An Optimized Decision Algorithm for Stratified Context Unification

Manfred Schmidt-Schauß

Fachbereich Informatik
Johann Wolfgang Goethe-Universität
Postfach 11 19 32, D-60054 Frankfurt
Germany
E-mail:schauss@ki.informatik.uni-frankfurt.de

Abstract. Context unification is a variant of second order unification. It can also be seen as a generalization of string unification to tree unification. Currently it is not known whether context unification is decidable. A specialization of context unification is stratified context unification, which is decidable. However, the previous algorithm has a very bad worst case complexity. Recently it turned out that stratified context unification is equivalent to satisfiability of one-step rewrite constraints.
This paper contains an optimized algorithm for stratified context unification exploiting sharing and power expressions. We prove that the complexity is determined mainly by the maximal depth of SO-cycles.
Two observations are used: i. For every ambiguous SO-cycle, there is a context variable that can be instantiated with a ground context of main depth $O(c * d)$, where c is the number of context variables, and d is the depth of the SO-cycle. ii. the exponent of periodicity is $O(2^n)$, which means it has an $O(n)$ sized representation.
From a practical point of view, these observations allow us to conclude that the unification algorithm is well-behaved, if the maximal depth of SO-cycles does not grow too large.

1 Introduction

Context unification is a variant of second order unification and also a generalization of string unification. There are unification procedures for the more general problem of higher-order unification (see e.g. [Pie73,Hue75,SG89,Wol93,Pre95]). It is well-known that general higher-order unification and second-order unification are undecidable [Gol81,Far91,LV99] and that string unification is decidable [Mak77]. Recent upper complexity estimations for string unification are NEXPTIME [Pla99a] and PSPACE [Pla99b].

Context unification problems are restricted second-order unification problems: context variables represent terms with exactly one hole in contrast to a term with an arbitrary number of (equally named) holes in the general case. The name *contexts* was coined in [Com93]. Currently, it is not known whether general context unification is decidable. There are some decidable fragments:

If the number of occurrences of every first order variable and context variable is at most two [Lev96], or if there are at most two context variables, but an arbitrary number of first order variables [SSS99], or if the context unification problems are stratified [SS99b]. Satisfiability in a logical theory of context unification is undecidable [NPR97,Vor98]. A decidable restriction of second order unification similar in spirit to context unification is bounded second order unification [SS99a], where second order variables represent terms with a number of holes that is bounded by some preselected number.

Applications of context unification are for example in computational linguistics [NPR97] and of (stratified) context unification in equational unification [SS98]. Recently it was noticed that satisfiability of one-step rewrite constraints and stratified context unification can be interreduced [NTT99].

This paper presents an algorithm for stratified context unification that improves the run time and space usage of the decision algorithm given in [SS99b]. Based on the methods in [SS99b] a proof of an estimation of the complexity of SCUA is given:

> **Theorem**: Stratified context unification can be performed in time polynomial in the size of the input and the depth of SO-cycles.

It follows from [NTT99] that the algorithm can be traslated and hence the complexity estimation holds also for satisfiability of one-step rewrite constraints.

The construction of the algorithm SCUA and the upper bounds have consequences for implementations. It demonstrates that the exploitation of sharing and the compression of iterated contexts is useful. Unfortunately, we were not able to give an upper bound on the depth of SO-cycles. On the other hand, we have found no example that has SO-cycles during the transformation algorithm of more than linear depth. This supports the conjecture that the depth of SO-cycles is small (perhaps polynomial).

2 Preliminaries

Let Σ be a signature, where we assume that the signature contains at least one non-constant function symbol, in particular we allow also that the signature may be infinite or monadic. With $ar(f)$ we denote the arity of the function symbol f. Let $\mathcal{V}_1$ be the set of first order variables $x, y, z, \ldots$, $\mathcal{V}_2$ be the set of context variables $X, Y, Z, \ldots$, and $\mathcal{V} := \mathcal{V}_1 \cup \mathcal{V}_2$. Terms are formed like first order terms, where context variables are unary and may occur in the position of function symbols. First order terms are terms without occurrences of holes and context variables. If we mean a first order context variable, we write $\mathcal{X}$. Contexts are first order terms with a single occurrence of the special constant $\cdot$, the *hole*. We denote contexts as $C[\cdot]$. The path from the root to the hole of C is called *main path*, denoted $mp(C)$; the length is called *main depth* and denoted as $mdt(C)$. With Id we denote the empty (or trivial) context. A *prefix* of a context C is a context C_1, such that $C_1 C_2 = C$ for some context C_2.

Substitutions σ replace first order variables by first order terms and context variables by contexts. We also use multi-contexts $C[\cdot_1, \ldots, \cdot_n]$, which are first order terms with occurrences of the holes $\cdot_1, \ldots, \cdot_n$, where every hole occurs exactly once. As is standard, positions in terms and contexts are words of positive integers, and $t|_p$ denotes the subterm (subcontext) of t at position p.

In the following we sometimes use the notation $C[\cdot]^n$, where $C[\cdot]$ is a context and n is an integer. This is defined as $C[\cdot]^1 := C[\cdot], C[\cdot]^{n+1} := C[C[\cdot]^n]$. If we use this notation in a term, it is meant as a meta-notation of a term, not as explicit syntax. For integers we use $i \ \mathtt{mod*}\ n$, which is the unique number $j \in [1..n]$ with $i \equiv j (mod\ n)$.
An equation system is a set of equations $s \doteq t$, also called unification problem.

A ground substitution σ has *exponent of periodicity* n ([Mak77,SSS98]), iff i) for every $\mathcal{X}$, if there are ground contexts A, B, C with B nontrivial, such that $\sigma(\mathcal{X}) = AB^mC$, then $m \leq n$; ii) there is some $\mathcal{X}$, such that $\sigma(\mathcal{X}) = AB^nC$, for appropriate ground contexts A, B, C where B is nontrivial.

The following lemma is a generalization of [KP96].

Lemma 2.1. *([SSS98]) There is a constant c, such that for every unifiable context unification problem Γ its exponent of periodicity is at most 2^{c*d}, where d is the size of Γ.*

Definition 2.2. *We define* SO-prefixes *as words in $\mathcal{V}_2^*$. An SO-prefix of a position p in a term t is the word consisting of the context variables in head positions that are met going from the root to the position p.*

In an equation system Γ, an SO-prefix *of a variable or context variable $\mathcal{X}$ is a word in $\mathcal{V}_2^*$, that either belongs to an equation $s \doteq t$, and a position of $\mathcal{X}$ in t or s; or it belongs to a path $x_1 \doteq t_1, \ldots, x_n \doteq t_n$ and is constructed as $w := w_1 \ldots w_n$, where w_i is the SO-prefix of some position of x_{i+1} in t_i for $1 \leq i < n$ and w_n is the SO-prefix of some position of $\mathcal{X}$ in t_n. An SO-prefix is* maximal*, if it either belongs to an occurrence in $s \doteq t$, and s, t are not variables, or to a path $x_1 \doteq t_1, \ldots, x_n \doteq t_n$ and every occurrence of x_1 has an empty SO-prefix.*

Let the following hold:

1. *for every variable and context variable $\mathcal{X}$, there exists a finite unique maximal SO-prefix $p(\mathcal{X})$, and*
2. *for every equation $C[\mathcal{X}] \doteq D[\mathcal{Y}] \in \Gamma$ the following holds: let the SO-prefixes of $\mathcal{X}, \mathcal{Y}$ in the terms $C[\mathcal{X}], C[\mathcal{Y}]$ be $p_{\mathcal{X}}, p_{\mathcal{Y}}$. Then $p_{\mathcal{X}} = p_{\mathcal{Y}}$ implies that the maximal SO-prefixes in Γ of $\mathcal{X}$, and $\mathcal{Y}$ are equal.*

Then Γ is called stratified.

This definition is consistent with the definitions in [SS94,SS99b,Lev96], but it is adapted to systems of equations that permit equations like $x = y, X(x) = z$. i.e., that variables may have different SO-prefix in the terms that occur in equations. It is consistent with this definition to consider equations as labeled with an SO-prefix, and then use the label of the equation and the SO-prefix of positions in terms to compute the SO-prefix in Γ.

For example, $X(x) \doteq x$ is not permitted, since there is no finite SO-prefix of x; $X(x) \doteq Y(x)$ is not stratified, but $X(x) \doteq Y(y), x \doteq f(z)$ is stratified.

3 The algorithm SCUA

The initial input is a set of equations. The intermediate data structure is more involved: Let a *skeleton context* B be a context of the form $B_1 \dots B_m$, where B_i is of the form $f(x_1, \dots, x_{j_i-1}, \cdot, x_{j_i+1}, \dots, x_n)$. With $|B|$ we denote the main depth of B. Let a *power expression* be $pow(B, n)$, where $n \geq 0$, and B is a skeleton context. Let the *terms* be $x \mid f(t_1, \dots, t_n) \mid X(t) \mid P(t)$ where x is a variable, f a function symbol of arity n, X a context variable, t, t_i are terms, and P is a power expression. $pow(B, n)(x)$ is a syntactically compressed form of $B^{n_1}C(x)$, where $n = n_1 * |B| + n_2$ with $n_1 \geq 0, 0 \leq n_2 < |B|$ and C is a prefix of B with $mdt(C) = n_2$.

Let $head(\cdot)$ be defined as: $head(x) := x$, $head(f(\dots)) := f$, $head(Y(y)) := Y$, $head(pow(f(\dots)\dots B_m, n)(s)) := f$ if $n \geq 1$.
$shift(B_1 \dots B_n, m)$ is defined as: $shift(B_1 \dots B_n, 0) := B_1 \dots B_n$ and $shift(B_1 \dots B_n, m) := shift(B_2 \dots B_n B_1, m-1)$.
Let $expand(pow(B_1 \dots B_m, n))$ be defined as follows:
$expand(pow(B_1 \dots B_m, 0)) := Id$, and
$expand(pow(B_1 \dots B_m, n)) := B_1(expand(pow(B_2 \dots B_m B_1, n-1)))$.
This is also used for terms: $expand(pow(B, n)(x)) := expand(pow(B, n))(x)$.

If we say a variable occurs in $pow(B, n)$, then we mean the occurrences in $expand(pow(B, n))$.

A stratified context unification problem (SCUP) is a stratified system of context equations Γ, where equations are denoted as $s \doteq t$ plus a set of disequations of the form $X \neq Id$. A substitution σ that maps first order variables to ground terms and context variables to ground contexts is a *unifier* (or a *solution*) of the SCUP Γ iff after applying σ and *expand*, the left and right hand sides of equations are syntactically equal, and for $(X \neq Id) \in \Gamma$: $\sigma(X) \neq Id$.

The algorithm SCUA has an initial input Γ_I. Let D_I be the size of Γ_I, let E_I be the upper bound on the exponent of periodicity given by the bound in [SSS98] for Γ_I, let $D_A := max\{2, ar(f)\}$, where f are the function symbols occurring in the initial input, and let $\#(CV)$ be the number of context variables in Γ_I.

The main technical advantages of SCUA over the algorithms in [SS99b] is the use of sharing by flattening equations, and the compressed representation of iterated contexts by power expressions. This forces to adapt the algorithm and to use new rules that operate on the new syntax.

3.1 Flattening

Initially, and after a replacement of context variables, a flattening may be required:

Definition 3.1. *Rule (Flatten)*

– $\dfrac{\{s \doteq t\} \cup \Gamma}{\{s \doteq x, x \doteq t\} \cup \Gamma}$ *if neither s nor t is a variable.*

– $\dfrac{\{f(s_1, \ldots, s_n) \doteq t\} \cup \Gamma}{\{f(x_1, \ldots, x_n) \doteq t, x_1 \doteq s_1, \ldots x_n \doteq s_n\} \cup \Gamma}$ *if some s_i is not a variable.*

– $\dfrac{\{X(s) \doteq t\} \cup \Gamma}{\{X(x) \doteq t, x \doteq s\} \cup \Gamma}$ *if s is not a variable.*

– $\dfrac{\{pow(B,n)(s) \doteq t\} \cup \Gamma}{\{pow(B,n)(x) \doteq t, x \doteq s\} \cup \Gamma}$ *if s is not a variable.*

Here the introduced variables are alway fresh ones.

If the flattening rules are not applicable, then Γ is called *flattened*. In this case, only terms of the form $x \mid f(x_1, \ldots, x_n) \mid X(x) \mid pow(B,n)(x)$ where x, x_i are variables, are permitted.

Definition 3.2. *Rule (Normalizing Power Expressions)*
A power expression $pow(B,n)(x)$ is replaced by $pow(C_1 \ldots C_k, n * m)(x)$, if $B = \underbrace{C_1 \ldots C_k C_1 \ldots C_k \ldots C_1 \ldots C_k}_{m \text{ times}}$

In the following we assume that SCUPs are flattened and that power expressions are normalized.

3.2 Decomposition Rules

Definition 3.3. (decomposition rules)

1. *(variable replacement):* $\dfrac{\{x \doteq y\} \cup \Gamma}{\Gamma[y/x]}$.
2. *(decomposition)* $\dfrac{\{x \doteq f(x_1, \ldots, x_n), x \doteq f(y_1, \ldots, y_n)\} \cup \Gamma}{\{x \doteq f(x_1, \ldots, x_n), x_1 \doteq y_1, \ldots, x_n \doteq y_n\} \cup \Gamma}$
3. *(clash)* $\dfrac{\{x \doteq f(x_1, \ldots, x_n), x \doteq g(y_1, \ldots, y_m)\} \cup \Gamma}{Fail}$, *if $f \neq g$.*
4. *(occurs-check) Fail, if there is a chain of equations $x_1 \doteq t_1, \ldots, x_n \doteq t_n$, and $x_{i+1 \texttt{ mod* } n}$ occurs in t_i, and at least one t_i has a function symbol as head.*
5. *(remove-fo)* $\dfrac{\{x \doteq x\} \cup \Gamma}{\Gamma}$.
6. *Remove disequations $X \neq Id$, if X does not occur in the rest of Γ.*
7. $\dfrac{\{x \doteq t\} \cup \Gamma}{\Gamma}$ *if x does not occur in t nor Γ.*
8. *(Remove-cv) For a context variable X, if $(X \neq Id) \notin \Gamma$, then select one of the following possibilities:*
   (a) Add the disequation $X \neq Id$.
   (b) Replace X by Id everywhere in Γ.

For every equation system Γ, the decomposition rules are performed with high priority. If no decomposition rule is applicable, then we say Γ is *decomposed*. If for every context variable in Γ, there is a disequation $X \neq Id$, then we say it is *disequation-complete*.

3.3 SO-cycles and SO-clusters

Definition 3.4. *A set of equations $s_1 \doteq t_1, \ldots, s_n \doteq t_n$ is called an* SO-cycle, *if the following holds: s_i is of the form x_i (or $X_i(y_i)$), and x_i (or X_i) occurs in $t_{i-1 \texttt{ mod* } n}$, but not below a context variable, and at least one such occurrence is not at the top, and there is no context variable that occurs twice in the SO-cycle. The* length *of an SO-cycle is the number of context variables at the top positions. An SO-cycle is called* ambiguous, *iff one of the following holds:*

- *There is an i, such that s_i is a first order variable, and has more than one occurrence in $t_{i-1 \texttt{ mod* } n}$.*
- *For some i, the term s_i is a first order variable, and the sequence of equations to the next context variable is $s_i \doteq t_i, \ldots, s_j \doteq t_j$, t_j contains the context variable X, and the sequence can be replaced by a different subsequence starting with $s_i \doteq s'_i$, ending with $s'_k \doteq t'_k$, the only term in the new subsequence that contains a context variable is t'_k, and this context variable is X. Let the term $\bar{s}_i$ result from instantiating the variables using $s_i \doteq t_i, \ldots, s_j \doteq t_j$, and $\bar{s'}_i$ resulting from instantiating the variables using $s_i \doteq t'_i, \ldots, s'_k \doteq t'_k$, and let the positions of the X in the terms $\bar{s}_i$ and $\bar{s'}_i$ be different.*

The SO-cycle that results from such a replacement of one subsequence by athe other is called an ambiguous variant.

If the SO-cycle is not ambiguous, then it is called path-unique. *The* depth *of an SO-cycle is the sum of the depths of X_i (or x_i) in $t_{i-1 \texttt{ mod* } n}$. The* amb-depth *of an ambiguous SO-cycle is the maximum of the two smallest depths of two ambiguous variants of the SO-cycle.*

Given the initial input Γ_I, let D_Z be the maximum of the depth of all SO-cycles that remains a maximum for all transformations.

Definition 3.5. *Let Γ be an SCUP. Let $\sim$ be the equivalence relation on $\mathcal{V}$ generated by $X_1 \sim X_2, X_1 \sim x_1$, or $x_1 \sim y_1$ if there is an equation $X_1(z_1) \doteq X_2(z_2), X_1(z_3) \doteq x_1$, or $x_1 \doteq y_1$, respectively in Γ.*

Let $\succ$ be the relation on $\mathcal{V}$ generated by $x \succ y$ if x, y have empty SO-prefix and there is an equation $x \doteq t \in \Gamma$, $t \not\equiv x$ and x occurs in t. I.e. $t \equiv f(\ldots, y, \ldots)$, or t is of the form $pow(B, n)(z)$. Let $\gtrsim$ be the quasi-ordering generated by the transitive and reflexive closure of $\succ \cup \sim$. If there are variables x, y with $x \succ y$ and $y \gtrsim x$, then we say $\gtrsim$ (or Γ) has cycles, *otherwise, it is called* cycle-free.

If $\gtrsim$ is cycle-free, then an equivalence class K of $\sim$ is called an SO-cluster. *An SO-cluster K is called a* top-SO-cluster, *iff the variables in K have empty SO-prefix and are maximal w.r.t. $\gtrsim$. The set of equations in Γ, where the variables from an SO-cluster K occur at top-level, is denoted as $EQ(K)$. Let K_C be the subset of context variables in K. A top-SO-cluster K is called flat, iff it is also $\gtrsim$-minimal.*

Note that a non-flat top-SO-cluster may consist of first order variables only. In a decomposed SCUP, a top-SO-cluster always contains a context variable.

3.4 Ambiguous SO-cycles

Definition 3.6. *(Eliminate ambiguous SO-cycles)*
If there is an ambiguous SO-cycle with involved SO-variables $X_1, \dots, X_n$, and amb-depth d, then select some $k \in \{1, \dots, n\}$ and a skeleton context B of depth $\leq (3 * \#(CV) + 1) * d$ and replace X_k by B.

This rule allows to get rid of all ambiguous SO-cycles and also eliminates a context variable from Γ after every application.

3.5 Expanding Small Powers

If in an expression $pow(B, n)$, the number n does not exceed $3 * D_Z$, then we will expand it with high priority:

Definition 3.7. *(expand-small-powers)*

$$\frac{\{x \doteq pow(B, n)(y)\} \cup \Gamma}{\{x \doteq t\} \cup \Gamma} \text{ if } n < 3 * D_Z, \text{ where } t \equiv expand(pow(B, n)(y))$$

3.6 Path-unique SO-cycles

By decomposition and expansion and By decomposition and expansion of small powers, we can assume, we can assume that a path-unique SO-cycle has no terms of the form $pow(B, n)(x)$ and no equations of the form $x \doteq y$.

A path-unique SO-cycle is usually written as $s_1 \doteq C_1[t_1], \dots, s_n \doteq C_n[t_n]$ where the top-variable of s_i occurs in $t_{i-1 \texttt{ mod* } n}$. A *plateau* P is every sequence of equations $s_j \doteq C_j[t_j], \dots, s_{j'} \doteq C_{j'}[t_{j'}]$, such that for $i = j, j+1, \dots, j' - 1(\texttt{ mod*} \; n)$, $C_i = Id$, but $C_{j'} \neq Id$, and $C_{j-1 \texttt{ mod* } n} \neq Id$.

The intuition of this rule is that an instantiation along the cycle is guessed, where the number of rounds is the exponent of periodicity. One distinction is that either some context variable is exhausted, or there is some deviation of a main path from the cycle.

Definition 3.8. *Given a path-unique SO-cycle with a minimal number of involved context variables, $s_1 \doteq C_1[t_1], \dots, s_n \doteq C_n[t_n]$. First select a plateau in the SO-cycle . For simplicity we assume that the SO-cycle is then renumbered such that it starts with index 1. Now we can use a different method of indexing: Let the SO-cycle be $s_1 \doteq t_1, \dots s_{j_1} \doteq C_{j_1}[t_{j_1}], \dots s_{j_n} \doteq C_{j_n}[t_{j_n}]$, where C_{j_i} is the last context in the i^{th} plateau. Denote the top context variables in plateau i as $X_{i,j}$.*

Select an integer $D_Z \leq e \leq E_I * D_Z$, and then select one of the following possibilities:

1. *Let B be a skeleton context with $|B| \leq D_Z$ and instantiate some $X_{i,j}$ with B.*
2. *– for $i > 1$: Replace $X_{i,h}$ by $pow(C_{j_i} C_{j_{i+1}} \dots C_{j_{i+n \texttt{ mod* } n}}, e - i + 1)(X'_{i,h})$*

– *for* $i = 1$*: For every* h *replace* $X_{1,h}$ *either by* $pow(C_{j_1}C_{j_2}\dots C_{j_n}, e)(X'_{1,h})$ *or by* $pow(C_{j_1}C_{j_2}\dots C_{j_n}, e)(Id)$*, where the last case should be selected at least once.*

3. *Fail, if the SO-cycle has length 1.*
   – *for* $i > 1$*: Replace* $X_{i,h}$ *by* $pow(C_{j_i}C_{j_{i+1}}\dots C_{j_{i+n \bmod^* n}}, e - i + 1)(X'_{i,h})$
   – *for* $i = 1$*: Let* $C_{j_1} \equiv f(x_{j_1,1}, \dots, \underbrace{C'_{j_1}}_{k_0}, \dots, x_{j_1,m})$*. For every* h *select an index* $1 \leq k_h \leq m$ *and replace* $X_{1,h} \to pow(C_{j_1}C_{j_2}\dots C_{j_n}, e) f(y_{h,1}, \dots, \underbrace{X'_{1,h}}_{k_h}, \dots, y_{h,m})$*. There should be at least one* k_h *that is different from* k_0*. After the replacement, a flattening is performed.*

We assume that the new introduced variables are fresh ones.

3.7 Flat SO-Clusters

The assumption is that Γ is flattened, decomposed, disequation-complete and there are no SO-cycles. Note that every flat top-SO-cluster contains a context variable.

Definition 3.9. *Let there be a flat top SO-cluster* K *with* $K_C = \{X_1, \dots, X_h\}$ *of minimal size in* Γ *with set of equations* $EQ(K)$*. Fail, if* $h = 1$ *or the maximal arity of function symbols in* Σ *is* ≤ 1*.*
Let F *be a new function symbol with* $2 \leq ar(F) \leq |K|$*.*
For every context variable $X_i \in K$*, select an index* $1 \leq k_i \leq ar(F)$ *and replace* $X_i(\cdot)$ *by* $F(x_{i,1}, \dots, \underbrace{X'_i(\cdot)}_{k_i}, \dots, x_{i,ar(F)})$*, where* $x_{i,j}, X'_i$ *are new. There should be different indices* k_i*.*

Then decompose the equations that result from instantiating and flattening the equations in $EQ(K)$*.*

Note that the symbol F disappears after decomposition, and that the only possible exit from the iterated application is to remove a context variable.

3.8 Non-Flat SO-clusters

This subsection treats the harder case of non-flat SO-clusters.
The assumption is that Γ is flattened, decomposed, disequation-complete, and there are no flat top-SO-clusters nor SO-cycles.

Definition 3.10. *Let* $BDEC(B_1, B_2)$ *be the following algorithm applied to two skeleton contexts* B_1, B_2*: It signals either Fail, or returns a set of equations between variables:*
$BDEC(B_1, B_2)$ *returns Fail if* $|B_1| \neq |B_2|$*.* $BDEC(Id, Id) = \emptyset$*.*
$BDEC(f(x_1, \dots, x_{j_1-1}, \cdot, x_{j_1+1}, \dots, x_n)B'_1, g(y_1, \dots, y_{j_2-1}, \cdot, y_{j_2+1}, \dots, y_m)B'_2)$ *results in: if* $j_1 \neq j_2$ *or* $f \neq g$*, then Fail, else* $\{x_i \doteq y_i \mid i \neq j_1\} \cup BDEC(B'_1, B'_2)$*.*

Note that the output of $BDEC$ is always a set of equations between variables, and thus the addition of the equations always makes the SCUP smaller after application of decomposition rules.

Definition 3.11. *(power-decomp). This rule is only used after an application of the rule (non-flat-SO-Cluster), and only for the equations in $EQ(K)$ of the non-flat top-SO-cluster after instantiations.*

There are four cases for a decomposition of terms starting with a power expression. We assume that after each rule, a flattening is performed if necessary.

1. $$\frac{\{x \doteq pow(B_1, n_1)(x_1), x \doteq pow(B_2, n_2)(x_2)\} \cup \Gamma}{\{x \doteq pow(B_1, n_1)(x_1), x \doteq pow(B_2, n_2)(x_2)\} \cup BDEC(C_1, C_2) \cup \Gamma}$$
   If $n_i \geq 3 * D_Z, i = 1, 2$ and $B_1 \not\equiv B_2$, where $C_i \equiv expand(pow(B_i, 2 * D_Z)), i = 1, 2$.
2. $$\frac{\{x \doteq pow(B_1, n_1)(z), x \doteq f(x_1, \ldots, x_n)\} \cup \Gamma}{\{x_i \doteq y_i \mid i \neq k\} \cup \{pow(B_1', n_1 - 1)(z) \doteq x_k\} \cup \{x \doteq f(x_1, \ldots, x_n)\} \cup \Gamma}$$
   where $B_2 := shift(B_1, 1)$, and $f(y_1, \ldots, y_{k-1}, \cdot, y_{k+1}, \ldots, y_n)$ is the first atomic context of B_1.
3. $$\frac{\{x \doteq pow(B_1, n_1)(z), x \doteq f(x_1, \ldots, x_n)\} \cup \Gamma}{Fail}$$
   if $f \neq head(pow(B_1, n_1)(z))$.
4. $$\frac{\{x \doteq pow(B, n_1)(x_1), x \doteq pow(B, n_2)(x_2)\} \cup \Gamma}{\{x \doteq pow(B, n_1)(x_1), x_1 \doteq pow(B, n_2 - n_1)(x_2)\} \cup \Gamma} \quad \text{if } n_2 \geq n_1.$$

In order to give some intuition of the cases in the rule for non-flat SO-cluster, the distinction for the context variables $\{X_1, \ldots, X_h\}$ in an SO-cluster is made on the basis of the relative position and depth of the holes of the instances of X_i. In the case where every non-variable equation in $EQ(K)$ has a power expression at the top, there are the cases i) that some $\sigma(X_i)$ has a small main depth, ii) that some $\sigma(X_i)$ is a prefix of all others and also covered by the power expressions, or iii) that the common prefix of the $\sigma(X_i)$ is long, such that we can decompose, or iv) that the common prefix is long enough, but the bases are already equal, but there is a forking of the instances of X_i.

Definition 3.12. *Rule (non-flat-SO-Cluster)*
This rule is only applicable if there are no SO-cycles, no flat top-SO-clusters, but a non-flat top-SO-cluster.
Let $K = \{X_1, \ldots, X_h\}$ be the context variables in a non-flat top-SO-cluster, where h is minimal.
Then select one of the following two possibilities:

1. *If there is an equation $s \doteq f(t_1, \ldots, t_n) \in EQ(K)$. Then:*
   - *If there is a further equation $x \doteq t \in EQ(K)$, and t is not of the form $Y(y)$ and $head(t) = g \neq f$, then Fail.*
   - *For every $i = 1, \ldots, h$, select an index k_i and replace every $X_i \in K$ by $f(x_{i,1}, \ldots, \underbrace{X_i'(\cdot)}_{k_i}, \ldots, x_{i,n})$ where $x_{i,j}, X_i'$ are new variables.*

2. *If there is no equation $s \doteq f(t_1,\ldots,t_n) \in EQ(K)$, but there are at least two equations $x_1 \doteq pow(B_1,n_1)(y_1), x_2 \doteq pow(B_2,n_2)(y_2)$ in $EQ(K)$ for different B_1, B_2. Let $\bar{n}$ be the minimum of the numbers n_i for all equations $x \doteq pow(B_i,n_i)(x_i)$ in $EQ(K)$. Note that we can assume that $\bar{n} \geq 3 * D_Z$. Then select one of the following:*
   - *Select some m with $1 \leq m \leq 3 * D_Z$ and replace every X_i by $pow(B_1,m)(X_i')$ or by $pow(B_1,m)(Id)$ where the last case should be selected at least once.*
   - *Select a number $0 \leq n \leq 2 * D_Z - 1$. Let $C_1 = shift(B_1,n)$, $f := head(C_1)$ and the hole of C_1 in direction k_0. Fail if $ar(f) \leq 1$. For every $1 \leq i \leq h$ select an index $1 \leq k_i \leq ar(f)$. Replace every X_i by $pow(B_1,n)(f(x_{i,1},\ldots,\underbrace{X_i'}_{k_i},\ldots,x_{i,ar(f)}))$, where for at least one index j: $k_0 \neq k_j$. For a fixed equation $x_0 \doteq pow(B_0,n_0)(y_0) \in EQ(K)$ and every other equation $x_1 \doteq pow(B_1,n_1)(y_1) \in EQ(K)$ perform $BDEC(pow(B_0,n), pow(B_1,n))$ and add the resulting equations to Γ.*
   - *For a fixed equation $x_0 \doteq pow(B_0,n_0)(y_0) \in EQ(K)$ and every other equation $x_1 \doteq pow(B_1,n_1)(y_1) \in EQ(K)$ perform $BDEC(pow(B_0, 2 * D_Z), pow(B_1, 2 * D_Z))$ and add the resulting equations to Γ.*
3. *If there is no equation $s \doteq f(t_1,\ldots,t_n) \in EQ(K)$, and for every pair of equations $x_1 \doteq pow(B_1,n_1)(y_1), x_2 \doteq pow(B_2,n_2)(y_2)$ it is $B_1 \equiv B_2$. Let $B := B_1$*
   Let $\bar{n}$ be the minimum of the numbers n_i for all equations $x \doteq pow(B,n_i)(x_i)$ in $EQ(K)$. Note that we can assume that $\bar{n} \geq 3 * D_Z$. Then select a number $1 \leq n \leq \bar{n}$ and select one of the following:
   - *Replace every X_i by $pow(B,n)(X_i')$ or by $pow(B,n)(Id)$, where the last case should be selected at least once.*
   - *Replace every X_i by $pow(B,\bar{n})(X_i')$.*
   - *Fail if $n = \bar{n}$. Let $C_1 := shift(B_1,n)$, $head(C_1) = f$ and the hole of C_1 in direction k_0. Fail if $ar(f) \leq 1$. For every $1 \leq i \leq h$ select an index $1 \leq k_i \leq ar(f)$. Replace every X_i by $pow(B,n)(f(x_{i,1},\ldots,\underbrace{X_i'}_{k_i},\ldots,x_{i,ar(f)}))$, where at least one index k_i must be different from k_0.*

After every replacement, flatten, decompose and power-decompose the resulting equations.

Note that the number D_Z is used as a known upper bound for the length of bases after normalization. If the n in the power expressions $pow(B,n)$ is larger than $2D_Z$, then we are sure that there are at least 2 periods.

3.9 How the rules work together: SCUA

The algorithm SCUA has as input a stratified context unification problem Γ consisting only of equations between terms. The following steps are performed.

First, the system is flattened in order to exploit sharing.
Then the following rules are applied until the system is empty, or a Fail is signalled, where between rule applications: decomposition rules, power-decomposition rules, flattening and normalization of power expressions is done with high priority.

1. If there is an ambiguous SO-cycle, then apply the instantiation in 3.4.
2. If there is no ambiguous SO-cycle, but a path-unique SO-cycle, then apply the rule in 3.6 to Γ.
3. If there are no SO-cycles, and a flat top-SO-cluster, then apply the rule 3.9 for flat top-SO clusters to a minimal one.
4. If there are no SO-cycles, no flat top-SO-clusters, then apply the rule 3.12 to a non-flat top-SO-cluster.

3.10 Upper Bounds on the Depth of Instances of Ambiguous SO-Cycles

Theorem 3.13. *Let Γ be a stratified context unification problem. Let Γ' be reached by transformations from Γ, such that Γ' is solvable by σ. Let there be an ambiguous SO-cycle that can be represented as the unflattened sequence of equations: $X_1(\cdot) \doteq C_1[X_2(\cdot)], \ldots, X_{m-1}(\cdot) \doteq C_{m-1}[X_m(\cdot)], X_m(\cdot) \doteq C_m[X_1(\cdot), X_1(\cdot)]$. Let d be the sum of $mdt(C_i)$ for $i = 1, \ldots, m$ plus the maximum of the depths of the two holes in $C_1[\cdot, \cdot]$.*

Then there is some context variable X_i, such that $\sigma(X_i)$ has main depth less than $(3 * m + 1) * d$,

The proof is given in the appendix.

4 Correctness and Complexity of SCUA

Given the methods in [SS99b], it is a straightforward to adapt the soundness and completeness proofs though it is tedious. The following holds for SCUA: the stratifiedness property is not destroyed by the rules. Furthermore, the number of occurrences of context variables is not increased.

A fact concerning periods in words is used for the completeness of decomposing powers and also for the completeness of solving no-flat top-SO-clusters:

Lemma 4.1. *Let w_i, w_i' be words over an alphabet. If $w_1^n w_1' = w_2^m w_2'$, and $n, m \geq 2$, and w_i' is a prefix of w_i, then $w_i = w_3^{n_i}, i = 1, 2$ for some n_i and w_3.*

Lemma 4.2. *Let $\sigma(pow(B_1, n_1)(x_1)) = \sigma(pow(B_2, n_2)(x_2))$ and $n_i \geq 3 * |B_j|$ for $i, j = 1, 2$. Then $\sigma(pow(B_1, n_1))$ is a prefix of $\sigma(pow(B_2, n_2))$ or vice versa.*

Proof. Using similar methods as in [SS99b] it is easy to see that if the paths deviate at a position of depth $\leq max(2|B_1|, 2|B_2|)$, then there is an occur-check failure. Then the fact on overlapping periods shows that the two power expressions are both powers of the same skeleton context, hence they have the same main path, and cannot deviate at larger depths. □

Another easy to establish fact is that a solvable context unification problem has a solution, where the maximal arity of function symbols is $D_A := max(2, ar(f))$, where f is a symbol in Γ_I.

Now we estimate the complexity of the algorithm:

Lemma 4.3. *SCUA can be performed in time polynomial in the size and the maximal depths of SO-cycles.*

Proof. We only argue on the complexity of the critical operations, and assume that the arity of function symbols is $O(n)$. This covers the case of an infinite signature.

First we estimate the number of applications and the size increase by the essential rules:

- The maximal number of applications of eliminating ambiguous SO-cycles is $\#CV$.
- The maximal number of applications of eliminating SO-cycles is $\#CV^2$: The length of an SO-cycle is at most $\#CV$, and in every application, either the SO-cycle gets shorter, or a context variable is removed.

Now we explore the space increase:
The decomposition rules do not increase space usage. The same holds for the rules for top-SO-clusters, if the subsequent high priority rule applications are taken into account. Every SO-cycle-rule application may add power expressions, but at most $\#CV$. Every power expression may generate $3 * D_Z$ occurrences of function symbols. Since the number of applications is $O(n^2)$, the size, and hence the number of function symbols is at most $D_I + \#CV^2 * 3 * D_A * D_Z$, which is of order $O(D_I^3 * D_Z)$.

The number of introduced power expressions is polynomial in D_I, since only solving cycles can introduce new ones. The rules for flat clusters may copy some power expressions, however these are removed again after decomposition, power-decomposition, and normalization. For simplicity, the resource requirement for expansions of power expressions are counted as contribution of the cycle-elimination rules.
Third we estimate the number of applications of the SO-cluster rules:

- The maximal number of applications of eliminating a flat top-SO-cluster is $\#CV^2$: Every application strictly reduces the number of context variables in a flat top-SO-cluster, or removes a context variable.
- The maximal number of applications of eliminating a non-flat top-SO-cluster is harder to establish. The tuple (number of context variables, number of context variables in the top-SO-cluster, number of variables not in any top-SO-cluster, number of occurrences of function symbols on top level) strictly

decreases lexicographically after every application of the rule for eliminating non-flat top-SO-clusters (including subsequent high priority rules). The last parameter is of order $O(D_I^3 * D_Z)$, the same for the number of variables. The worst case corresponds to multiplication, i.e. the number of applications is smaller than $p(D_I) * D_Z^2$, where p is some polynomial.

It remains to estimate the contribution of every single rule application together with the following flattening, decomposition etc. It is easy to see that flattening, decomposition, and normalization require time polynomial in the size.

- Instantiating an ambiguous SO-cycle: Is of order $p(D_I) * D_Z$, where p is some polynomial.
- Solving path-unique SO-cycles is of order $p(D_I) * D_Z$, where p is some polynomial.
- SO-cluster elimination: Eliminating a flat SO-cluster requires comparably less resources.
  If it is a non-flat SO-cluster, then the overall usage of time is $p(D_I) * D_Z^2$.

Corollary 4.4. *The time required to perform a non-deterministic run of SCUA is polynomial in the initial size D_I and the maximal depth D_Z of an SO-cycle.*

5 Future Work

The complexity of stratified context unification remains an issue. Among others, the paper shows that an upper bound on the depth of SO-cycles implies a complexity estimate of stratified context unification. Proving better complexity estimations of stratified context unification or proving an upper bound for the depth of SO-cycles is left for future work.

The decision algorithm for bounded second order unification [SS99a] is very similar to the decision algorithm for stratified context unification. The methods developed in this paper may be used in an estimation of its complexity.

Perhaps the tools in this paper help in computing an upper bound for the complexity of D-unification [SS98], however, this would require a careful inspection of the algorithm since context unification is not used as a module.

References

[Com93] Hubert Comon. Completion of rewrite systems with membership constraints, part I: Deduction rules and part II: Constraint solving. Technical report, CNRS and LRI, Université de Paris Sud, 1993. to appear in JSC.

[Far91] W.A. Farmer. Simple second-order languages for which unification is undecidable. *J. Theoretical Computer Science*, 87:173–214, 1991.

[Gol81] W.D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

[Hue75] Gerard Huet. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[KP96] Antoni Kościelski and Leszek Pacholski. Complexity of Makanin's algorithms. *Journal of the Association for Computing Machinery*, 43:670–684, 1996.

[Lev96] Jordi Levy. Linear second order unification. In *Proceedings of the 7th International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 332–346, 1996.

[LV99] Jordi Levy and Margus Veanes. On the undecidability of second-order unification, 1999. to appear in Information and Computation.

[Mak77] G.S. Makanin. The problem of solvability of equations in a free semigroup. *Math. USSR Sbornik*, 32(2):129–198, 1977.

[NPR97] Joachim Niehren, Manfred Pinkal, and P. Ruhrberg. On equality up-to constraints over finite trees, context unification, and one-step rewriting. In *Proceedings of the International Conference on Automated Deduction*, volume 1249 of *Lecture Notes in Computer Science*, pages 34–48, 1997.

[NTT99] Joachim Niehren, Ralf Treinen, and Sophie Tison. On rewrite constraints and context unification, 1999. to appear in Information Processing Letters.

[Pie73] T. Pietrzykowski. A complete mechanization of second-order type theory. *J. ACM*, 20:333–364, 1973.

[Pla99a] W. Plandowski. Satisfiability of word equations with constants is in NEXPTIME. In T. Leighton, editor, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC'99)*, Atlanta, Georgia, 1999. ACM Press. To appear.

[Pla99b] W. Plandowski. Satisfiability of word equations with constants is in PSPACE, 1999. To appear.

[Pre95] Christian Prehofer. *Solving Higher-order Equations: From Logic to Programming*. Ph.D. thesis, Technische Universität München, 1995. In German.

[SG89] Wayne Snyder and Jean Gallier. Higher-order unification revisited: Complete sets of transformations. *J. Symbolic Computation*, 8:101–140, 1989.

[SS94] Manfred Schmidt-Schauß. Unification of stratified second-order terms. Internal Report 12/94, Fachbereich Informatik, J.W. Goethe-Universität Frankfurt, Frankfurt, Germany, 1994.

[SS98] Manfred Schmidt-Schauß. A decision algorithm for distributive unification. *Theoretical Computer Science*, 208:111–148, 1998.

[SS99a] Manfred Schmidt-Schauß. Decidability of bounded second order unification. Technical Report Frank-report-11, FB Informatik, J.W. Goethe-Universität Frankfurt am Main, 1999. submitted for publication.

[SS99b] Manfred Schmidt-Schauß. A decision algorithm for stratified context unification. Frank-Report 12, Fachbereich Informatik, J.W. Goethe-Universität Frankfurt, Frankfurt, Germany, 1999. submitted for publication.

[SSS98] Manfred Schmidt-Schauß and Klaus U. Schulz. On the exponent of periodicity of minimal solutions of context equations. In *Proceedings of the 9th Int. Conf. on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 61–75, 1998.

[SSS99] Manfred Schmidt-Schauß and Klaus U. Schulz. Solvability of context equations with two context variables is decidable. In *Proceedings of the International Conference on Automated Deduction*, Lecture Notes in Computer Science, pages 67–81, 1999.

[Vor98] S. Vorobyov. The $\forall\exists^8$-equational theory of context unification is co-recursively enumerable hard, 1998. Talk at CCL'98 workshop.

[Wol93] D.A. Wolfram. *The clausal theories of types*. Number 21 in Cambridge tracts in theoretical computer science. Cambridge University Press, 1993.

Appendix

A Proof of Theorem 3.13

The main purpose of this section is to show that in an ambiguous SO-cycle there exists at least one context variable that has an instance of a main depth $O(n*d)$, where n is the number of context variables in the cycle and d is the amb-depth of the SO-cycle.

In this section we only compute in instantiated terms and contexts: Hence in this section the variables X, Y denote contexts, so we can speak of their main depth and their size.

An SO-cycle is a chain of equalities $X_i(s_i) = r_i$, i=1,...,n, such that X_i is contained in $r_{i-1 \texttt{ mod* } n}$, and such that at least one occurrence in r_i is not at the top. The *length* of the SO-cycle is n, i.e., the number equations. The *depth* of the SO-cycle is the maximal sum of the depths of X_i in $r_{i-1 \texttt{ mod* } n}$.

In the following proof, the argument term t in $X(t)$ does not really matter, so we use the notation $X \sim Y$ as abbreviation of an equality $X(s) = Y(t)$. Analogously, we use $X \sim C[Y]$ for $X(s) \sim C[Y[t]]$ and $X \sim C[Y, Z]$ for $X(s) \sim C[Y[t_1], Y[t_2]]$ where C is a multicontext.

Lemma A.1. *Let the following chain of equations be given,* $X_1 \sim \ldots \sim X_n \sim C[X_1, \ldots, X_1]$ *and let* C *have at least* $n+1$ *holes and let* d *be the maximum of the depths of the holes in* C.

Then for some $i : mdt(X_i) \leq d$.

Proof. We can assume that $mdt(X_i) > d$ for all i.

The proof is by induction on n: If $n = 1$, then the lemma holds. Otherwise consider the cases for X_n.

- X_n has all the holes of C in its side area, then the derived chain is $X_1 \sim \ldots \sim X_{n-1} \sim C'[X_1, \ldots, X_1]$, where the depths of the holes are the same as in C, and we can use induction on n.
- Wlog. $X_n = C[X'_n, X_1(), \ldots, X_1()]$. Then the obtained chain is $X_1 \sim \ldots \sim X_{n-1} \sim C[X'_n, X_1, \ldots, X_1]$, and we can use induction.

□

Lemma A.2. *Let the following equations be given:*

$$X_1 \sim X_2 \sim \ldots \sim X_n \sim C[X_{\varphi(n)}, X_{\varphi(n-1)}, \ldots, X_{\varphi(0)}]$$

where $\varphi : [0..n] \to [1..n]$ *is monotone. Assume the depth of the holes corresponding to* $X_{\varphi(j)}, j = 1, \ldots, n$ *in* C *is at most* $(n - j + h) * d$. *Then for some* i: $mdt(X_i) \leq (n - i + h) * d$

Proof. We can assume that the main depths are larger than given in the lemma. If $n = 1$, then the lemma holds, since there are two occurrences of X_1 and the depth is at most $h * d$.

Now let $n > 1$. We can assume that $mdt(X_i) > (n - i + h) * d$.

Consider the possibilities for X_n.

- If $\varphi(n) = \varphi(n-1) = n$, then there are two occurrences of X_n and $mdt(X_n) \leq h * d$.
- If $\varphi(n) = n > \varphi(n-1)$, then only $X_n = C[X'_n, X_{\varphi(n-1)}(), \ldots, X_{\varphi(0)}()]$ is possible, and the new equation chain is

$$X_1 \sim X_2 \sim \ldots \sim X_{n-1} \sim C[X'_n, X_{\varphi(n-1)}, \ldots, X_{\varphi(0)}]$$

  We can ignore X'_n, since $\varphi(n-1) \leq n - 1$ and then apply induction on n using $n' := n - 1$ and $h' := h + 1$.
- If $\varphi(n) > n$, there are two possibilities for X_n.
  - X_n has all the holes of C in its side-area, then: $X_1 \sim X_2 \sim \ldots \sim X_{n-1} \sim C'[X_{\varphi(n)}, X_{\varphi(n-1)}, \ldots, X_{\varphi(0)}]$, and the holes in C' are at the same depth as in C. We can use induction on n, using $n' := n - 1, h' := h, \varphi'(j) := \varphi(j + 1)$.
  - $X_n = C[\ldots, \underbrace{X'_n}_{j}, \ldots]$: Then the new chain is: $X_1 \sim X_2 \sim \ldots \sim X_{n-1} \sim$

    $C'[X_{\varphi(n)}, \ldots, \underbrace{X'_n}_{j}, \ldots, \ldots, X_{\varphi(0)}]$

    The chain is shortened if the j^{th} hole is ignored, and we can use induction on n using $n' := n-1$ and $h' := h+1$, and $\varphi'(i) = \varphi(i)$ or $\varphi'(i) = \varphi(i+1)$. Then $d(X_{\varphi'(i)}) = d(X_{\varphi(i \vee i+1)}) \leq n - i + h = n' - i + h'$. The result is $mdt(X_i) \leq (n' - i + h') * d = (n - i + h) * d$.

□

Lemma A.3. *Let m, k, n, N be numbers with $m, n \geq 1$, $n + 1 \geq k$, $N = m + n$, and let the following equations be given:*

$$\begin{array}{l} X_1 \sim \ldots \sim X_n \sim D[X_{\varphi(k)}, \ldots, X_{\varphi(1)}, Y_h] \\ Y_1 \sim \ldots \sim Y_m \ \sim C[Y_1, X_1] \end{array}$$

where $\varphi : [1..k] \to [1..n]$ is monotone, the depths of the holes of $X_{\varphi(j)}$ in D is $\leq (k - j + 1) * d$, the depth of Y_h is $\leq k * d$, and the depth of Y_1, X_1 in C is $\leq d$.

Then one of the following holds:

- *for some X_i: $mdt(X_i) \leq (N - i + 1) * d$ or*
- *for some Y_i with $i \geq h$: $mdt(Y_i) \leq (2N - k + 1) * d$ or*
- *for some Y_i with $i < h$: $mdt(Y_i) \leq (2N - k) * d$.*

Proof. If $n + 1 = k$, then we can use Lemma A.2, which shows that for some i: $mdt(X_i) \leq n - i + 1$. Hence in the following we can assume that $n \geq k$.

The proof is by induction on the following parameters: the number m, then $m-h$, then $n-k$.

Base case: $m=1$. Then $Y_1 \sim C[Y_1, X_1]$ implies that Y_1 has X_1 in its side area at least $N-k$ times, where the depths are: $d, \ldots, (N-k)*d$. Using an appropriate φ, Lemma A.2 implies that some $mdt(X_i) \leq (N-i+1)*d$.

Induction:

case $h < m$: We distinguish the cases for Y_m in the equation $Y_m \sim C[Y_1, X_1]$. There are three cases:

- Y_m has the holes of C in its side area, i.e. X_1, Y_1 are in the side area. Then

$$\begin{array}{ll} X_1 \sim \ldots \sim X_n & \sim D[X_{\varphi(k)}, \ldots, X_{\varphi(1)}, Y_h] \\ Y_1 \sim \ldots \sim Y_{m-1} & \sim C'[Y_1, X_1] \end{array}$$

  where the depths in C' are the same. We can use induction on m.

- $Y_m = C[Y'_m, X_1()]$. Then

$$\begin{array}{ll} X_1 \sim \ldots \sim X_n & \sim D[X_{\varphi(k)}, \ldots, X_{\varphi(1)}, Y_h] \\ Y'_m \sim Y_1 \sim \ldots \sim Y_{m-1} & \sim C[Y'_m, X_1] \end{array}$$

  and by induction on the distance of $m-h$, we get the upper bounds for md. We have to check only Y_m: If $mdt(Y'_m) \leq (2N-k)*d$, then $mdt(Y_m) \leq (2N-k+1)*d$.

- $Y_m = C[Y_1(), Y'_m]$. Then

$$\begin{array}{ll} Y'_m \sim X_1 \sim \ldots \sim X_n & \sim D[X_{\varphi(k)}, \ldots, X_{\varphi(1)}, Y_h] \\ Y_1 \sim \ldots \sim Y_{m-1} & \sim C[Y_1, Y'_m] \end{array}$$

  We use induction on m. We have to check only Y_m: If $mdt(Y'_m) \leq N$, then $mdt(Y_m) \leq (2N-k+1)*d$, since $n \geq k$.

case $h = m$:

The equations are of the form:

$$\begin{array}{ll} X_1 \sim \ldots \sim X_n & \sim D[X_{\varphi(k)}, \ldots, X_{\varphi(1)}, Y_m] \\ Y_1 \sim \ldots \sim Y_m & \sim C[Y_1, X_1] \end{array}$$

A case analysis for Y_m using the equations $Y_m \sim C[Y_1, X_1]$ gives three cases

- Y_m has the holes of C in its side area, i.e., X_1, Y_1 are in the side area. Then

$$\begin{array}{ll} X_1 \sim \ldots \sim X_n & \sim D[X_{\varphi(k)}, \ldots, X_{\varphi(1)}, C'[Y_1, X_1]] \\ Y_1 \sim \ldots \sim Y_{m-1} & \sim C'[Y_1, X_1] \end{array}$$

  We use induction on m. Note that $N' = N-1, k' = k+1$, hence $N-k = N'-k'$.

– $Y_m = C[Y'_m, X_1()]$. Then

$$\begin{array}{ll} X_1 \sim \ldots \sim X_n & \sim D[X_{\varphi(k)}, \ldots, X_{\varphi(1)}, C[Y'_m, X_1]] \\ Y'_m \sim Y_1 \sim \ldots \sim Y_{m-1} & \sim C[Y'_m, X_1] \end{array}$$

We use induction on $m-h$: It is $k' = k+1$: $N' = N, m' = m$. The conditions for X_i is the same. We have to check that from the induction hypothesis we can prove the estimations for Y_i. The estimation is $mdt(Y_i) \leq (2N-k'+1)*d$ and $mdt(Y'_m) \leq (2N-k'+1)*d$. This implies $mdt(Y_i) \leq (2N-k)*d$ for $i < m$ and $mdt(Y'_m) \leq (2N-k)*d$, hence $mdt(Y_m) \leq (2N-k+1)*d$.

– $Y_m = C[Y_1(), Y'_m]$. Then

$$\begin{array}{ll} Y'_m \sim X_1 \sim \ldots \sim X_n & \sim D[X_{\varphi(k)}, \ldots, X_{\varphi(1)}, C[Y_1, Y'_m]] \\ Y_1 \sim \ldots \sim Y_{m-1} & \sim C[Y_1, Y'_m] \end{array}$$

We use induction on m. If $mdt(Y'_m) \leq N$, then also $mdt(Y_m) \leq N-k$. It is $k' = k+1$, which implies the estimation of Y_i as given in the lemma.

□

Lemma A.4. *Let*

$$\begin{array}{lll} X_1 \sim \ldots \sim & X_j \sim \ldots \sim X_n \sim C[Y_1, X_1] \\ Y_1 \sim \ldots \sim Y_m \sim & X_j \end{array}$$

and the depth of the holes in C is less than d. Let $N = n + m$.

Then there is some X_i such that $mdt(X_i) \leq (2N)*d$ for $j < i$ or $mdt(X_i) \leq (2N+1)*d$ for $i \leq j$ or some Y_i with $mdt(Y_i) \leq 2*N*d$

Proof. If $j = n$, then there are three cases. In every case we can use lemma A.3 and get a bound of $2Nd$.

In the case $j \neq n$, we use induction on $n+m$ and then on $n-j$.

Case analysis for X_n:

– X_n may have the holes of C in its side area, then we can use induction on n.
– $X_n = C[X'_n, X_1()]$, then the new chains of equations are $X_1 \sim \ldots \sim X_j \sim \ldots \sim X_{n-1} \sim C[X'_n, X_1]$ and $X'_n \sim Y_1 \sim \ldots Y_m \sim X_j$. Induction on $n-j$ shows the claim.
– $X_n = C[Y_1(), X'_n]$, then the new chains of equations are $X'_n \sim X_1 \sim \ldots \sim X_j \sim \ldots \sim X_{n-1} \sim C[Y_1(), X'_n]$ and $Y_1 \sim \ldots Y_m \sim X_j$. We can use induction on $n-j$.

□

Lemma A.5. *Let $X_1 \sim \ldots \sim X_j \sim \ldots \sim X_n \sim C[X_j, X_1]$ and the depth of the holes in C is less than d.*

Then there is some X_i such that $mdt(X_i) \leq (2N)*d$ for $j < i$ or $mdt(X_i) \leq (2N+1)*d$ for $i \leq j$.

Proof. Follows easily by considering the possibilities for X_n in the same way as already demonstrated using induction and Lemma A.4 □

Proposition A.6. *Let $X_1 \sim \ldots \sim X_n \sim C[X_1, X_1]$, and the depth of the holes in C is less than d. Then there is some X_i of with $mdt(X_i) \leq (2 * n + 1) * d$*

Proof. Follows from Lemma A.5 □

Lemma A.7. *Let an ambiguous SO-cycle $X_1 \sim C_1[X_2] \sim \ldots \sim X_n \sim C_n[X_1, X_1]$ be given, let $d_j = mdt(C_j), j = 1, \ldots, n-1$; d_n be the maximum of the depths of C_n and $d = \sum_{i=j}^{n} d_i$.*

Let $e_i = d - \sum_{k=1}^{i} d_k$. Then there is some X_i with $mdt(X_i) \leq (3n+1-e_i) * d$

Proof. Using induction on the torque $\sum_{i=1}^{n-1} i * d_i$. If all $d_i = 0$ for all $i < n$, then Proposition A.6 shows the claim.

Let j be the first index such that C_j is not trivial. We can assume that $j < n$. Consider the possibilities for X_j for $j > 1$ If X_j contains the hole of C_j in the side area, then $X_{j-1} \sim C'_j[X_{j+1}]$ where $mdt(C'_j) = mdt(C_j)$ and induction shows the claim.

If $X_j = C_j[X'_j]$, then the part of the chain is modified to: $\ldots, X_{j-1} \sim C_{j-1}C_j[X'_j], X'_j \sim X_{j+1}, \ldots$. Since in the torque $(j-1) * d_{j-1} + j * d_j$ is replaced by $(j-1) * (d_{j-1} + d_j)$, we can use induction.

In the cases where $j = 1$, the torque is also decreased, however, C_n is increased. □

Theorem A.8. *In an ambiguous SO-cycle $X_1 \sim C_1[X_2]$, $X_2 \sim C_2[X_3], \ldots \ldots X_n \sim C_n[X_1, X_1]$, let d be the sum of the depths of X_{i+1} in C_i for $i = 1, \ldots, n-1$ plus the maximal depths of the holes in C_n.*

Then there is some X_i of main depth less than $(3 * n + 1) * d$

Proof. This is a special case of Lemma A.7. □

Example A.9. Consider the equation chain $X \sim Y \sim f(X, X, t)$, where we omit the arguments of context variables. Then either $Y = f(X, X, Y')$, hence $X \sim f(X, X, Y')$, $X = Id$. If $Y = f(Y', X, t)$, then $Y' \sim X \sim f(Y', X, t)$, and $X = f(Y', f(Y', X', t), t)$. Then $Y' \sim f(Y', f(Y', X', t), t)$, hence $Y' = Id$. This gives a bound of 1 in contrast to 7 as given by Theorem A.8.