Let $(\mathcal{R}, <)$ be an <u>rws</u> on X^*

If $\mathcal{R}$ is confluent $\Rightarrow$ solving the word problem on $X^*/_{\mathcal{R}}$ is the same as fining the canonical forms w.r.t $<$ on X^* i.e. rewriting words w.r.t. $\mathcal{R}$.

If confluence for $\mathcal{R}$ fails $\Rightarrow$ local confluence fails at

$$u \xleftarrow{*} W \xrightarrow{*} v$$

<u>Proposition:</u> Suppose that local confluence fails at W, but doesn't for any proper subwords of W. Then one of the following holds:

1) W is the <u>lhs</u> for two different rules of $\mathcal{R}$.
2) W is the <u>lhs</u> for a rewrule in $\mathcal{R}$ and W contains <u>lhs</u> of a different rwrule as a <u>proper subword</u>.

3) $W = ABC$, $A, B, C \in X^*$, nonempty & AB, BC are <u>lhs</u>es of rwrules from $\mathcal{R}$.

<u>Proof:</u> By local confluence failure: $\exists$ words A_1, P_1, B_1, A_2, P_2, B_2 (A_i, B_i - possibly empty) s.t.

- $W = A_1 P_1 B_1 = A_2 P_2 B_2$
- $P_1 \to Q_1$, $P_2 \to Q_2$ $\in \mathcal{R}$
- There is no common word derivable from $U_1 = A_1 Q_1 B_1$ and $U_2 = A_2 Q_2 B_2$.

• If the occurences of P_1 an P_2 don't overlap

$\Rightarrow$
$$\begin{tikzcd}[column sep=small]
 & W = A_1 P_1 C P_2 B_2 \arrow[dl] \arrow[dr] & \\
A_1 Q_1 C P_2 B_2 \arrow[dr] & & A_1 P_1 C Q_2 B_2 \arrow[dl] \\
 & A_1 Q_1 C Q_2 B_2 &
\end{tikzcd}$$

$\lightning$.

$\Rightarrow$ $W = A_1\, \overbrace{A\,B}^{P_1}\hspace{-0.7em}\underbrace{\hspace{0.7em}C}_{P_2}\, B_2$, $B \neq \varepsilon$ and either

$(\cdot)$ $P_1 = AB$, $P_2 = BC$ $(A, C \neq \varepsilon)$, or

$(\cdot\cdot)$ $P_1 = ABC$, $P_2 = B$ $(A, C$ possibly empty$)$.

If $A_1 B_2 \neq \varepsilon \Rightarrow ABC$ is a proper subword $\Rightarrow$ local confluence for ABC doesn't fail.

if $(\cdot)$ holds then
$$\begin{tikzcd}[column sep=small]
 & ABC \arrow[dl] \arrow[dr] & \\
Q_1 C \arrow[dr, "*"] & & A Q_2 \arrow[dl, "*"'] \\
 & Q &
\end{tikzcd}$$

hence
$$\begin{tikzcd}[column sep=small]
 & W = A_1\, ABC\, B_2 \arrow[dl] \arrow[dr] & \\
U_1 = A_1 Q_1 C B_2 \arrow[dr] & & U_2 = A_1 A Q_2 B_2 \arrow[dl] \\
 & A_1 Q B_2 &
\end{tikzcd}$$

$\lightning$.

(Similarly for $(\cdot\cdot)$)

Suppose that $A_1 = B_2 = \varepsilon$ i.e. $W = ABC$

Suppose $(\cdot)$ holds.

$AC \neq \varepsilon \Rightarrow P_2$ is a proper subword of $P_1 \Rightarrow$ condition (2).

$AC = \varepsilon \Rightarrow P_1 = P_2 \Rightarrow$ condition (1)

$(\cdot\cdot) \Rightarrow$ condition (3).

$\square$

<u>Example</u>:

$X = \{x, y, z\}, \quad \mathcal{R} = \{xyz \to \varepsilon,\ yzx \to \varepsilon,\ zxy \to \varepsilon\}$

$$\begin{tikzcd} & xyzx \arrow[dl] \arrow[dr] & \\ x & \checkmark & x \end{tikzcd} \qquad \begin{tikzcd} & yzxy \arrow[dl] \arrow[dr] & \\ y & \checkmark & y \end{tikzcd} \qquad \begin{tikzcd} & zxyz \arrow[dl] \arrow[dr] & \\ z & \checkmark & z \end{tikzcd}$$

$$\begin{tikzcd} & xyzxy \arrow[dl] \arrow[dr] & \\ xy & \checkmark & xy \end{tikzcd} \qquad \begin{tikzcd} & yzxyz \arrow[dl] \arrow[dr] & \\ yz & \checkmark & yz \end{tikzcd} \qquad \begin{tikzcd} & zxyzx \arrow[dl] \arrow[dr] & \\ zx & & zx \end{tikzcd}$$

locally confluent.

$F_2 = \langle a, b, a^{-1}, b^{-1} \mid aa^{-1} = a^{-1}a = bb^{-1} = b^{-1}b \rangle$

$f(a) = x, \quad f(b) = y, \quad f(a^{-1}) = yz, \quad f(b^{-1}) = zx$

f jest a homomorphism,

$g(x) = a$, $g(y) = b$, $g(z) = b^{-1}a^{-1}$

$\underline{xyz \longmapsto \varepsilon}$

$f(g(x)) = f(a) = x$

$f(g(y)) = f(b) = y$

$f(g(z)) = f(b^{-1}a^{-1}) = f(b^{-1})\, f(a^{-1}) = zxyz \xrightarrow[R]{} z.$

$\Rightarrow$ Also $g \circ f = id$ $\Rightarrow$ $X^*/R \cong F_2$.

$X = \{x, y, z\}$

$x^2 \to \varepsilon$

$yz \to \varepsilon$

$zy \to \varepsilon$

$$\begin{tikzcd} & x \cdot x \cdot x \arrow[dl] \arrow[dr] & \\ x & & x \end{tikzcd} \quad \checkmark$$

$$\begin{tikzcd} & yzy \arrow[dl] \arrow[dr] & \\ y & & y \end{tikzcd} \qquad \begin{tikzcd} & zyz \arrow[dl] \arrow[dr] & \\ z & & z \end{tikzcd}$$

locally confluent.

$X = \{a, b\}$, $R = \{abab \to \varepsilon$, $baba \to \varepsilon\}$

$$\begin{tikzcd} & ababa \arrow[dl] \arrow[dr] & \\ a & & a \end{tikzcd} \qquad \begin{tikzcd} & ababab \arrow[dl] \arrow[dr] & \\ ab & & ab \end{tikzcd} \qquad \begin{tikzcd} & abababa \arrow[dl] \arrow[dr] & \\ aba & & aba \end{tikzcd}$$

the same for baba

$\longrightarrow$ locally confluent.

Example:

$$X = \{a, b\} \qquad R = \{\, a^2 \to \varepsilon,\ b^3 \to \varepsilon,\ abab \to b^2a,\ abba \to bab,\ baba \to ab^2,\ b^2ab^2 \to aba \,\}.$$

aaa: $\quad aaa \to a$, $\quad aaa \to a$

$aabab$: $\quad aabab \to bab$, $\quad aabab \to ab^2a \to bab$ ✓

$aabba$: $\quad aabba \to bba$, $\quad aabba \to abab \to b^2a$ ✓

$bbbb$: $\quad bbbb \to \varepsilon$, $\quad bbbb \to \varepsilon$

$bbbabb$: $\quad bbbabb \to abb$, $\quad bbbabb \to baba \to abb$ ✓

$bbbaba$: $\quad bbbaba \to aba$, $\quad bbbaba \to bbabb \to aba$ ✓

$ababa$: $\quad ababa \to b^2aa \to b^2$, $\quad ababa \to aab^2 \to b^2$ ✓

$ababab$: $\quad ababab \to b^2aab \to b^3 \to \varepsilon$, $\quad ababab \to abb^2a \to a^2 \to \varepsilon$ ✓

$ababba$: $\quad ababba \to bbaba \to bab^2$ ✓ $\quad ababba \to abbab \to babb$

$ababbb$: $\quad ababbb \to b^2ab^2 \to aba$, $\quad ababbb \to aba$ ✓

$abababa$: $\quad abababa \to b^2aaba \to b^3a \to a$, $\quad abababa \to abaab^2 \to ab^3 \to a$

$ababbab^2$: $\quad ababbab^2 \to b^2abab^2 \to b^2b^2ab \to bab$ ✓ $\quad ababbab^2 \to abaaba \to abba \to bab$ . . .

$X = \{ a, b, a^{-1}, b^{-1} \}$

$\mathcal{R} = \{ aa^{-1} \to \varepsilon,\ bb^{-1} \to \varepsilon,\ a^{-1}a \to \varepsilon,\ b^{-1}b \to \varepsilon,$

$ba \to ab, \quad ba^{-1} \to a^{-1}b$

$b^{-1}a \to ab^{-1}, \quad b^{-1}a^{-1} \to a^{-1}b^{-1}$

order: $\mathrm{lenlex}(a < a^{-1} < b < b^{-1})$

$(\mathcal{R}, <)$ is locally confluent.

$\mathcal{S} = \{ aa^{-1} \to \varepsilon, \ldots\ldots ,$

$ba \to ab, \quad a^{-1}b \to ba^{-1}$

$b^{-1}a \to ab^{-1}, \quad a^{-1}b^{-1} \to b^{-1}a^{-1}. \}$

order: $\mathrm{lenlex}(a < b < b^{-1} < a^{-1})$

$(\mathcal{S}, <)$ is not <u>locally</u> <u>confluent</u>!

$$
\begin{array}{ccc}
 & \underbrace{ba}\overbrace{aa^{-1}} & \\
\swarrow & & \searrow \\
aba^{-1} & & b
\end{array}
\quad \Longrightarrow \quad aba^{-1} \to b
$$

?

$$
\begin{array}{ccc}
 & b\underbrace{ba}\overbrace{aa^{-1}} & \\
\swarrow & & \searrow \\
\underbrace{ba}ba^{-1} & & bb \\
\downarrow & & \vdots \\
abba^{-1} & &
\end{array}
\quad \Longrightarrow \quad abba^{-1} \to bb
$$

This leads to an <u>infinite</u> <u>confluent</u> rws.

ALGORITHM: isconfluent

INPUT: $\mathcal{R}$ – a rewriting system

OUTPUT: true or false and a witness for confluence failure

```
begin
for (P1 → Q1) in rwrules(R)
  for S in suffixes(P1, 1:length(P1))
    for (P2 → Q2) in rwrules(R)
      W = lcp(S, P2)          // longest common prefix
      W = ε && continue
      if length(u) = length(S) // P2 starts with u
        A = P1[1:length(P1)-length(S)]
        B = P2[length(S)+1 : length(P2)]
        // ASB can be rewritten as
        //  Q1B      AQ2
        U = Rewrite(Q1B, R); V = Rewrite(AQ2, R)
        U ≠ W && return false, (ASB, U, W)
      elseif length(u) = length(P2) // P2 is a subword of S
        A = P1[1:length(P1)-length(S)]
        B = P1[length(A)+length(u)+1 : length P1]
        // P1 = A·u·B = A·P2·B rewrites as
        //       Q1          A·Q2
        U = Rewrite(Q1, R); V = Rewrite(A·Q2, R)
        U ≠ W && return false, (P1, U, W)
      end
    end
  end
end
return true; end.
```

Rewriting strategies

Given rws $(R,<)$ and W – a word to be rewritten. How to pick the order in which we choose rewrules in R to do so?

It could be an optimization problem:

- the result minimizes $<$.
- the result minimizes wl.

Since rewriting is done so often we will almost all pick the first one that fits

but we may periodically reorder rw-rules of R

→ usually we want to sort them w.r.t. wl of the lhs

ALGORITHM: destructive_rewrite. (rewrite from right)

input: U – word to be rewritten

R – rewriting system

output: V – $U \xrightarrow[R]{*} V$

```
begin
  V = zero(U)
  while !iszero(U)
    x = popfirst!(U)
    push!(V, x)
    for (P → Q) in rwrules(R)
      if P is a suffix of V
        prepend!(U, Q)
        resize!(V, length(V) - length(P))
        break
      end
    end
  end
  return V ; end
```

we are allowed to break here, as all rules of R have been checked against the suffixes of the current V.

Knuth-Bendix procedure

Given an RWS $(R, <)$ we want to compute $RC(R, <)$ – reduced, confluent rws which generates $\sim$ defined by $(R, <)$.

Algorithm : Knuth-Bendix

INPUT : $(R, <)$ – a finite rws

OUTPUT : $RC(R, <)$ – reduced, confluent rws

```
begin
   S = Rewriting system( )
   for (P -> Q) in rwrules(R)
       push!(S, P -> Q)
   end
   for (P1 -> Q1) in rwrules(S)
       for (P2 -> Q2) in rwrules(S)
           if (P2 -> Q2) = (P1 -> Q1)
               break
       end
       resolve_overlaps(S, P1, P2).
     end
   end
   return reduce(S)
end
```

Algorithm : push!

INPUT : $(\mathcal{R}, <)$ – rewriting system

$P \to Q$ – rwrule

OUTPUT : $(\mathcal{R}, <)$ which contains $(P \to Q)$.

```
begin
    U = rewrite(P, R)
    V = rewrite(Q, R)
    if U ≠ V
        U, V = U > V ? (U, V) : (V, U)
        add U → V to rewriting rules of R.
    end
    return R
end
```

Algorithm: resolve-overlaps

INPUT : $(\mathcal{R}, <)$ – rews

$P_1 \to Q_1$ – rwrule

$P_2 \to Q_2$ – rwrule

OUTPUT : $(\mathcal{R}, <)$ s.t. all rewrites using the rules above are locally confluent

```
begin
  for s in suffixes(P1, 1:length(P1))
      if isprefix(s, P2)
          //    A S B
          //  Q1 B   A Q2
          push!(R, Q1 B → A Q2)
      elseif occursin(P2, s)
          // P1 = A P2 B
          push!(R, Q1 → A Q2 B)
      end
  end
  return R; end.
```

ALGORITHM: reduce

INPUT: $(R,<)$ - an rws

OUTPUT: $(S,<)$ - a reduced version of $(R,<)$

```
Begin
    S = empty(R)
    for (P → Q) in rewrules(R)
        for P' in proper_subwords(P)
            if !is_irreducible(P', R)
                break
            end
        end
        push!(S, P → rewrite(R, Q))

    end
    return S
end
```

Proposition:

If $RC(R,<)$ is finite, then Knuth_Bendix terminates and returns it.

Proof: Since S is initially a subsystem, rewrites of $P \to Q$ in S follow also in R, so that we didn't change the equivalence relation $\sim$ generated by R.

During the while loop this property is preserved.

Suppose that Knuth_Bendix doesn't terminate.

$\Rightarrow$ there's an infinite sequence of rules U added to S.

Prop: $\mathcal{U}$ + the initial rules form a confluent rewriting system.

Proof: If $\mathcal{U}$ is not confluent let W be the least ($<$) word for which local confluence fails.

We know that $W = ABC$, where $B \neq \varepsilon$ and either

$$\left.\begin{array}{l} \bullet\ P_1 = ABC,\ P_2 = B \\ \\ \bullet\ P_1 = AB,\ P_2 = BC \end{array}\right\} \text{ for } \begin{array}{l} P_1 \to Q_1 \\ P_2 \to Q_2 \end{array} \in \mathcal{U}$$

and (*) there is no word derivable in S from applying the rewrites.

In either of cases at some point in the procedure a call to

resolve_overlaps(S, $P_1 \to Q_1$, $P_2 \to Q_2$)

is made thus contradicting (*).

$\square$

Let $\mathcal{C}$ be the set of canonical forms for $\sim$ generated by $(\mathcal{R}, <)$.

Let $\mathcal{P} \subset X^* \setminus \mathcal{C}$ be the set of words s.t. every proper subword is in $\mathcal{C}$.

for every $P \in \mathcal{P}$ there exists a rule $P \to Q$ ($Q \in \mathcal{C}$) in $\mathcal{U}$.
(no other rule shares P as lhs).

Let $\mathcal{V} = \{P \to Q \text{ from } \mathcal{U} \text{ s.t. } P \in \mathcal{P}\}$

$\nwarrow$ finite set.

If we pick $(P \to Q) \in \mathcal{U}$, $P \notin \mathcal{P}$

$\Rightarrow$ P is not in $\mathcal{C}$, but is also irreducible w.r.t. $\mathcal{V}$.

$\square$

Examples

1. $aA \Rightarrow \varepsilon$
2. $Aa \Rightarrow \varepsilon$
3. $bB \Rightarrow \varepsilon$
4. $Bb \Rightarrow \varepsilon$
5. $ba \Rightarrow ab$

$$\begin{tikzcd} & baA \arrow[dl] \arrow[dr] & \\ abA \arrow[rr] & & b \end{tikzcd} \qquad (5,1)$$

6. $abA \to b$

$$\begin{tikzcd} & Bba \arrow[dl] \arrow[dr] & \\ a & & Bab \arrow[ll] \end{tikzcd} \qquad (4,5)$$

7. $a \leftarrow Bab$

$$\left(\begin{tikzcd} & abAa \arrow[dl] \arrow[dr] & \\ ba \arrow[rr] & & ab \end{tikzcd} \right)$$

$$\begin{tikzcd} & AabA \arrow[dl] \arrow[dr] & \\ bA \arrow[rr] & & Ab \end{tikzcd} \qquad (2,6)$$

8. $bA \to Ab$

$$\begin{tikzcd} & BabB \arrow[dl] \arrow[dr] & \\ aB & & Ba \arrow[ll] \end{tikzcd} \qquad (7,3)$$

9. $aB \leftarrow Ba$

$$\begin{tikzcd} & BbA \arrow[dl] \arrow[dr] & \\ A & & BAb \arrow[ll] \end{tikzcd} \qquad (4,8)$$

10. $A \leftarrow BAb$

$$BabA \qquad (8,7)$$

$$aA \qquad B\underbrace{aA}b$$

$$\varepsilon \qquad Bb$$

$$\varepsilon$$

$$\begin{tikzcd} & BaA \arrow[dl] \arrow[dr] & \\ aBA \arrow[rr] & & B \end{tikzcd} \qquad (9,1)$$

11. $aBA \to B$

$$\begin{tikzcd} & \underbrace{BAb}B \arrow[dl] \arrow[dr] & \\ AB & & BA \arrow[ll] \end{tikzcd} \qquad (10,3)$$

12. $AB \leftarrow BA$

Second version:

Keep the set of rules *always* *reduced*.

Whenever we add a new rule – scan all of the others to determine those which become simpler / redundant. ⇒ push them to a stack ⇒ operate until stack is empty

---

ALGORITHM : append!

INPUT : • $(R,<)$ – reduced rws
• stack – a list of rules to be added

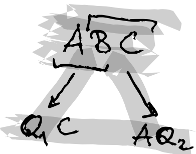
OUTPUT : • $(R,<)$ – reduced rws

```
begin
   while !isempty(stack)
        P → Q = pop!(stack)
        A = rewrite(P, R); B = rewrite(Q, R)
        if A ≠ B
            A, B = A > B ? (A, B) : (B, A)
            add A → B as rule to R
            for P → Q in active_rules(R)
                (P → Q) ≺ (A → B) && continue
                if occursin(A, P)        // rule is reducible
                    push!(stack, P → Q)
                    deactivate!(R, P → Q)
                elseif occursin(A, Q)
                    rewrite!(Q, A → B)   } in place
                    rewrite!(Q, R)       } modifications
                end
            end
        end
   return R
end
```

ALGORITHM: resolve_overlaps!

INPUT : • $(\mathcal{R}, <)$ - reduced rws
• $(P_1 \to Q_1)$ - rwrule
• $(P_2 \to Q_2)$ - rwrule
• stack - a stack of rules

OUTPUT : $(\mathcal{R}, <)$ - rws where all critical pairs from P_1 and P_2 are resolved




```
begin
   m = min(length(P1), length(P2))
   while isactive(P1 -> Q1) && isactive(P2 -> Q2)
      for B in suffixes(P1, 1:m-1)
         if isprefix(B, P2)
            A = P1[1:length(P1) - length(B)]
            B = P2[length(B)+1 : length(P2)]
            push!(stack, AQ2 -> Q1C)      # any ordering is fine
            append!(R, stack)
         end
      end
   end
   return R
end
```

ALGORITHM : Knuth-Bendix_always_reduced

INPUT : $(\mathcal{R}, <)$ – rws

OUTPUT : $RC(\mathcal{R}, <)$ – the unique, reduced, confluent rws.

```
begin
    stack = ∅
    for r in rules(R)
        push!(stack, r)
    end
    S = empty(R)
    append!(S, stack)

    for r₁ in active_rules(S)
        for r₂ in active_rules(S)
            isactive(r₁) || break
            isactive(r₂) || continue
            resolve_overlaps!(S, r₁, r₂, stack)
        end
    end
    delete inactive rules from S
    return S
end
```

Example:

$a^2 \to \varepsilon$

$b^3 \to \varepsilon$

$(ab)^7 \to \varepsilon$

$(abab^2)^8 \to \varepsilon$

hard

with

1, 2, 3, 5 – collapses

4 – 40 rules

6 – 119 rules

7 – 147 rules

8 – ???

9 – ???