u \leftrightarrow^*_{R_2} v$. Let (r_1) $\forall_{i=1}^{n} u_i = v_i :: u_0 \to v_0$ be the rule used to reduce u to v, i.e. $u = xu_0y, v = xv_0y$ and for all $1 \leq i \leq n, xu_iy \downarrow_{R_1} xv_iy$.

case 1: x, y are irreducible.

Then x has a suffix x', y has a prefix y' such that x'_y' is a minimal solution of r_1. By assumption we have $x'u_0y' \leftrightarrow^*_{R_2} x'v_0v'$ and we conclude $u = xu_0y \leftrightarrow^*_{R_2} xv_0y = v$.

case 2: at least one of x, y is reducible.

Let $\overline{x}, \overline{y}$ be the normalforms of x resp. y modulo R_1. Then $v = xv_0y \to^*_{R_1} \overline{x}v_0\overline{y}$. Since at least one of x, y is reducible we have $(xu_0y, xv_0y) \twoheadrightarrow_{R_1} (xu_0y, \overline{x}v_0\overline{y})$ and $(xu_0y, xv_0y) \twoheadrightarrow_{R_1} (xv_0y, \overline{x}v_0\overline{y})$. By using the induction hypothesis twice we conclude $u = xu_0y \leftrightarrow^*_{R_2} \overline{x}v_0\overline{y} \leftrightarrow^*_{R_2} xv_0y = v$. This is depicted in figure 1. □

$$\begin{tikzcd}
 & & u \arrow[dl] \arrow[dr] & & \\
 & w_1 \arrow[dl, "*"'] \arrow[dr, "*"] & & w_2 \arrow[dl, "*"'] \arrow[dr, "*"] & \\
u_2 \arrow[dr, "*"'] & \text{ind. hyp.} & w \arrow[dl, "*"] & & u_1 \arrow[ddll, "*"] \\
 & u' \arrow[dr, "*"] & & \text{ind. hyp.} & \\
 & & u_0 & &
\end{tikzcd}$$

Figure 2: confluence

4.2 Confluence

Since right conditional semi-Thue systems correspond to conditional term rewriting systems with unary function symbols, the proof of confluence can be carried over from term rewriting to semi-Thue systems, see e.g [Kap87, JW86]. To show confluence of a system R we take the usual approach. We show that R is reductive and thereby noetherian. If all critical pairs are joinable then R is locally confluent, hence it is confluent [Hue80].

Definition 1 *Let (r) $\forall_{i=1}^{n} u_i = v_i :: u_0 \rightarrow v_0$ and (r') $\forall_{i=1}^{n'} u_i' = v_i' :: u_0' \rightarrow v_0'$ be two (not necessarily different) rules of a right conditional Thue system R.*
If u_0' is a substring of u_0, i.e. $u_0 = xu_0'y$,
then $\forall_{i=1}^{n} u_i = v_i \wedge \forall_{i=1}^{n'} u_i'y = v_i'y :: v_0 = xv_0'y$ is a critical pair of r and r'.
If u_0 and u_0' have an overlap, i.e. $u_0 = xy, u_0' = yz, x, y, z \in \Sigma^+$,
then $\forall_{i=1}^{n} u_iz = v_iz \wedge \forall_{i=1}^{n'} u_i' = v_i' :: v_0z = xv_0'$ is a critical pair of r and r'.
$CP(R)$ is the set of all critical pairs of R.
A critical pair (p) $\forall_{i=1}^{n} u_i = v_i :: u_0 = v_0$ is joinable *in R if and only if for all $y \in sol_R(p), u_0y \downarrow_R v_0y$.*

Lemma 9 *Let R be a finite reductive right-conditional semi-Thue system. If and only if all critical pairs of R are joinable in R, then R is locally confluent and thereby confluent.*
proof: Follows the proof of confluence for conditional term rewriting systems in [JW86].

only if: Let (p) $\forall_{i=1}^{n} u_i = v_i :: u_0 = v_0$ be a critical pair of two rules $r, r' \in R$, $y \in sol_R(p)$ and let w be the overlap corresponding to p, then wy reduces to u_0y resp. v_0y. Since R is confluent we have $u_0y \downarrow v_0y$. We conclude that all critical pairs are joinable within their minimal solutions.

if: Let $P(u)$ be valid if and only if $\forall u_1, u_2$ with $u_1 \leftarrow_R^+ u \rightarrow_R^+ u_2$ there exists $u_0 \in \Sigma^*$ such that $u_1 \downarrow_{u_0} u_2$. We show that $P(u)$ is valid for all $u \in \Sigma^*$ by noetherian induction on Σ^*, see e.g. [New42]. Figure 2 turns out the existence of u_0, provided that $w_1 \leftarrow_R u \rightarrow_R w_2$ implies $\exists w \in \Sigma^*$ with $w_1 \downarrow_w w_2$.

Let us assume that we used the rules (r) $\forall_{i=1}^{n} u_i = v_i :: u_0 \rightarrow v_0$ and (r') $\forall_{i=1}^{n'} u_i' = v_i' :: u_0' \rightarrow v_0'$ to reduce u to w_1 resp. w_2. There are 3 cases according to the positions of u_0 and u_0' in u.

case 1: u_0 and u_0' do not overlap, i.e. $u = x_1u_0x_2u_0'x_3$ and $w_1 = x_1v_0x_2u_0'x_3 \leftarrow_r u \rightarrow_{r'} x_1u_0x_2v_0'x_3 = w_2$. This corresponds to a variable overlap in conditional term rewriting systems.

It is easy to see that w_1 reduces to $x_1v_0x_2v_0'x_3$. Also for all $1 \leq i \leq n, u_ix_2u_0'x_3 \rightarrow_R^* u_ix_2v_0'x_3$ and $v_ix_2u_0'x_3 \rightarrow_R^* v_ix_2v_0'x_3$. Similar to figure 2 we may apply the induction hypothesis twice to get common descendants of $u_ix_2v_0'x_3$ and $v_ix_2v_0'x_3$. Hence $x_2v_0'x_3$ is a solution of r and we may reduce $w_2 = x_1u_0x_2v_0'x_3$ to $w = x_1v_0x_2v_0'x_3$.

case 2: u_0' is a substring of u_0, i.e. $u = x_1u_0x_4 = x_1x_2u_0'x_3x_4$ and $w_1 = x_1v_0x_4 \leftarrow_r u \rightarrow_{r'} x_1x_2v_0'x_3x_4 = w_2$.

Let us assume first that x_4 is irreducible. Then x_4 has a prefix x_4' which is a minimal solution of the critical pair corresponding to the overlap $u_0 = x_2u_0'x_3$ of r and r'. This is joinable by hypothesis and we conclude $x_1v_0x_4 \downarrow x_1x_2v_0'x_3x_4$.

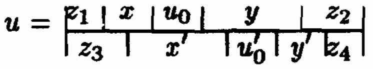
If x_4 is reducible, let $\overline{x_4}$ be its normalform. There are $u' = x_1u_0\overline{x_4} = x_1x_2u_0'x_3\overline{x_4}, w_1' = x_1v_0\overline{x_4}$ and $w_2' = x_1x_2v_0'x_3\overline{x_4}$. Again as in figure 2 we can show that $u_i\overline{x_4} \downarrow v_i\overline{x_4}$ for $1 \leq i \leq n$ and $u_i'x_3\overline{x_4} \downarrow v_i'x_3\overline{x_4}$ for $1 \leq i \leq n'$. Hence $\overline{x_4}, x_3\overline{x_4}$ are solutions of r resp. r', and we get $w_1' \leftarrow u' \rightarrow w_2'$. This is the case above, w_1', w_2' and thereby w_1 and w_2 have a common descendant w.

case 3: u_0 and u_0' have an overlap. Similar to case 2 we show the existence of the common descendant w.

Hence $w_1 \leftarrow w \rightarrow w_2$ implies $w_1 \downarrow w_2$ which finishes the proof. □

Though we have a finite number of critical pairs of a finite right conditional semi-Thue system, this characterization does not give a finite test for confluence. There may be infinitely many minimal solutions of a critical pair.

For left-right conditional systems the situation is even worse. By extending definition 1 to left-right conditional systems, joinability of critical pairs does not suffice to show local confluence. Let us have a word $u \in \Sigma^*$ reducible by two rules $(r)\ \forall_{i=1}^{n} u_i = v_i :: u_0 \rightarrow v_0$, $(r')\ \forall_{i=1}^{n'} u_i' = v_i' :: u_0' \rightarrow v_0'$. x_y and x'_y' are the solutions used by the rules. Let us assume that the left hand sides do not overlap with themselves but with the solution of the other rule, e.g.



Since u_0 and u_0' do not overlap, there is no critical pair corresponding to this situation. But it is possible that the descendants $z_1xv_0yz_2$ and $z_3x'v_0'y'z_4$ are not joinable, since the rules disturb mutually their contexts.

To show local confluence we expand R to $R_e = \{xu_0y \rightarrow xv_0y | r$ is a rule in R, x_y is a solution of r and no suffix x' of x, no prefix y' of y is a solution of $r\}$. Notice that R_e is unconditional and in general infinite. Now, local confluence of R can be easily deduced from local confluence of R_e.

Lemma 10 *Let R be a reductive left-right conditional semi-Thue system, then R is confluent if and only if R_e is confluent.*

Thus we have a formal characterization of (local) confluence of left-right conditional systems. But usually we have to examine an infinite number of critical pairs of R_e.

5 Examples

Now we are able to study the examples mentioned in the introduction. We first examine the example of Narendran. After that we briefly investigate an example due to Kirchner and Hermann [KH89] and finally we present the example of Squier.

5.1 A representation with no equivalent finite canonical presentation

We consider the one rule system $R = \{aba \to ba\}$ over the alphabet $\Sigma = \{a, b\}$. As already stated in lemma 1 there is no finite canonical unconditional semi-Thue system equivalent to R. Using a completion procedure we get the infinite system $R_{KB} = \{ab^n a \to b^n a | n \geq 1\}$, which is canonical and equivalent to R. As announced there is a conditional semi-Thue system which is finite, canonical, and equivalent to R.

Theorem 2 *The right conditional system $R_c = \{aba \to ba; ab = b :: abb \to bb\}$ is a finite canonical system equivalent to $R = \{aba \to ba\}$.*

We will use r_u and r_c as abbreviations of the unconditional resp. conditional rule of R_c. At first we determine the set of minimal solutions of the premise of r_c.

Lemma 11 $sol_{R_c}(r_c) = \{b^n a | n \geq 0\}$.
proof: Let $S = \{b^n a | n \geq 0\}$

$S \subseteq sol_{R_c}(r_c)$: Induction on n shows that $abb^n a \downarrow bb^n a$.

$sol_{R_c}(r_c) \subseteq S$: Let $x \in sol(r_c)$, by definition x is irreducible. We show by induction on length l of x, that $x \in S$, i.e. $x = b^n a$.

$l = 0$: that means $x = \lambda$, but no left or right-hand side of r_u or r_c is a substring of ab resp. b. Therefore they are the only elements in their equivalence classes and they are not congruent. This contradicts $x \in sol_{R_c}(r_c)$.

$l = 1$: $x = a$, by definition of S we have $x \in S$ and $aba \downarrow ba$.
$x = b$, this again contradicts $x \in sol_{R_c}(r_c)$, since we would have to show $abb \downarrow bb$. Only r_c could be applied on abb, but the premise of r_c is not joinable within the context λ.

$l > 1$: $x = ax'$, this is a contradiction to $x \in sol_{R_c}(r_c)$ since a is a prefix of x and $a \in sol_{R_c}(r_c)$.
$x = bx'$, by assuming $x \in sol_{R_c}(r_c)$ we have $abx \downarrow bx$, i.e. $abbx' \downarrow bbx'$. x' is irreducible and therefore the only possibility to reduce $abbx'$ is $abbx' \to_{r_c} bbx'$. This rule may be applied if $x' \in sol_{R_c}(r_c)$. By induction hypothesis we have $x' = b^{l-2}a$, hence $x = b^{l-1}a$. □

Lemma 12 *The system R_c is canonical.*
proof: Since both rules of R_c are length reducing and the premises are shorter than the left-hand sides, R_c is reductive. To show confluence we have to consider two overlaps. The critical pair $baba = abba$ corresponding to the overlap $ababa$ is joinable in bba. The other overlap $ababb$ results in the critical pair $ab = a :: babb = abbb$. It has the same premise as r_c, hence it has the same set $S = \{b^n a | n \geq 0\}$ of minimal solutions and we have to show $babbb^n a \downarrow abbbb^n a$ for all $n \geq 0$. These words can be reduced to $bbbb^n a$ using r_c. Therefore all critical pairs are joinable and we conclude that R_c is canonical. □

Finishing the proof of theorem 2 we show that R and R_c are equivalent.

Lemma 13 *R and R_c are equivalent.*
proof: Since R is a subset of R_c it suffices to show for all $x \in sol_{R_c}(r_c)$ that $abbx \leftrightarrow_R^* bbx$. R and R_{KB} are equivalent, hence we can use R_{KB} to show this. By Lemma 11 we have $x = b^n a, n \geq 0$. But now $abbb^n a \rightarrow_{R_{KB}} bbb^n a$, which completes the proof. □

In [KH89] Kirchner and Hermann presented a term rewriting system which can be converted into the semi-Thue system $R' = \{fgf \rightarrow fh\}$. We use the reversed system $R = \{fgf \rightarrow hf\}$ as an example. Completing R gives the infinite canonical system $R_{KB} = \{fgh^n f \rightarrow h^{n+1} f \mid n \geq 0\}$. Again the monoid M_R can be presented by a finite canonical conditional semi-Thue system, the proof is left to the reader.

Lemma 14 *Let $R = \{fgf \rightarrow hf\}$, then $R_c = \{fgf \rightarrow hf; fg = h :: fgh \rightarrow hh\}$ is a finite canonical right conditional semi-Thue system equivalent to R.*

5.2 A monoid without a finite canonical presentation

Squier [Squ87, Squ88] defined a family of monoids $S_k, k \geq 1$, which cannot be presented by finite unconditional semi-Thue systems. We show that each of these monoids can be presented by a finite canonical right conditional system. We will study the monoid S_1 in detail, the results can be easily generalized to $S_k, k > 1$.

Let $\Sigma_1 = \{a, b, t, x, y\}, R_1' = \{xa \rightarrow atx; xt \rightarrow tx; xb \rightarrow bx; xy \rightarrow \lambda\}$ and $R_1 = R_1' \cup \{at^n b \rightarrow \lambda \mid n \geq 0\}$. S_1 is the monoid presented by (Σ_1, R_1), it is finitely presented by the system $R_1'' = R_1' \cup \{ab \rightarrow \lambda\}$, [Squ87]. Using a syllable or collected ordering as defined in [Sim87], based on the precedence $a > t > b > x > y$ we can show that R_1 is noetherian. Furthermore R_1 is confluent, hence R_1 is canonical. Therefore we are able to compute unique normal forms modulo R_1, hence the word problem of S_1 is decidable. Though we have an infinite semi-Thue system to solve the word problem of S_1, there is no finite system with this property.

Lemma 15 [Squ88]. *S_1 has no finite canonical unconditional presentation.*

To show that S_1 can be presented by a finite canonical conditional system, we need a canonical unconditional system $\overline{R}_1$ equivalent to R_1. It is similar to R_1, except that the rules $\{at^n b \rightarrow \lambda \mid n \geq 0\}$ are replaced by a set of rules $\{at^{n+1} b \rightarrow at^n b \mid n \geq 0\} \cup \{ab \rightarrow \lambda\}$. The rules of the conditional system will resemble the rules in $\overline{R}_1$. Especially the rules $at^{n+1}b \rightarrow at^n b$ will be replaced by a conditional rule.

Theorem 3 *The right conditional system $R_c = R_1' \cup \{ab \rightarrow \lambda; atb \rightarrow \lambda; at = a :: att \rightarrow at\}$ is a canonical presentation of the monoid S_1.*

The last 3 rules will be abbreviated with r_1, r_2 and r_c. Similar to lemma 11 we get $sol_{R_c}(r_c) = \{t^n b \mid n \geq 0\}$. Then we can show that R_c is canonical.

Lemma 16 $sol_{R_c}(r_c) = \{t^n b \mid n \geq 0\}$.
proof: Let $S = \{t^n b \mid n \geq 0\}$.

$S \subseteq sol_{R_c}(r_c)$: Let $z \in S$, i.e. $z = t^n b$. Induction on n shows that z is a solution of r_c. Since z is irreducible and the proper prefixes of z are no solutions, z is a minimal solution.

$sol_{R_c}(r_c) \subseteq S$: Let $z \in sol_{R_c}(r_c)$, but $z \notin S$, i.e. $z \neq t^n b$, z is of minimal length with this property. By assuming $z \in sol_{R_c}(r_c)$ we have that z is irreducible and no proper prefix of z is a solution. We distinguish the following cases by looking at the first letter of z.

$z = az'$: but $ataz'$ and aaz' are irreducible, hence z cannot be a solution.

$z = bz'$: Either z' is λ, then z is of the form $t^n b$ ($n = 0$) or $z' \neq \lambda$, then a proper prefix of z is a solution which contradicts $z \in sol_{R_c}(r_c)$.

$z = tz'$: $attz'$ is reducible by r_c if $atz' \downarrow az'$. z itself is irreducible and no pioper prefix of z' is a solution of r_c (Assuming the contrary gives a contradiction to the property, that no proper prefix of z is a solution). Hence $z' \in sol_{R_c}(r_c)$. Since $|z'| < |z|$ and z is of minimal length with $z \neq t^n b$ we get $z' = t^m b$, contradicting $z \neq t^n b$.

$z = xz'$: $atxz'$ and axz' are irreducible, hence z cannot be a solution of r_c.

$z = yz'$: analogous. □

Lemma 17 *R_c is canonical.*

proof: Since the rules of R_c can be ordered with the ordering defined above, R_c is reductive and therefore noetherian. To prove that R_c is confluent we have to consider the overlaps $xab, xatb$ and $xatt$. It is easy to see that the critical pairs $atxb = x$ and $atxtb = x$ corresponding to the first two overlaps are joinable. The overlap $xatt$ gives the critical pair (p) $at = a :: atxtt = xat$. Since p and r_c have the same premises, $sol_{R_c}(r_c) = sol(p) = \{t^n b | n \geq 0\}$. For all $n \geq 0, atxttt^n b$ and $xatt^n b$ are joinable in $attt^n bx$, hence R_c is confluent. □

We have equivalence of R_c and $\overline{R}_1$, the proof is similar to that of lemma 13. Since $\overline{R}_1$ and R_1 are equivalent, R_c and R_1 are equivalent too. This finishes the proof of theorem 3.

6 Embeddability of monoids

In section 5 we used right conditional systems only and concentrated on single examples. To achieve a general result we resume to the use of left-right conditional semi-Thue systems.

We are able to strengthen the result of Bauer [Bau81] stated in lemma 3. There is a finite canonical conditional presentation of the embedding monoid.

Theorem 4 *Let M be a monoid with decidable word problem which is finitely generated by (Σ, S). Then M can be embedded in a monoid M' which is finitely presented by a canonical left-right conditional semi-Thue system C over an alphabet Δ.*

Throughout the rest of this section we give the proof of the theorem. The basic idea follows the proof of Bauer. We choose an unique representative $\hat{w}$ of each equivalence class $[w]_S$, $w \in \Sigma^*$ and we define a function $\varphi : \Sigma \to \Delta^*$ as $\varphi(a_i) = (a_i), a_i \in \Sigma$. φ can be extended to a function $\varphi^* : \Sigma^* \to \Delta^*$ by $\varphi^*(\lambda) = \lambda$ and $\varphi^*(a_i w) = \varphi(a_i)\varphi^*(w)$. This induces an homomorphism $\hat{\varphi} : M \to M', \hat{\varphi}([w]_S) = [\varphi^*(w)]_C$. For $w = a_{i_1} \ldots a_{i_n}$ the system C is intended to do the reduction

$$\varphi^*(w) = (a_{i_1}) \ldots (a_{i_n}) \to_C^* \begin{cases} (\hat{w}) & \text{if } \hat{w} \neq \lambda \\ \lambda & \text{if } \hat{w} = \lambda \end{cases}$$

C has to perform two tasks. First, given a word w within delimiting parentheses, called a configuration, compute the representative $\hat{w}$ of w: $(\hat{w})$. Second, given two configurations $(w_1)(w_2)$ which are concatenated as words, concatenate them into one configuration $(w_1 w_2)$. The main problem is to prevent the rules to apply on words other than (w) resp. $(w_1)(w_2)$. Notice that we must not reduce (w) if $w = \hat{w}$, otherwise we would have an infinite reduction. Then we can show that C is canonical and we get $\varphi^*(u) \downarrow_C \varphi^*(v)$ if and only if $u \leftrightarrow_S^* v$. Hence M is embedded in M'.

6.1 Construction of the system C

The choice of the representatives of the equivalence classes modulo S is the most obvious one. We define a length-lexicographical ordering on Σ^* and choose the smallest element of an equivalence class $[w]_S$ as its representative $\hat{w}$. This ordering is well-founded and total, hence the representative is unique. We also speak of $\hat{w}$ as a representative of the words in $[w]_S$. To compute the representative of a word we define a function $f : \Sigma^* \to \Sigma^*$ by

$$f(w) = \mu_{z \leq w} (z \leftrightarrow_S^* w)$$

Since the word problem of M is decidable, the function f is computable. Hence there is a deterministic Turing machine M_f which computes f. According to Davis [Dav56] this machine can be constructed such that it halts when started from an arbitrary configuration. We simulate M_f by an unconditional semi-Thue system using the method described in [DW83]. This system is noetherian as M_f halts when started from an arbitrary configuration and it is locally confluent since we use two different copies of the input alphabet Σ to the left and to the right of the letters denoting the state symbols of the Turing machine. Hence there are no overlaps and no critical pairs.

Before starting the simulation some preprocessing is done using a system R_f^1. Its main purpose is to prepare a configuration (w), such that M_f can be simulated. It uses two copies Σ_c and Σ_c' of the alphabet Σ, [,] as copies of (,) and state symbols q^1, q^2. First it scans to the right and copies all letters $a_i \in \Sigma$ to $c_i \in \Sigma_c$ until it reaches a letter not in Σ. If this is] it turns to the left and copies all letters in Σ_c to Σ_c' until it reaches a letter not in Σ_c. If this is [the state changes to q_0, the starting state of the simulation of M_f. This simulation is encoded in the system R_f^2. It uses letters $q_0, q_1, \ldots, q_f$, where q_f is the final state. Changing to state q_f is the last action the Turing machine M_f performs. In addition to Σ_c and Σ_c', R_f^2 uses two blank symbols b_c, b_c'. We have

$$\begin{array}{lll}
R_f^1 : & q^1 a_i \to c_i q^1 & q^1) \to q^2] \\
 & c_i q^2 \to q^2 c_i' & [q^2 \to [q_0
\end{array}$$

where $a_i \in \Sigma, c_i \in \Sigma_c, c_i' \in \Sigma_c'$.

$$\begin{array}{lll}
R_f^2 : & q_i c_j' \to q_k c_l' & \\
 & q_i c_j' c_k' \to c_j q_l c_k' & q_i c_j'] \to c_j q_l b_c'] \\
 & c_j q_i c_k' \to q_l c_j' c_k' & [q_i c_k' \to [q_l b_c' c_k'
\end{array}$$

where $q_i, q_l \in \{q_o, q_1, \ldots, q_f\}, c_j \in \Sigma_c \cup \{b_c\}$, and $c_j', c_k', c_l' \in \Sigma_c' \cup \{b_c'\}$.

Started from a correct configuration, i.e. $[q_0 w_c'), w_c' \in \Sigma_c'^*$, R_f^2 computes the configuration $[b_c^* q_f w_c' b_c'^*]$. This computation is continued by a system R_f^3 to remove the blanks and to copy the letters back to the input alphabet, we get $(\hat{w})$.

$$\begin{array}{llll}
R_f^3 : & q_f \to q^3 & & \\
 & q^3 c_i' \to c_i q^3 & q^3 b_c' \to q^4 & q^3] \to q^5) \\
 & q^4 b_c' \to q^4 & q^4] \to q^5) & \\
 & c_i q^5 \to q^5 a_i & b_c q^5 \to q^6 & [q^5 \to (\\
 & b_c q^6 \to q^6 & [q^6 \to (&
\end{array}$$

where $a_i \in \Sigma, c_i \in \Sigma_c, c_i' \in \Sigma_c'$.

Let R_f be $R_f^1 \cup R_f^2 \cup R_f^3$ and Q be the set of states used in R_f. Since R_f^1, R_f^2 and R_f^3 are noetherian and the left-hand sides of each system do not overlap with the right-hand sides of the other systems, R_f itself is noetherian, see e.g. [Der81]. Furthermore, R_f has no critical pairs, hence R_f is locally confluent and thereby confluent and canonical.

Let Δ be an alphabet which contains all the letters used in the rules of R_f. R_f computes f in the following sense:

Lemma 18 *Let $z_1[q^1a_iz_2 \in \Delta^*$.
Then $z = z_1[q^1a_iz_2 \rightarrow^*_{R_f} z_1(z_3$ if and only if $z_2 = w)z_4$ with $w \in \Sigma^*$,
giving us $z_1[q^1a_iw)z_4 \rightarrow^*_{R_f} z_1(\widehat{a_iw})z_4$.*

proof:

$$\begin{array}{lll} \text{if:} \quad z_1[q^1a_iw)z_4 & \rightarrow^*_{R^1_f} & z_1[q_0c'_iw'_c]z_4 \\ & \rightarrow^*_{R^2_f} & z_1[b^*_cq_f\widehat{c'_iw'_c}b'^*_c]z_4 \\ & \rightarrow^*_{R^3_f} & z_1(\widehat{a_iw})z_4 \end{array}$$

only if: All rules in R_f contain exactly one letter in $Q \cup \{(\}$ in their left-hand side as well as in their right-hand side. Hence a rule can be applied only where such a letter is located and it does not change the number of these letters. To the left of these symbols all rules contain only letters in $\Sigma_c \cup \{b_c, [\}$, to the right letters in $\Sigma \cup \Sigma'_c \cup \{b'_c,],)\}$. Since these sets have an empty intersection, each reduction of z uses a subword which does not overlap with the part of z used by another reduction. Therefore we may assume without loss of generality that q^1a_i is the only position in z where a rule can be applied. Now assume that $z_2 \notin \Sigma^*)\Delta^*$: $z_2 \in \Sigma^*$ or $z_2 \in \Sigma^*\delta\Delta^*$ where $\delta \notin \Sigma \cup \{)\}$, i.e. $z_2 = w\delta z'$. In the first case $z_1[q^1a_iz_2$ reduces to $z_1[c_iw_cq^1$ which is irreducible. In the second case, $z_1[q^1a_iw\delta z' \rightarrow^*_{R_f} z_1[c_iw_cq^1\delta z'$ which is irreducible too. Each word in these reductions starts with $z_1[$, hence we cannot reduce $z_1[q^1a_iz_2$ to $z_1(z_3$, a contradiction. □

R_f will be started by a conditional rule in C. The premise of this rule tests whether there is actually a word w with $w \in \Sigma^*$ and $w \neq \hat{w}$. To test the second property we use a function $p : \Sigma^* \rightarrow \Delta^*$ defined by

$$p(w) = \begin{cases} w & \text{if } w \neq \hat{w} \\ \$w & \text{if } w = \hat{w} \end{cases}$$

Thereby \$ is a new letter. Since we can compute $\hat{w}$ using the function f, p is computable too. Similar to the construction above there is a Turing machine M_p and an unconditional semi-Thue system $R_p = R^1_p \cup R^2_p \cup R^3_p$. It uses two new copies $\Sigma_d \cup \{\$_d\}, \Sigma'_d \cup \{\$'_d\}$ of $\Sigma \cup \{\$\}$ and copies $\{,\}$ of $(,)$. R^1_p works completely analogous to R^1_f, R^2_p is the simulation of M_p using state symbols $p_0, p_1, \ldots, p_f$ and blank symbols b_d and b'_d. R^3_p finishes the simulation similar to R^3_f. It gives normalforms $|w)$ resp. $|\$w)$ according to the test $w = \hat{w}$.

$$\begin{array}{lll} R^1_p: & p^1a_i \rightarrow d_ip^1 & p^1) \rightarrow p^2\} \\ & d_ip^2 \rightarrow p^2d'_i & \{p^2 \rightarrow \{p_0 \end{array}$$

where $a_i \in \Sigma, d_i \in \Sigma_d, d'_i \in \Sigma'_d$.

$$\begin{array}{lll} R^2_p: & p_id'_j \rightarrow p_kd'_l & \\ & p_id'_jd'_k \rightarrow d_jp_ld'_k & p_id'_j\} \rightarrow d_jp_lb'_d\} \\ & d_jp_id'_k \rightarrow p_ld'_jd'_k & \{p_id'_k \rightarrow \{p_lb'_dd'_k \end{array}$$

where $p_i, p_l \in \{p_o, p_1, \ldots, p_f\}, d_j \in \Sigma_d \cup \{b_d, \$_d\}$, and $d'_j, d'_k, d'_l \in \Sigma'_d \cup \{b'_d, \$'_d\}$.

$$\begin{array}{llll} R^3_p: & p_f \rightarrow p^3 & & \\ & p^3d'_i \rightarrow d_ip^3 & p^3b'_d \rightarrow p^4 & p^3\} \rightarrow p^5) \\ & p^4b'_d \rightarrow p^4 & p^4\} \rightarrow p^5) & \\ & d_ip^5 \rightarrow p^5a_i & b_dp^5 \rightarrow p^6 & \{p^5 \rightarrow | \\ & b_dp^6 \rightarrow p^6 & \{p^6 \rightarrow | & \end{array}$$

where $a_i \in \Sigma \cup \{\$\}, d_i \in \Sigma_d \cup \{\$_d\}, d'_i \in \Sigma'_d \cup \{\$'_d\}$.

We now fix the alphabet Δ to be the set of all letters used in R_f and R_p. Notice that lemma 18 remains valid using this extended alphabet. Analogously we can show

Lemma 19 *Let* $z_1\{p^1 a_i z_2 \in \Delta^*$.
Then $z_1\{p^1 a_i z_2 \to^*_{R_p} z_1|z_3$ *if and only if* $z_2 = w)z_4$ *with* $w \in \Sigma^*$, *giving us*

$$z_1\{p^1 a_i w)z_4 \to^*_{R_p} \begin{cases} z_1|a_i w)z_4 & \textit{if } a_i w \neq \widehat{a_i w} \\ z_1|\$a_i w)z_4 & \textit{if } a_i w = \widehat{a_i w} \end{cases}$$

As announced, R_f is started by a conditional rule. To ensure that the configuration has at least one letter, the conclusion of the rule has the form $(a_i \to [q^1 a_i$. There is one such rule for each letter $a_i \in \Sigma$. We use $\{p^1 a_i = |a_i$ as premise, thereby achieving that R_p must be used to evaluate it. That is, we have the conditional rules:

$$\{p^1 a_i \;=\; |a_i \;::\; (a_i \;\to\; [q^1 a_i \quad (1)$$

for all $a_i \in \Sigma$. We define the conditional semi-Thue system R as $R = (1) \cup R_f \cup R_p$. For $w \in \Sigma^*$ we are able to reduce $(w) \to^*_R (\hat{w})$ without getting stuck in infinite loops.

Lemma 20 *Let* x_y *be a solution of* $\{p^1 a_i = |a_i :: (a_i \to [q^1 a_i$,
then $x \in \Delta^*, y = w)y_1 \in \Sigma^*)\Delta^*$ *and* $a_i w \neq \widehat{a_i w}$.
proof: By definition we have $x\{p^1 a_i y \downarrow_R x|a_i y$. The rules in R do not overlap and do not change the number of the state symbols and 'opening brackets' $(, \{, [$ and $|$. Hence the reductions in $x[p^1 a_i y$ do not interfere with each other, they use disjoint parts of $x[p^1 a_i y$. We may assume without loss of generality, that there is exactly one rule which can be used to reduce this word.

Since there is no rule with $|$ in its left-hand side, $x|a_i y$ is irreducible and we have $x\{p^1 a_i y \to^*_R x|a_i y$. The state symbol p^1 can only be changed to $p^1, \ldots, p^6, p_0, \ldots, p_f$ or $|$, which again are used in R_p only. Therefore $x\{p^1 a_i y \to^*_{R_p} x|a_i y$, lemma 19 gives $y = w)y_1$, $w \in \Sigma^*$ and $a_i w \neq \widehat{a_i w}$, finishing the proof. □

As for R_f we have that $R_f \cup R_p$ is noetherian. The left-hand sides of (1) do not overlap with the right-hand sides of the rules in $R_f \cup R_p$, hence R is noetherian too, see again [Der81]. The rules to concatenate two configurations resemble the rules used in the examples before, see section 5. Together with a rule to delete empty configurations they form the system T:

$$\begin{array}{rclcrcll} & & & & () & \to & \lambda & (2) \\ & & & & (a_i)(a_j) & \to & (a_i a_j) & (3) \\ (a_i)(a_k & = & (a_i a_k & :: & (a_i)(a_j a_k & \to & (a_i a_j a_k & (4) \\ a_i)(a_k) & = & a_i a_k) & :: & a_i a_j)(a_k) & \to & a_i a_j a_k) & (5) \\ a_i)(a_l & = & a_i a_l & :: & a_i a_j)(a_k a_l & \to & a_i a_j a_k a_l & (6) \end{array}$$

where $a_i, \ldots, a_l \in \Sigma$. Using $(a_i)(a_j) \to (a_i a_j)$ we get $(a_i a_k)(a_j) \to_T (a_i a_k a_j), (a_i)(a_k a_j) \to_T (a_i a_k a_j)$ and $(a_i a_k)(a_l a_j) \to_T (a_i a_k a_l a_j)$. This can be used to get $(a_i)(a_l a_k a_j) \to_T (a_i a_l a_k a_j)$ etc. and finally we get $(w_1)(w_2) \to_T (w_1 w_2)$, for all $w_1, w_2 \in \Sigma^*$.

T does not introduce new letters, hence we do not need to extend the alphabet Δ. We use $C = R \cup T$ to present the monoid M' of theorem 4 by (Δ, C). In the next section we show that the properties of R and T do not change essentially when using C.

6.2 Properties of C

A central part in concluding properties of C is to determine the sets of solutions of the conditional rules. Especially for the rules (4) – (6) this is very difficult. Therefore we proceed indirectly. We examine a restricted relation $\to_C$, then we compare $\to_C$ and $\to_C$, concluding properties of $\to_C$.

Up to now the premises of a rule had to be joinable within its context if the rule should be applied. To restrict the application we demand that one part of the premise can be reduced to the other, each within the current context. That is, let $(r)\ u = v :: l \rightarrow r$ be a conditional rule in C, then

$$xly \rightharpoonup_r xry \text{ if and only if } xuy \rightharpoonup^*_C xvy$$

Notice that the premises are no longer symmetrical, the left part should reduce to the right one. Furthermore, this is closer to the intended use of the rules. In fact we have for rule (1) $x\{p^1a_iy \rightharpoonup^*_C x|a_iy$, see the proof of lemma 20. For the rules (4) – (6) we can show that we actually need one reduction step only to evaluate the premises. Therefore we have $xly \rightharpoonup xry$ if and only if $xuy \rightharpoonup xvy$. Unless stated otherwise, we use the relation $\rightharpoonup$ in the sequel.

Though we changed the system as well as the reduction relation considerably, the solutions of the rules (1) do not change:

Lemma 21 $sol_C(\ \{p^1a_i = |a_i :: (a_i \rightarrow [q^1a_i\ ) = \{\ \lambda\_w) \,|\, w \in \Sigma^*$ *and* $a_iw \neq \widehat{a_iw}\}$.
proof: The rules in T each decrease the number of opening brackets, but both parts of the premise of (1) contain the same number, counting $|$ as opening brackets. Thus only rules in R can be used to reduce $x\{p^1a_iy$ to $x|a_iy$. As in the proof of Lemma 20 we conclude $x\{p^1a_iy \rightharpoonup^*_{R_p} x|a_iy$. Since R_p is unconditional we have $\rightharpoonup_{R_p} = \rightarrow_{R_p}$ and again Lemma 19 finishes the proof. □

We now determine the solutions of the rules (4) – (6). As indicated above all $\lambda\_w)$ with $w \in \Sigma^*$ are solutions of (4), the solutions of (5) and (6) have a similar form. But it is difficult to show that there are no other minimal solutions of (4) resp. (5), (6).

The main part of the proof is to show that we need exactly one reduction to evaluate the premise, if we want to apply one of the rules (4) – (6).

Lemma 22 $ua_i)(a_jv \rightharpoonup^*_C ua_ia_jv$ *if and only if* $ua_i)(a_jv \rightharpoonup^1_C ua_ia_jv$
proof: The 'if' direction is trivial, thus let us have a look at the 'only if' direction. We have $w_1 = ua_i)(a_jv \rightharpoonup^n_C ua_ia_jv = w_2$. Since $w_1 \neq w_2$ n is greater than 0, $n = 1$ gives the result, hence it remains to show that $n > 1$ is not possible.

To prove this is a longish task, hence we only give the main idea here, the complete proof can be found in the appendix. w_2 has one opening parenthesis less than w_1, hence exactly one of the the rules of T has to be used in this reduction, i.e. we have $w_3, w_4 \in \Delta^*$ such that $w_1 \rightharpoonup^*_R w_3 \rightharpoonup^1_T w_4 \rightharpoonup^*_R w_2$. By exhaustive case analysis we can show that we must have $w_1 = w_3$ and $w_4 = w_2$, implying $w_1 \rightharpoonup^1_C w_2$. □

In the reduction $w_1 = ua_i)(a_jv \rightharpoonup^1_C ua_ia_jv$ one pair of parentheses is removed, hence we use one of the rules (3) – (6) in this reduction step. Furthermore, $a_i)(a_j$ is the occurrence in w_1 where the reduction applies. This can be used to show

Lemma 23 *Let* $w_1 = ua_i)(a_jv \rightharpoonup_C ua_ia_jv = w_2$ *then* $u \in \Delta^*(\Sigma^*, v \in \Sigma^*)\Delta^*$.
proof: by induction on the length l of w_1.
Since there is no rule in (3) – (6) with a left-hand side shorter than 6 letters, l must be at least 6. If $w_1 = (a_i)(a_j)$ we use (3) to get (a_ia_j), if $w_1 \neq (a_i)(a_j)$, w_1 is irreducible by (3) – (6) since (3) cannot be applied because of its left-hand side and for (4) – (6) the left part of the premise cannot be reduced to the right part.

Now let us assume that the lemma is true for all w with $6 \leq |w| < l$. There are 4 cases according to the rule used in the reduction of $w_1 \rightharpoonup w_2$.

(3) i.e. $ua_i)(a_jv = u'(a_i)(a_j)v'$ and we are ready.

(4) i.e. $ua_i)(a_jv = u'(a_i)(a_ja_kv'$. Rule (4) can be applied if $u'(a_i)(a_kv' \rightharpoonup_C^* u'(a_ia_kv'$. By lemma 22 we get $u'(a_i)(a_kv' \rightharpoonup_C^1 u'(a_ia_kv'$ and by induction hypothesis $v' \in \Sigma^*)\Delta^*$ and thereby $v \in \Sigma^*)\Delta^*$. Since $u = u'(\in \Delta^*(\Sigma^*$ we are finished.

(5) i.e. $ua_i)(a_jv = u'a_ka_i)(a_j)v'$, similar to the case of rule (4).

(6) i.e. $ua_i)(a_jv = u'a_ka_i)(a_ja_lv'$. We may apply rule (6) if $u'a_k)(a_lv' \rightharpoonup_C^* ua_ka_lv'$. By lemma 22 and by induction hypothesis we get $v' \in \Sigma^*)\Delta^*, u' \in \Delta^*(\Sigma^*$. Hence $v \in \Sigma^*)\Delta^*, u \in \Delta^*(\Sigma^*$. □

Evaluation of the premise does not use the Δ^*-parts in the lemma above and we have $(w_1a_i)(a_jw_2) \rightharpoonup_C (w_1a_ia_jw_2)$ for all $w_1, w_2 \in \Sigma^*$. But we may not omit one or both of the outside parentheses. Since (w_1 and w_2) are irreducible, we thus have determined the minimal solutions of the rules (4) – (6):

Lemma 24
$sol_C(4) = \{\lambda\_w_r)|w_r \in \Sigma^*\}$
$sol_C(5) = \{(w_l\_\lambda|w_l \in \Sigma^*\}$
$sol_C(6) = \{(w_l\_w_r)|w_l, w_r \in \Sigma^*\}$.

As the next step we show that $\rightharpoonup_C$ is noetherian and decidable. The usual way to prove this is to show first that $\rightharpoonup_C$ is reductive and then to apply lemma 7. But we have not been able to find an appropriate ordering. Thus we have to show explicitly that $\rightharpoonup_C$ is decidable and noetherian.

Lemma 25 *$\rightharpoonup_C$ is decidable and noetherian.*
proof: There could be two sources of infinite computations using $\rightharpoonup_C$. It may be non-noetherian as an ordinary unconditional semi-Thue system. Second, in general it is undecidable whether a conditional rule may be applied, see lemma 6. When evaluating a premise we might try to apply a conditional rule, the premise of which is evaluated by use of just another conditional rule and so on.

Now let us assume that there is an infinite computation. There is no rule which increases the number of opening brackets (, [, neither by replacing the left-hand side by its right-hand side, nor by evaluating its premise. Hence there must be a word w with a minimal number of these brackets which starts an infinite computation.

At first we show that w cannot be reduced ad infinitum. None of the rules in T can be used to reduce w, since they decrease the number of brackets. Furthermore this number is smaller in the premise of rule (1) than in its left-hand side. Therefore it is decidable whether we may apply rule (1). It can be applied if and only if we used rules in R_p to evaluate its premise. Hence there is an infinite reduction using rules in R only, which is a contradiction.

If there is an infinite reduction it results from an infinite evaluation of premises. This does not concern the application of rule (1), see above. Since w contains a finite number of opening brackets, at least one of them must be involved infinitely often in this computation. Let us have a closer look at the left-most of these brackets. We have $w = xa_i)(a_jy$. x and y are split into x_1, x_2 resp. y_1, y_2 such that x_2, y_1 are of maximal length and in Σ^*.

Using the rules in C there is no possibility to increase a_jy_1, neither by reduction nor by evaluation of premises. Regarding x_2 the situation is worse, rules in R_f and R_p may produce new letters in Σ as suffix of x_1. But this cannot happen infinitely often, hence we can split x into $x'_1x'_2$ such that x'_2 is the maximal suffix which is reducible to a word in Σ^*. Again there is no possibility to increase the length of x'_2a_i.

But each evaluation of a premise of (4) – (6) when reducing $xa_i)(a_jy$ removes one or both of a_i, a_j, thus decreasing the length of x'_2a_i or a_jy_1. We get a new x'_2 or y_1, but this cannot be repeated ad infinitum. If x_2' and y_1 are empty, we can no longer apply a rule (4) – (6) due to the form of the left-hand sides. Hence these parentheses cannot be involved in an infinite computation, contradicting our assumption. □

To show that $\rightharpoonup_C$ is confluent and thereby canonical it suffices to show that $\rightharpoonup_C$ is locally confluent. We proceed as described in section 4.2. Expanding the rules (1) and (4) – (6) we get the system C_e, which will be shown to be locally confluent. Then as a direct consequence $\rightharpoonup_C$ is locally confluent too. The expanded rules are

$$\begin{array}{rcll l}
(a_iw) & \rightarrow & [q^1a_iw) & \text{for } a_i \in \Sigma, w \in \Sigma^*, a_iw \neq \widehat{a_iw} & (1e)\\
(a_i)(a_ja_kw) & \rightarrow & (a_ia_ja_kw) & \text{for } a_i, a_j, a_k \in \Sigma, w \in \Sigma^* & (4e)\\
(wa_ia_j)(a_k) & \rightarrow & (wa_ia_ja_k) & \text{for } a_i, a_j, a_k \in \Sigma, w \in \Sigma^* & (5e)\\
(w_la_ia_j)(a_ka_lw_r) & \rightarrow & (w_la_ia_ja_ka_lw_r) & \text{for } a_i, a_j, a_k, a_l \in \Sigma, w_l, w_r \in \Sigma^* & (6e)
\end{array}$$

Lemma 26 *The system C_e is locally confluent.*
proof: There are only two kinds of overlaps:

1) We have the overlap $(w_1)(w_2), w_1, w_2 \in \Sigma^+$. (w_1) is reducible by $(1e)$, $(w_1)(w_2)$ is reducible by one of $(3), (4e) - (6e)$. (The case (w_2) reducible by $(1e)$ is analogous). The critical pair is $[q_1w_1)(w_2) = (w_1w_2)$. Since $[q_1w_1)(w_2) \rightarrow_{C_e}^* (\widehat{w_1})(w_2) \rightarrow_{C_e} (\widehat{w_1}w_2) \rightarrow_{C_e}^* (\widehat{\widehat{w_1}w_2}) = (\widehat{w_1w_2})$ and $(w_1w_2) \rightarrow_{C_e}^* (\widehat{w_1w_2})$ the critical pair is joinable.
2) We have the overlap $(w_1)(w_2)(w_3), w_1, w_2, w_3 \in \Sigma^+$. $(w_1)(w_2)$ and $(w_2)(w_3)$ are reducible by $(3), (4e) - (6e)$. The corresponding critical pair is $(w_1w_2)(w_3) = (w_1)(w_2w_3)$ and we have $(w_1w_2)(w_3) \rightarrow_{C_e} (w_1w_2w_3), (w_1)(w_2w_3) \rightarrow_{C_e} (w_1w_2w_3)$. Hence this critical pair is joinable. □

Lemma 27
a) $\rightharpoonup_C$ is locally confluent.
b) $\rightharpoonup_C$ is canonical.

We now turn to the determination of properties of $\rightarrow_C$. It is easy to see that $\rightharpoonup_C \subseteq \rightarrow_C$, hence all solutions modulo $\rightharpoonup_C$ of the conditional rules are solutions modulo $\rightarrow_C$ too. But there are additional solutions, hence $\rightharpoonup_C \subsetneq \rightarrow_C$. Let us give a typical example: We take $w_{1c} \in \Sigma_c^*, w_2 \in \Sigma^+$, then $[w_{1c}q^1w_2a_i)(a_j)$ is irreducible modulo $\rightharpoonup_C$. But $[w_{1c}q^1w_2a_j) \rightarrow_C^* (\widehat{w_1w_2})(a_j) \rightarrow_C^* (w_1\widehat{w_2}a_j)$ and $[w_{1c}a_ia_j) \rightarrow_C^* (w_1\widehat{w_2}a_j)$. Hence we may apply rule (5) to reduce $[w_{1c}q^1w_2a_i)(a_j)$ to $[w_{1c}q^1w_2a_ia_j)$. Remark that $[w_{1c}q^1w_2a_i)(a_j)$ has the same irreducible descendant modulo $\rightharpoonup_C$ as well as modulo $\rightarrow_C$: $[w_{1c}q^1w_2a_i)(a_j) \rightharpoonup_C^* (w_1\widehat{w_2}a_i)(a_j) \rightharpoonup_C (w_1\widehat{w_2}a_ia_j) \rightharpoonup_C (w_1\widehat{w_2a_i}a_j)$ and $[w_{1c}q^1w_2a_i)(a_j) \rightarrow_C [w_{1c}q^1w_2a_ia_j) \rightarrow_C^* (w_1\widehat{w_2a_i}a_j)$.

Using the same proof as in lemma 25 we can show

Lemma 28 *$\rightarrow_C$ is decidable and noetherian*

To prove confluence we show that $\rightharpoonup_C$ and $\rightarrow_C$ are equivalent and both have the same set of normalforms. Since $\rightharpoonup_C$ is canonical and $\rightarrow_C$ is noetherian we can conclude that $\rightarrow_C$ is confluent too.

$\rightarrow_C$ is decidable and noetherian, hence there is a noetherian ordering $>$, such that $x > y$ if $x \rightarrow_C y$ or x contains the left-hand side of a conditional rule and y one part of the corresponding premise. That is, let $u = v :: l \rightarrow r$ be a conditional rule, then $z_1lz_2 > z_1uz_2, z_1vz_2, z_1rz_2$. By $>_{lex}$ we denote the lexicographical extension of $>$ on tuples of words.

Lemma 29 $\leftrightarrow_C^* = \rightleftharpoons_C^*$

proof: Since $u \rightharpoonup_C v$ implies $u \rightarrow_C v$ we only have to show $\leftrightarrow_C^* \subseteq \rightleftharpoons_C^*$. The proof is by noetherian induction on tuples of words which are equivalent modulo $\rightarrow_C$. Thereby we use the lexicographic extension to tuples of $\rightharpoonup_C$ as well-founded ordering. It suffices to show that $w_1 \rightarrow_C w_2$ implies $w_1 \rightleftharpoons_C^* w_2$. For the unconditional rules we have $\rightarrow_C = \rightharpoonup_C$.

Thus let us assume we used a conditional rule $u = v :: l \rightarrow r$, i.e. $w_1 = xly, w_2 = xry$. If x or y are reducible by $\rightharpoonup_C$, then let $\overline{x}$ resp. $\overline{y}$ denote the normalforms of x and y mod $\rightharpoonup_C$. We have $xly \rightharpoonup_C^* \overline{x}l\overline{y}, xry \rightharpoonup_C^* \overline{x}r\overline{y}$ and since $\rightharpoonup_C \subseteq \rightarrow_C$, $xly \rightarrow_C^* \overline{x}l\overline{y}, xry \rightarrow_C^* \overline{x}r\overline{y}$. By transitivity of $\rightarrow_C$ we get $\overline{x}l\overline{y} \leftrightarrow_C^* \overline{x}r\overline{y}$. Since $(\overline{x}l\overline{y}, \overline{x}r\overline{y}) <_{lex} (xly, xry)$ we may apply the induction hypothesis and get $\overline{x}l\overline{y} \rightleftharpoons_C^* \overline{x}r\overline{y}$ and again by transitivity $xly \rightleftharpoons_C^* xry$.

Now, let x and y be irreducible modulo $\rightharpoonup_C$. We have $xly \rightarrow_C xry$ and by definition $xuy \downarrow_C xvy$. Again by definition $(xuy, xvy) <_{lex} (xly, xry)$, hence $xuy \rightleftharpoons_C^* xvy$ and since $\rightharpoonup_C$ is confluent $xuy \downharpoonleft_C xvy$. There are 4 cases according to the rule used in the reduction $xly \rightarrow_C xry$.

(1) i.e. $l = (a_i$

x and y are irreducible, therefore $x|a_iy$ is irreducible modulo $\rightharpoonup_C$ too. Hence $xuy \rightharpoonup_C^* xvy$ and $xly \rightharpoonup_C xry$.

(4) i.e. $l = (a_i)(a_ja_k$

If $xuy \rightharpoonup_C^* xvy$ we are finished, thus let us assume $xuy \not\rightharpoonup_C^* xvy$. This implies that $xvy = x(a_ia_ky$ is reducible by $\rightharpoonup_C$. Since x, y are irreducible, this reduction has to use $(a_ia_k$ as part of a left-hand side or when evaluating a premise. We have to use a conditional rule, because there is no unconditional rule overlapping with $(a_ia_k$. Due to the form of the solutions we have $y = y_1)y_2, y_1 \in \Sigma^*$. Hence $xuy = x(a_i)(a_ky_1)y_2$ and $xuy \rightharpoonup_C xvy$, implying $xly \rightharpoonup_C xry$

(5) i.e. $l = a_ia_j)(a_k)$

Thereby we have $v = a_ia_k)$. If we use a conditional rule to reduce xvy we may argue as above. But now the rules $p^1a_i \rightarrow d_ip^1$ and $q^1a_i \rightarrow c_iq^1$ may also be applied to $xa_ia_k)y$. Notice that these are the only possibilities to reduce $xa_ia_k)y$ by an unconditional rule.

$q^1a_i \rightarrow c_iq^1$: x can be split into $x_1\delta x_{2c}$, where x_{2c} is the maximal suffix of x in Σ_c^* and δ is the letter to the left of x_{2c}, it may be λ. We have the reductions $xvy = x_1\delta x_{2c}q^{1\,\neg};a_k)y \rightharpoonup_{R_f}^* x_1\delta q^2x'_{2c}c'_ic'_k]y$ and $xuy = x_1\delta x_{2c}q^1a_i)(a_k)y \rightharpoonup_{R_f}^* x_1\delta q^2x'_{2c}c'_i](a_k)y$. These words should be joinable, but if $\delta \neq [$, the first one is irreducible and each successor of the second one has $x_1\delta q^2x'_{2c}c'_i]$ as a prefix. This is not equal to $x_1\delta q^2x'_{2c}c'_ic'_k]$, hence they cannot be joined, contradicting our assumption $\delta \neq [$.

For $\delta = [$, xly and xry are joinable:
$xly = x_1[x_{2c}q^1a_ia_j)(a_k)y \rightharpoonup_C^* x_1(x_2\widehat{a_ia_j})(a_k)y \rightharpoonup_C^* x_1(x_2\widehat{a_ia_j}a_k)y$ and
$xry = x_1[x_{2c}q^1a_ia_ja_k)y \rightharpoonup_C^* x_1(x_2\widehat{a_ia_j}a_k)y$.

$p^1a_i \rightarrow d_ip^1$: Analogously to the case above we may split x into $x_1\delta x_{2d}, x_{2d} \in \Sigma_d^*$. $\delta \neq \{$ gives the same contradiction as above. For $\delta = \{$ we get the successors $x_1|x_2a_i)(a_k)y$ and $x_1|x_2a_ia_k)y$ of xuy resp. xvy, again a contradiction. There may be successors $x_1|\$x_2a_i)(a_k)y$ and $x_1|\$x_2a_ia_k)y$, but this does not change the situation. Hence xvy cannot be reduced by $p^1a_i \rightarrow d_ip^1$.

(6) i.e. $l = a_ia_j)(a_ka_l$

Again we will only look at the case $xuy \not\rightharpoonup_C xvy$. xvy must be reducible and the reduction must concern $v = a_ia_l$. Using a conditional rule as in the case of rule (4) we get $xly \rightleftharpoons_C^* xry$. Assume that we use the rule $q^1a_i \rightarrow c_iq^1$. We split x into $x_1\delta x_2$ and similarly y into $y_1\gamma y_2$ with $y_1 \in \Sigma^*$. If $\delta = [$ and $\gamma =)$ then we have $xly \rightleftharpoons_C^* xry$, otherwise we get a contradiction as in the case of rule (5). The rule $p^1a_i \to d_ip^1$ cannot be used by an argument similar to those above.

Hence reducibility of xvy implies $xly \rightleftharpoons_C^* xry$ or contradicts $xuy \downarrow_C xvy$. $xuy \rightharpoonup_C^* xvy$ implies $xly \rightharpoonup_C xry$ and therefore we have $\leftrightarrow_C^* \subseteq \rightleftharpoons_C^*$, finishing the proof. □

Lemma 30 *$x \in \Delta^*$ is irreducible modulo $\to_C$ if and only if it is irreducible modulo $\rightharpoonup_C$.*
proof: If x is irreducible modulo $\to_C$ then it is irreducible modulo $\rightharpoonup_C$ too. Let us assume that x is reducible by $\to_C$, but not by $\rightharpoonup_C$, i.e. $x \to_C y$. By lemma 29 we have $x \rightleftharpoons_C^* y$. x is irreducible, hence $y \rightharpoonup_C^* x$. Since $\rightharpoonup_C \subseteq \to_C$ it follows $y \to_C^* x$ and $x \to_C y \to_C^* x$, contradicting termination of $\to_C$ □

Two words x, y which are equivalent modulo $\to_C$ are also equivalent modulo $\rightharpoonup_C$ and they have a common irreducible descendant z. Due to the lemmata above we have $x \to_C^* z \leftarrow_C^* y$, giving us the confluence of $\to_C$.

Lemma 31
a) $\to_C$ is confluent.
b) $\to_C$ is canonical.

It remains to show that for $u, v \in \Sigma^*, u \leftrightarrow_S^* v$ if and only if $\hat{\varphi}([u]_S) = \hat{\varphi}([v]_S)$, i.e. $\varphi^*(u) \leftrightarrow_C^* \varphi^*(v)$. Since $\to_C$ is canonical we have to show $u \leftrightarrow_S^* v$ if and only if $\varphi^*(u) \downarrow_C^* \varphi^*(v)$. To do this we first show that C computes the representative of concatenated configurations correctly.

Lemma 32 *For $u_1, \ldots, u_n \in \Sigma^*$ we get $(u_1)\ldots(u_n) \to_C^* (u_1\widehat{\ldots}u_n)$.*
proof: We have $(u_1)(u_2)\ldots(u_n) \to_C (u_1u_2)\ldots(u_n) \to_C^* (u_1\ldots u_n) \to_C^* (u_1\widehat{\ldots}u_n)$ □

The next lemma shows that M is embedded in M', i.e. $u \leftrightarrow_S^* v$ if and only if $\hat{\varphi}([u]_S) = \hat{\varphi}([v]_S)$.

Lemma 33 *For $u, v \in \Sigma^*$ we get $u \leftrightarrow_S^* v$ if and only if $\varphi^*(u) \downarrow_C \varphi^*(v)$.*
proof:

if . Let us assume first that $u, v \neq \lambda$, i.e. $u = a_1\ldots a_n, v = b_1\ldots b_m, n, m \geq 1$. Then $\varphi^*(u) = (a_1)\ldots(a_n) \to_C^* (a_1\widehat{\ldots}a_n) = (\hat{u})$ and similarly $\varphi^*(v) \to_C^* (\hat{v})$. If $\hat{u}, \hat{v} \neq \lambda$, then $(\hat{u}), (\hat{v})$ are irreducible and we conclude $(\hat{u}) = (\hat{v})$. We have $u \leftrightarrow_S^* \hat{u} = \hat{v} \leftrightarrow_S^* v$.
If $\hat{u} = \lambda$ then $(\hat{u}) = () \to_C \lambda$, which is irreducible. Hence we have $(\hat{v}) \to_C^* \lambda$. Since $(\hat{v})$ cannot be reduced by rule (1), it must be reduced by $() \to \lambda$, implying $\hat{v} = \lambda$. Again we have $\hat{u} = \hat{v}$ and we may conclude $u \leftrightarrow_S^* v$.
Now let us assume that $u = \lambda$ and $v \neq \lambda$. Then $\varphi^*(u) = \lambda$, which implies $\varphi^*(v) \to_C^* (\hat{v}) \to_C \lambda$. As above we have $\hat{v} = \lambda$ and $v \leftrightarrow_S^* \lambda = u$.

only if : We have to distinguish $u, v \neq \lambda$ and u or $v = \lambda$.
Let us assume that $u, v \neq \lambda$, i.e. $u = a_1\ldots a_n, v = b_1\ldots b_m, n, m \geq 1$. Then $\varphi^*(u) = (a_1)\ldots(a_n) \to_C^* (a_i\widehat{\ldots}a_n) = (\hat{u})$ and $\varphi^*(v) = (b_1)\ldots(b_n) \to^* (b_1\widehat{\ldots}b_n) = (\hat{v})$. Since $u \leftrightarrow_S^* v$ we have $\hat{u} = \hat{v}$ and thereby $\varphi^*(u) \downarrow_C \varphi^*(v)$.
If both $u = v = \lambda$ then there is nothing to show, thus let us assume $u = \lambda, v = b_1\ldots b_m \neq \lambda$. Since $v \leftrightarrow_S^* u = \lambda$ we have $\hat{v} = \lambda$. Hence $\varphi^*(v) = (b_1)\ldots(b_n) \to^* (b_1\widehat{\ldots}b_n) = () \to_C \lambda = \varphi^*(u)$. □

C is a finite and canonical presentation of M' and M is embedded in M', finishing the proof of theorem 4.

6.3 Complexity Issues

By lemma 33 it is possible to solve the word problem of the monoid M presented by (Σ, S) using the conditional system C. But how does the complexity of the use of C relate to the complexity of the word problem? We will show, that the complexity of the word problem (if it is at least exponential) is of the same complexity as the solution of the word problem using the system C.

To state the results, we use the complexity classes $E_n, n \geq 0$, of the Grzegorczyk hierarchy, see e.g. [Wei74]. Avenhaus and Madlener showed in [AM77, AM78], that the complexity of the word problem of a monoid is independent of its representation, therefore we may speak of the complexity of the word problem of the monoid M. Bauer and Otto [BO84] showed, that this complexity may be arbitrarily large.

Slightly varying a definition in [BO84], a finite conditional semi-Thue system T over an alphabet Γ is *E_n-bounded*, $n \geq 1$, if there exists a function $k \in E_n\{\Gamma\}$, such that for all $u \in \Gamma^*$, $|k(u)|$ reductions are sufficient to reduce u to its normalform. The function k gives the length of the sequence $u \rightarrow_C^* \hat{u}$ as well as the number of reductions which are necessary to evaluate the premises of conditional rules used in the sequence.

Lemma 34 *Let M be a monoid which is finitely generated by (Σ, S) and the complexity of the word problem of M is bounded above by a function in Σ_n, then the corresponding conditional system C is E_m-bounded, with $m = max\{3, n\}$.*

proof: Let $w \in \Delta^*$. To proof the lemma it is sufficient to give a reduction sequence $w \rightarrow_C^* \hat{w}$ which uses no more than $|k(w)|$ reductions with $k \in E_{max\{3,n\}}$. To reduce w to $\hat{w}$ we will use at first the rules in $R_f \cup R_p$ as far as possible, then those in T and at last the rules in R. Finally we use the rule $() \rightarrow \lambda$ again. There may be several cycles of this kind.

The reductions with R_f and R_p do not overlap, hence we have at most $|w|$ distinct reduction sequences modulo $R_f \cup R_p$. Each of these sequences simulates a Turing machine computing the functions f resp. p, see page 11. Since the number of words smaller than a given word u is exponential in the length of u, f is in the complexity class $E_{max\{3,n\}}$. As it is shown in [She65] the Turing machines M_f and M_p stop after at most $|g_f(u)|$ resp. $|g_p(u)|$ steps when started from an arbitrary configuration, with g_f resp. g_p in $E_{max\{3,n\}}$. Hence each of the reduction sequences using $R_f \cup R_p$ is of length of at most $|g_f(w)|$ resp. $|g_p(w)|$.

After performing these reductions only the rules (1) – (6) can be applied. That is, we have substrings of the form (w') or $(w_1)(w_2)$ with $w', w_1, w_2 \in \Sigma^*$. At first we remove empty configurations using rule (2), at most $|w|$ reductions with this rule are possible. Then we concatenate all adjoining configurations using the rules (3) – (6). Again at most $|w|$ reductions are possible, but we may have to evaluate the premises of conditional rules. But $|w|$ reductions are sufficient to evaluate the premise of one of the rules (4) – (6), hence there are at most $|w|^2$ reductions in this phase of reduction, but $|w|^2$ is a function in E_2.

To use rule (1) we have to evaluate its premise using R_p, and to compute the representatives we have to use R_f. Following the argumentation above we have at most $|w|$ distinct reduction sequences, which are bounded above in length by functions in $E_{max\{3,n\}}$.

As the representative may be λ, it may be possible to use the rule $() \rightarrow \lambda$ again, we get a word w'. Now, w' may be irreducible, summarizing the number of reductions we see that this number is bounded by a function k with $k \in E_{max\{3,n\}}$.

However it is possible, that w' is reducible again by $R_f \cup R_p$, as there may be configurations which are nested within another. But in w' there is at least one pair of parentheses less than in w, hence there can be at most $|w|$ cycles of this kind. The number of reductions in each cycle is bounded above by a function in $E_{max\{3,n\}}$, hence the total number is bounded above by a function $k \in E_{max\{3,n\}}$, too. □

Notice, that if the word problem of M is in E_0, E_1 or E_2, then C may use an exponential number of reductions. Furthermore, the use of C to solve the word problem of M is not a pseudo-natural algorithm in the sense of [MO85]. It does not give us a derivation $u \leftrightarrow_S^* v$ if u and v are congruent modulo S.

6.4 Concluding Remarks

Up to now it is an open question whether we can restate the embeddability theorem by using right conditional systems only. This might be suggested by the systems we used in the example of section 5, as they are all right conditional. But there was no progress in this direction, though we used several alternatives instead of Turing machines, especially Post machines, see e.g. [SS63, Man74], and string rewriting systems as defined in [Sat91].

Notice that we do not have an identical embedding as in the theorem of Bauer. But this is no serious restriction. We may use a system $R_\varphi = \{a_i \rightarrow (A_i) | a_i \in \Sigma, A_i \text{ a copy of } a_i\}$ to simulate the embedding. Now $R_\varphi \cup C$ has the same properties as C itself, hence M is identically embeddable in a monoid M'', such that M'' has a finite, canonical, and conditional representation.

Acknowledgements

I would like to thank Prof. Madlener for initiating these investigations and Birgit Reinert for valuable discussion about various versions of the conditional system C.

References

[AM77] Jürgen Avenhaus and Klaus Madlener. Subrekursive Komplexität bei Gruppen; I. Gruppen mit vorgeschriebener Komplexität. *Acta Informatica*, 9:87–104, 1977.

[AM78] Jürgen Avenhaus and Klaus Madlener. Subrekursive Komplexität bei Gruppen: II. Der Einbettungssatz von Higman für entscheidbare Gruppen. *Acta Informatica*, 9:183–193, 1978.

[Bau81] Günther Bauer. *Zur Darstellung von Monoiden durch konfluente Regelsysteme*. PhD thesis, Fachbereich Informatik, Universität Kaiserslautern, 1981. in German.

[Bau85] Günther Bauer. n-level rewriting systems. *Theoretical Computer Science*, 40:85–99, 1985.

[BO84] Günther Bauer and Friedrich Otto. Finite complete rewriting systems and the complexity of the word problem. *Acta Informatica*, 21:521–540, 1984.

[Boo85] Ronald V. Book. Thue systems as rewriting systems. In *Proc. of 1st Rewriting Techniques and Applications*, pages 63–94. Springer, 1985. LNCS 202.

[Dav56] Martin D. Davis. A note on universal turing machines. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 167–175. Princeton Press, 1956.

[Dei92] Thomas Deiß. Conditional semi-Thue systems for presenting monoids. In Alain Finkel and Matthias Jantzen, editors, *Proc. of STACS'92*, volume 577 of *LNCS*, pages 557–565. Springer, 1992.

[Der81] Nachum Dershowitz. Termination of linear rewriting systems. In S. Even and O. Kariv, editors, *Proc. 8th ICALP*, pages 448–458. Springer, 1981. LNCS 115.

[DW83] Martin D. Davis and Elaine J. Weyuker. *Computability, Complexity, and Languages.* Academic Press, 1983.

[Esc86] Carola Eschenbach. Die Verwendung von Zeichenkettenordnungen im Zusammenhang mit Semi Thue Systemen. Technical Report 122, Universität Hamburg, Fachbereich Informatik, 1986. in German.

[Gan87] Harald Ganzinger. A completion procedure for conditional equations. Technical Report 234, Fachbereich Informatik, Universität Dortmund, 1987.

[Hue80] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797-821, oct 1980.

[JW86] Jean Pierre Jouannaud and Bernard Waldmann. Reductive conditional term rewriting systems. In *Proceedings of the 3rd IFIP Working Conference on Formal Description of Programming Concepts.* North-Holland, 1986.

[Kap84] Stéphane Kaplan. Conditional rewrite rules. *Theoretical Computer Science*, 33:175–193, 1984.

[Kap87] Stéphane Kaplan. Simplifying conditional term rewriting systems: Unification, termination and confluence. *Journal of Symbolic Computation*, 4:295–334, 1987.

[KH89] Hélène Kirchner and Miki Hermann. Computing meta-rules from crossed rewrite systems. Technical report, CRIN, Nancy, 1989.

[KN85] Deepak Kapur and Paliath Narendran. A finite Thue system with decidable word problem and without equivalent finite canonical system. *Theoretical Computer Science*, 35:337–344, 1985.

[Man74] Zohar Manna. *Mathematical Theory of Computation.* Computer Science Series. McGraw-Hill, 1974.

[MO85] Klaus Madlener and Friedrich Otto. Pseudo natural algorithms for the word problem for finitely presented monoids and groups. *Journal of Symbolic Computation*, 1:383–418, 1985.

[New42] M. H. A. Newman. On theories with a combinatorial definition of equivalence. *Annals of Mathematics*, 43(2):223–243, 1942.

[Sat91] Andrea Sattler-Klein. Divergence phenomena during completion. In Ronald V. Book, editor, *Proc. of 4th Rewriting Techniques and Applications*, pages 374–385. Springer, 1991. LNCS 488.

[She65] J. C. Shepherdson. Machine configuration and word problems of given degree of unsolvability. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 11:149–175, 1965.

[Sim87] C. C. Sims. Verifying nilpotence. *Journal of Symbolic Computation*, 3:231–247, 1987.

[SO87] Craig Squier and Friedrich Otto. The word problem in finitely presented monoids and finite canonical rewriting systems. In *Proc. of 2nd Rewriting Techniques and Applications*, pages 74–82. Springer, 1987. LNCS 256.

[Squ87] Craig Squier. Word problems and a homological finiteness condition for monoids. *Journal of pure and applied algebra*, 49:201–217, 1987.

[Squ88] Craig Squier. A finiteness condition for rewriting systems. Department of Mathematical Sciences, SUNY–Binghamton, Binghamton, NY 13901, 1988.

[SS63] J. C. Shepherdson and H. E. Sturgis. Computability of recursive functions. *Journal of the ACM*, 10:217–255, 1963.

[SS82] Jörg Siekmann and P. Szabo. A noetherian and confluent rewrite system for idempotent semigroups. *semigroup forum*, 25:83–110, 1982.

[Wei74] K. Weihrauch. Teilklassen primitiv-rekursiver Wortfunktionen. Technical Report 91, GMD Bonn, 1974.

Appendix: Proof of lemma 22

Lemma 22 *$ua_i)(a_jv \rightarrow_C^* ua_ia_jv$ if and only if $ua_i)(a_jv \rightarrow_C^1 ua_ia_jv$*
proof: The 'if' direction is trivial, thus let us have a look at the 'only if' direction. We have $w_1 = ua_i)(a_jv \rightarrow_C^n ua_ia_jv = w_2$ and we show that the case $n > 1$ cannot occur. Since $w_1 \neq w_2$, $n = 0$ cannot occur, $n = 1$ gives the result.

We construct a partition of w_1 into useful parts and garbage. The useful parts correspond to substrings of w_1 which could be used in a reduction, these substrings are separated by irreducible garbage. An useful part in w_1 is the occurrence of a left-hand side of a rule in $R \cup \{() \rightarrow \lambda\}$. Remark that each left-hand side of (3) – (6) contains the left-hand side of (1), hence we do not use the rules (3) – (6) to construct useful parts. This occurrence is extended to the left and right by letters which might be useful in a configuration. That is, we extend to the left by 'left' letters in $\Sigma_c \cup \Sigma_d \cup \{b_c, b_d, \$, \$_d\}$ up to the first opening bracket $(, \{, [$ or $|$. This bracket is included to the extension. To the right we extend by letters in $\Sigma \cup \Sigma'_c \cup \Sigma'_d \cup \{b'_c, b'_d, \$'_d\}$ up to the first closing bracket $), \},]$, again included. Remark, that for the rules (1) we do not extend to the left, for the rule $() \rightarrow \lambda$ not at all. This partition is unique, the useful parts u_i do not overlap. Two useful parts u_i, u_{i+1} are separated by 'garbage' g_i, which is irreducible: It cannot be reduced by a rule in $R \cup \{() \rightarrow \lambda\}$, otherwise it would contain a useful part. It cannot be reduced by (3) – (6) either, since then the left-hand side of (1) would be part of it. Thus we have $w_1 = g_0u_1g_1 \dots u_ng_n$. E. g. $w_1 = \$\,(a_i\,a_j)\,)\,c_k\,p^1a_l\,\}\,b'_c$ can be partitioned into $w_1 = g_0u_1g_1u_2g_2$ with $g_0 = \$, u_1 = (a_ia_j), g_1 =), u_2 = c_kp^1a_i\}$ and $g_2 = b'_c$. This example shows that the useful parts may be improper configurations: u_1 may be irreducible, u_2 does not contain a left bracket.

Since w_2 has one opening parenthesis less than w_1, exactly one rule in T has to be used when reducing $w_1 \rightarrow_C^* w_2$, i.e. there are w_3, w_4 such that $w_1 \rightarrow_R^* w_3 \rightarrow_C^1 w_4 \rightarrow_R^* w_2$. The reduction $w_3 \rightarrow_C w_4$ uses a rule in T, but if it is a conditional rule we may use all rules in C to evaluate the premise. The reductions $w_1 \rightarrow_R^* w_3$ and $w_4 \rightarrow_R^* w_2$ use only rules in R, as according to lemma 21 only rules in R itself are necessary to evaluate the premises of conditional rules.

Only the useful parts $u_1, \dots, u_n$ can be changed when reducing $w_1 \rightarrow_R^* w_3$, no g_j will be involved in a reduction. Hence $w_3 = g_0 u_1' g_1 \dots u_n' g_n$.

Applying the T-rule to reduce w_3 to w_4 changes at least one useful part, let u_{l+1} be the rightmost of these parts. Due to the form of the rules, either the useful part u_l or the garbage g_l to the left of u_{l+1} can be involved in this reduction. Notice, that the garbage g_{l+1} will not be changed. Hence, $u_l' g_l u_{l+1}'$ is reduced to a string U', this might be an useful part as well as garbage, we have $w_3 = g_0 u_1' g_1 \dots g_{l-1}\ u_l' g_l u_{l+1}'\ g_{l+1} \dots u_n' g_n \rightarrow_C g_0 u_1' g_1 \dots g_{l-1}\ U'\ g_{l+1} \dots u_n' g_n = w_4$.

While reducing $w_4 \rightarrow_R^* w_2$, again the u_i' and U' are reduced only, no g_j is reducible, hence $w_2 = g_0 u_1'' g_1 \dots g_{l-1}\ U''\ g_{l+1} \dots u_n'' g_n$.

Since $w_1 = ua_i)(a_j v$ we know that $(a_j$ is the prefix of some u_k. We now have to show that $k = l+1$. Let us assume to the contrary that $l + 1 < k$. Since $w_1 = ua_i)(a_j v$ and $w_2 = ua_i a_j v$ we have $u_{k+1} = u_{k+1}'', \dots, u_n = u_n''$, the rightmost useful part which is reduced in the reduction $w_1 \rightarrow_C^* w_2$ is u_k. u_k is reduced by rule (1), hence there is a $w \in \Sigma^*$ such that $u_k = (a_j w)$, it has to be reduced to $xa_i a_j w)$, where $x \in \Delta^*$. This reduction uses rules in R, strictly speaking rules in R_f. Looking at these rules we can see that this reduction is impossible, contradicting $w_1 \rightarrow_C^* w_2$. Analogously, $k < l+1$ gives a contradiction, hence $l + 1 = k$.

As above we have $u_1 = u_1'', \dots, u_{l-1} = u_{l-1}'', u_{l+2} = u_{l+2}'', \dots, u_n = u_n''$. Hence the only possibilities to use the rules of R are the reductions $u_l \rightarrow_R^* u_l', u_{l+1} \rightarrow_R^* u_{l+1}'$ and $U' \rightarrow_R^* U''$. We show now that we also have $u_l = u_l', u_{l+1} = u_{l+1}', U' = U''$. Thereby we distinguish the cases $g_l = \lambda$ and $g_l \neq \lambda$

$g_l \neq \lambda$:

Then $a_i)$ is the suffix of g_l, hence it does not overlap with u_l' and we have $u_l = u_l' = u_l''$. I.e. $g_l u_{l+1} = v_l a_i)(a_j v_{l+1}$ for $v_l, v_{l+1} \in \Delta^*$.

Let us assume $u_{l+1} = u_{l+1}'$, hence we have $w_1 = w_3$. Rule (3) cannot be used to reduce w_3, otherwise (a_i) would be suffix of g_l, but this is an useful part. Using one of the rules (4) – (6) gives us $U' = u_l v_l a_i a_j v_{l+1}$, but this does not contain a left-hand side of a rule except in u_l. Since $u_l = u_l''$ we have $U' = U''$ and $w_4 = w_2$, hence $w_1 \rightarrow_C^1 w_2$.

Let us assume to the contrary that $u_{l+1} \neq u_{l+1}'$. Due to the form of the rules $u_{l+1} = (a_j v_{l+1}$ must be reduced by (1). Lemma 20 gives us $v_{l+1} = w)$, with $w \in \Sigma^*, a_i w \neq \widehat{a_i w}$. Furthermore, $u_{l+1}' = (\widehat{a_i w})$, this is the unique descendant of u_{l+1} beginning with (.

If $\widehat{a_j w} = \lambda$ then $() \rightarrow \lambda$ must be used to reduce $g_l u_{l+1}'$, hence $U'' = g_l$ and $w_4 = g_0 u_1 \dots u_l g_l g_{l+1} u_{l+2} \dots u_n g_n$. Since $g_l = v_l a_i)$, $g_l g_{l+1}$ does not contain the left-hand side of a rule, hence w_4 is irreducible.

If $\widehat{a_j w} \neq \lambda$ then $u_l g_l u_{l+1}' = u_l v_l a_i)(\widehat{a_j w})$ and $U' = u_l v_l a_i \widehat{a_j w})$. As we have seen above, u_l is not reduced and $v_l a_i \widehat{a_j w})$ does not contain an useful part, hence U' and w_4 cannot be reduced to U'' resp. w_2.

It is easy to see that in both cases $w_4 \neq w_2$ and therefore we have $w_4 \not\rightarrow_R^* w_2$, contradicting our assumption $w_1 \rightarrow_C^* w_2$.

$g_l = \lambda$: i.e. $u_l u_{l+1} = v_l a_i)(a_j v_{l+1}$ for $v_l, v_{l+1} \in \Delta^*$.

We show that $u_l = u_l'$ and $u_{l+1} = u_{l+1}'$. The proof is by contradiction. First, let us assume $u_{l+1} \neq u_{l+1}'$. To reduce $u_{l+1} = (a_j v_{l+1}$ we have to use rule (1), hence $v_{l+1} = x_{l+1})$ with $x_{l+1} \in \Sigma^*, a_j x_{l+1} \neq \widehat{a_j x_{l+1}}$. Furthermore, u_{l+1}' has to begin with (, hence $u_{l+1}' = (\widehat{a_j x_{l+1}})$.

If $\widehat{a_j x_{l+1}} = \lambda$ then $U' = u_l'$ and $u_l = v_l a_i) \rightarrow_R^* U' \rightarrow_R^* U'' = v_l a_i a_j x_{l+1})$. Due to the construction, $v_l a_i$ must contain at most one state symbol δ or if there is no such symbol we have one opening parenthesis (and we take $\delta =$ (. Hence $v_l = v_l^1 \delta v_l^2$. We now show, that the reduction $u_l \rightarrow^*_R U''$ is not possible, thus contradicting $a_j\widehat{x_{l+1}} = \lambda$. According to δ we have the following cases.

- $\delta \in \{p^1, \ldots, p^6\}$: The reduction $u_l \rightarrow^*_R U''$ uses only rules in R_p. At least one rule is applied, shifting the state symbol to one side. Further reductions can only change δ to another letter or shift it in the same direction again. δ cannot be inserted again or shifted to its original position, hence $u_l \not\rightarrow^*_R U''$.
- $\delta \in \{p_0, \ldots, p_f\}$: The reduction has to insert letters from Σ, but the rules in R_p^2 use only letters of copies of Σ. These letters can be inserted only using rules of R_p^3, but then we have state symbols $p^3, \ldots, p^6$ and we cannot return to the original one which occurs in U'' , again $u_l \not\rightarrow^*_R U''$.
- $\delta =$ (: i.e. we have to use rule (1), this implies $v_l^1 = \lambda$ and $v_l^2 \in \Sigma^*$ such that $v_l^2 a_i \neq \widehat{v_l^2 a_i}$. $(v_l^2 a_i)$ has only one descendant which begins with (: $(\widehat{v_l^2 a_i})$. Since $a_j x_{l+1} \neq \lambda$ and the length of $\widehat{v_l^2 a_i}$ is less or equal than that of $v_l^2 a_i$ we have $v_l^2 a_i \neq v_l^2 a_i a_j x_{l+1}$, hence $u_l \not\rightarrow^*_R U''$.
- $\delta \in \{q^2, \ldots, q^4, q_0, \ldots, q_f\}$: To insert letters of Σ, each reduction which uses these states has to use a closing bracket], but we have $u_l = v_l a_i)$ and v_l contains no brackets, hence we cannot insert these letters and thereby $u_l \not\rightarrow^*_R U''$.
- $\delta = q^1$: q^1 can be shifted only rightward up to the parenthesis), then it is replaced by q^2. To insert it again we have to move q^2 to the left and then we change it to q_0. This implies that $v_l^1 \in [\Sigma_c^*$ and $v_l^2 \in \Sigma^*$. q_0 starts the computation of the representative of the corresponding word in Σ^*. This final configuration is irreducible by rule (1), hence we cannot insert q^1, and we have $u_l \not\rightarrow^*_R U''$.
- $\delta \in \{q^5, q^6\}$: These state symbols can be moved to the left, then they are deleted. To insert them again we have to use rule (1), initiating the computation of the representative of the word in Σ^* corresponding to u_l. Hence, when inserting q^5 or q^6 we have a word which is not longer than u_l. But u_l is shorter than U'', hence $u_l \not\rightarrow^*_R U''$.

We have shown, that the case $a_j\widehat{x_{l+1}} = \lambda$ cannot occur. Now, the same tedious task has to be done for $a_j\widehat{x_{l+1}} \neq \lambda$. In this case we have to use one of the rules (3) – (6) to reduce $u_l' u_{l+1}'$ to U', hence u_l' has a suffix in Σ^*, i.e. $u_l' = x_l x_l'), x_l' \in \Sigma^*$. We have $u_l u_{l+1} = v_l a_i)(a_j x_{l+1}) \rightarrow^*_{..} x_l x_l')(a_j\widehat{x_{l+1}})$ and $U' = x_l x_l' a_j \widehat{x_{l+1}}) \rightarrow^*_R v_l a_i a_j x_{l+1}) = U''$. Since $a_j x_{l+1} \neq a_j\widehat{x_{l+1}}$ we have to reduce $x_l x_l'$ by R. Therefore it contains a state symbol γ or an opening parenthesis (, if there is no state symbol we take $\gamma =$ (. There are $v_l'^1, v_l'^2 \in \Delta^*$ such that $u_l' = x_l x_l') = v_l'^1 \gamma v_l'^2)$. Analogously we have $u_l = v_l a_i) = v_l^1 \delta v_l^2 a_i)$.

As above we distinguish the following cases according to δ

- $\delta \in \{p^1, \ldots, p^6\}$: Each of these symbols is moved to one side using R_p, then it may be changed or there are no more rules which can be applied. In the first case we cannot insert it again, hence no descendant of u_l resp. U' begins with $v_l^1 \delta$ and we have $U' \not\rightarrow^*_R U''$.
- $\delta \in \{p_0, \ldots, p_f\}$: Only rules in R_p can be used to reduce u_l and U'. Therefore we cannot change the suffix $a_j\widehat{x_{l+1}})$ of U', but $a_j x_{l+1} \neq a_j\widehat{x_{l+1}}$, hence $U' \not\rightarrow^*_R U''$.
- $\delta =$ (: i.e. we have to use rule (1) to reduce u_l, hence $v_l^1 = \lambda, v_l^2 \in \Sigma^*$ such that $v_l^2 a_i \neq \widehat{v_l^2 a_i}$. There are only two descendants of $u_l u_{l+1}$ beginning with (: $(\widehat{v_l^2 a_i} a_j \widehat{x_{l+1}})$ and $(v_l^2 \widehat{a_i a_j} x_{l+1})$. The second one is irreducible, both are not equal to U'', again $U' \not\rightarrow^*_R U''$.

$\delta \in \{q^2, \ldots, q^4, q_0, \ldots, q_f\}$: The reductions $u_l \rightarrow_R^* u_l'$ and $U' \rightarrow_R^* U''$ are restricted to v_l since the rules corresponding to these state symbols cannot use the suffix $a_i)$ of u_l resp. the suffix in Σ^+ of x_l'.

$\delta = q^1$: This symbol can be only deleted or shifted to the right. Before inserting it again we have to use rule (1), this must be done in one of the reductions $u_l \rightarrow_R^* u_l'$ or $U' \rightarrow_R^* U''$. The word which is reduced by rule (1) must have the form (w) with $w \in \Sigma^*$ and $w \neq \hat{w}$. Therefore rule (1) cannot be applied in the reduction $u_l \rightarrow_R^* u_l'$, because the first descendant of u_l beginning with (is a word of the form (w), but with $w = \hat{w}$.

Now, there are only 4 possibilities for γ, we have $\gamma \in \{q^1, q^5, q^6, (\}$.

If $\gamma = q^1$ and rule (1) can be applied on a descendant of U', then $U' \in [\Sigma_c^* q^1 \Sigma^*)$. Therefore u_l is in $[\Sigma_c^* q^1 \Sigma^*)$ too, we have $u_l = [y_{1c} q^1 y_2)$ with $y_{1c} \in \Sigma_c^*, y_2 \in \Sigma^*$. Now the first descendant of U' beginning with (is the word $(y_1 y_2 \widehat{a_j} x_{l+1})$. This word is irreducible, thus contradicting that we use rule (1) in the reduction $U' \rightarrow_R^* U''$.

If $\gamma =$ (then again $u_l = [y_{1c} q^1 y_2)$ and $U' = (\widehat{y_1 y_2} a_j \widehat{x_{l+1}})$. If it were possible to reduce U' to $U'' = [y_{1c} q^1 y_2 a_j x_{l+1})$ we would have $a_j x_{l+1} = a_j \widehat{x_{l+1}}$, a contradiction. U' can be reduced to $(y_1 y_2 \widehat{a_j} x_{l+1})$, but this cannot be reduced to U'', since otherwise $(y_1 y_2 \widehat{a_j} x_{l+1}) \rightarrow_R^+ [y_{1c} q^1 y_2 a_j x_{l+1}) \rightarrow_R^* (y_1 y_2 \widehat{a_j} x_{l+1})$, contradicting termination of $\rightarrow_R$.

If $\gamma = q^5$ or $\gamma = q^6$, we can shift γ to the left and change it to (, we proceed as in the case $\gamma =$ (.

$\delta \in \{q^5, q^6\}$: Using the rules in R_f^3 we reach the left end of u_l and change the state symbol to (. To insert δ again we have to use rule (1). We proceed as in the case $\delta =$ (.

In all cases we have shown that $U' \not\rightarrow_R^* U''$ and thereby $w_1 \not\rightarrow_C^* w_2$. Hence our assumption $u_{l+1} \neq u_{l+1}'$ is false. But what about the case $u_l \neq u_l'$ and $u_{l+1} = u_{l+1}'$? Arguing as above this case is contradictory too.

Therefore $u_l = u_l' = v_l a_i), u_{l+1} = u_{l+1}' = (a_j v_{l+1}$ and $U' = v_l a_i a_j v_{l+1} = U''$. But now $w_1 = w_3, w_4 = w_2$, finishing the proof, we have $w_1 \rightarrow_C^1 w_2$. □