Czechoslovak Mathematical Journal

Robert Szymański

Decidability of weak equational theories

Czechoslovak Mathematical Journal, Vol. 46 (1996), No. 4, 629–664

Persistent URL: http://dml.cz/dmlcz/127325

Czechoslovak Mathematical Journal, 46 (121) 1996, Praha

DECIDABILITY OF WEAK EQUATIONAL THEORIES

ROBERT SZYMAŃSKI, Warsaw

(Received June 10, 1994)

1. INTRODUCTION

Transitivity is known to cause serious problems in establishing decidability. In checking whether an equation $p \approx q$ follows from Rudak's *weak transitivity* condition we shall use the set $\mathbf{D}_{pq}^{E}$ of all terms, which are defined in each model of E whenever both p and q are defined (on a given tuple). The main part of the paper contains notions and theorems describing the structure of $\mathbf{D}_{pq}^{E}$. This set being in general infinite, we present its finite graph representation and algorithms yielding basic information on its structure via the corresponding graph. This representation of $\mathbf{D}_{pq}^{E}$ leads finally to an algorithm deciding whether an equation $p \approx q$ can be deduced from E by the weak transtivity condition.

The paper is divided into two parts. In the first part we solve the decidability problem for weak equational theories in the case of unary signatures. In the second part, starting with Ch. 8, we generalize the results of the first part to arbitrary signature.

In Ch. 2 we define the basic notions, which will be used throughout the paper.

Ch. 3 contains Birkhoff's deduction rules for equational logic, the new weak transitivity condition, the definition of the set $\mathbf{D}_{pq}^{E}$ induced by a set of equations E and an equation $p \approx q$ and a completeness theorem for weak equational logic, proved by Rudak in [2]. The definition of $\mathbf{D}_{pq}^{E}$ is recursive and the set which gives the basis for the construction is called the *initial set.*

We shall learn more of the set $\mathbf{D}_{pq}^{E}$ in Ch. 4. We shall first define a "horizontal" structure called *level.* The levels contain those terms from the set $\mathbf{D}_{pq}^{E}$, which appear in the consecutive steps of its construction. Then we introduce a "vertical" structure in $\mathbf{D}_{pq}^{E}$, based on *braids.* The number of all braids is finite and each of them starts with a term from the initial set. The main feature of braids is the fact that they contain all the information on equations which were used for the construction of the

set $\mathbf{D}_{pq}^{E}$. We prove that the braids cover $\mathbf{D}_{pq}^{E}$ and we shall show how the step-by-step construction of this set can be reduced to a step-by-step construction of braids. The Duplication Theorem turns out to be a very strong tool, enabling to add in one step infinitely many terms to a braid.

The description of $\mathbf{D}_{pq}^{E}$ being still very cumbersome (braids can be infinite), we introduce in Ch. 5 another representation of this set, which will be a finite directed graph. Its vertices are terms from the initial set (which is finite), while its edges represent basic connections (*bridges*) between braids. Using this graph we shall be able to decide whether a term t is in $\mathbf{D}_{pq}^{E}$.

The application of the tools defined in previous chapters will be presented in Ch. 6. The Bridge Theorem contains an algorithm for finding bridges and this algorithm only uses the information represented by the finite graph.

In Ch. 7 we prove that the construction of the graph representing $\mathbf{D}_{pq}^{E}$ can be performed in a finite number of steps and that the set of all terms which can be recovered from the graph is exactly the set $\mathbf{D}_{pq}^{E}$. Then we present an algorithm which decides for a given equation $t \approx s$ whether it can be deduced from a set E of equations by weak transitivity.

This algorithm solves the decidability problem for unary signatures. In Ch. 8 we return to arbitrary signatures and we show that all that has been done for the unary ones can be easily generalized to other cases. In particular we prove that again the set of terms recovered from the graph is exactly the whole set $\mathbf{D}_{pq}^{E}$. The main deciding algorithm is essentially the same as for the unary case.

2. Preliminary notions

Algebraic notions not defined here will be used as in Grätzer[1]. A signature is a pair $(\mathbf{F}, \delta)$, where $\mathbf{F}$ is a set and $\delta\colon \mathbf{F} \longrightarrow \omega$ is an arity function. $\mathbf{A}$ is a partial algebra of signature $(\mathbf{F}, \delta)$ iff $\mathbf{A} = (A, (f^{\mathbf{A}})_{f\in\mathbf{F}})$ and for each $f \in \mathbf{F}$, $f^{\mathbf{A}}$ is a partial $\delta(f)$-ary operation in $\mathbf{A}$ (the domain of a partial function g will be denoted by $\operatorname{dom} g$). We shall often denote the family $(f^{\mathbf{A}})_{f\in\mathbf{F}}$ by $\mathbf{F}^{\mathbf{A}}$.

Fix a signature $\tau = (\mathbf{F}, \delta)$ and let X be any countable set. Let $\mathbf{P} = (P, \mathbf{F}^{\mathbf{P}})$ be the (total) algebra of τ-terms over X. A relative subalgebra $\mathbf{D} = (D, \mathbf{F}^{\mathbf{P}})$ of $\mathbf{P}$ is an *initial segment* of $\mathbf{P}$ iff $X \subseteq D$ and whenever

$$f(r_1, \ldots, r_{\delta(f)}) \in D$$

for $f \in \mathbf{F}$, then also

$$r_1, \ldots, r_{\delta(f)} \in D.$$

Thus, for instance, the relative subalgebra $\mathbf{D}(r) = (D(r), \mathbf{F}^{\mathbf{D}(r)})$, where $D(r)$ is the set consisting of all the variables in X and of all the subterms of the term $r \in \mathbf{P}$, is an initial segment of $\mathbf{P}$ and it is the least initial segment of $\mathbf{P}$ containing the term r.

We say that an equation $p \approx q$ *holds weakly* in $\mathbf{A}$ or, equivalently, is a *weak equation* in $\mathbf{A}$ (and we write $\mathbf{A} \models_w p \approx q$), iff for any $a_1, \ldots, a_n \in A$, whenever $(a_1, \ldots, a_n) \in \operatorname{dom} p^{\mathbf{A}} \cap \operatorname{dom} q^{\mathbf{A}}$, then $p^{\mathbf{A}}(a_1, \ldots, a_n) = q^{\mathbf{A}}(a_1, \ldots, a_n)$. For instance, consider an algebra $\mathbf{B} = (\{a, b\}, f^{\mathbf{B}})$ with $a \neq b$ and f binary. Assume $\operatorname{dom} f^{\mathbf{B}} = \{(a, b), (b, a)\}$ and

$$f^{\mathbf{B}}(a, b) = a, \quad f^{\mathbf{B}}(b, a) = b.$$

Then $\mathbf{B} \models_w f(x, y) \approx f(x, x)$ and $\mathbf{B} \models_w f(x, x) \approx f(y, x)$, while $\mathbf{B} \models_w f(x, y) \approx f(y, x)$ does not hold.

3. Weak transitivity

G. Birkhoff has formulated a complete set of inference rules for the equational logic of total algebras. Thus for any set E of equations, $E = EqMod(E)$ iff

(i) $x \approx x \in E$ for every $x \in X$

(ii) $p \approx q \in E$, if $q \approx p \in E$

(iii) $p' \approx q' \in E$, if $p \approx q \in E$ and p', q' are obtained from p and q, respectively, by replacing all occurrences of some variable by a term

(iv) $f(p_1, \ldots, p_{\delta(f)}) \approx f(q_1, \ldots, q_{\delta(f)}) \in E$, if $p_i \approx q_i \in E$ for $i = 1, 2, \ldots, \delta(f)$ and $f \in \mathbf{F}$

(v) $p \approx q \in E$, if there is an $r \in \mathbf{P}$ such that $p \approx r \in E$ and $r \approx q \in E$.

The example at the end of the previous Chapter shows that transitivity is not a valid rule for weak equations in partial algebras. However, some kind of transitivity is still needed. In [2] the following new rule has been introduced and proved adequate:

(v′) If for some terms $p, q \in \mathbf{P}$ there are terms $r_1, \ldots, r_n \in \mathbf{D}_{pq}^{E}$ such that $p \approx r_1, r_1 \approx r_2, \ldots, r_{n-1} \approx r_n, r_n \approx q \in \mathbf{E}$ for some n, then $p \approx q \in \mathbf{E}$.

Here $\mathbf{E}$ denotes the closure of E with respect to rules (i)–(iv) and $\mathbf{D}_{pq}^{E}$ is the set of all terms which induce in every model of E a term function defined on a given tuple of arguments whenever both term functions induced by p and q are defined on that tuple. The precise definition is the following:

Definition 3.1. Define $\mathbf{D}_0 = \mathbf{D}(p) \cup \mathbf{D}(q)$ and for $n \in \omega$,

$$\mathbf{D}_{n+1} = \mathbf{D}_n \cup \bigcup_{f \in \mathbf{F}} \{f(r_1, \ldots, r_{\delta(f)}) : \text{ there are } s_1, \ldots, s_{\delta(f)} \in \mathbf{P} \text{ such that } r_1, \ldots, r_{\delta(f)}, f(s_1, \ldots, s_{\delta(f)}) \in \mathbf{D}_n \text{ and } s_i \approx r_i \in \mathbf{E}, \text{ for } i = 1, 2, \ldots, \delta(f)\}$$

Then

$$\mathbf{D}^E_{pq} = \bigcup_{n\in\omega} \mathbf{D}_n.$$

The following theorem shows that (v′) can indeed replace transitivity in Birkhoff-like rules for weak equational logic:

Theorem 3.2. (The Completeness Theorem ([2])**.)** *For any set of equations* E, $E = Eq_w \operatorname{Mod}_w(E)$ *iff* E *is closed with respect to rules* (i)–(iv) *and* (v′).

Here $\operatorname{Mod}_w(E)$ denotes the class of all partial algebras of the appropriate signature, in which all the equations in E hold weakly; $Eq_w(K)$ is the set of all equations (of the appropriate signature) which hold weakly in all partial algebras from the class K.

4. Duplication theorem

From now on and until Ch. 8 we shall restrict our considerations to unary signatures only. This will be very convenient, since it will significantly simplify the notation. We shall see in Ch. 8 that this simplification does not restrict the generality of the solution, since it can be easily extended to non-unary cases.

Fix a signature $\tau = (\mathbf{F}, \delta)$ such that $\delta(f) = 1$ for each $f \in \mathbf{F}$. Let E be a fixed finite set of τ-equations and let $\alpha, \beta \in \mathbf{P}$ be fixed (unary) terms.

The definition of the set $\mathbf{D}^E_{\alpha\beta}$ is now somewhat simpler:

Definition 4.1. Define

$\mathbf{D}_0 = \mathbf{D}(\alpha) \cup \mathbf{D}(\beta)$ and for $n \in \omega$,

$$\mathbf{D}_{n+1} = \mathbf{D}_n \cup \bigcup_{f\in\mathbf{F}} \{f(r)\colon \text{ there is } s \in \mathbf{P} \text{ such that } r, f(s) \in \mathbf{D}_n \text{ and } s \approx r \in \mathbf{E}\}$$

Then

$$\mathbf{D}^E_{\alpha\beta} = \bigcup_{n\in\omega} \mathbf{D}_n.$$

Remark 4.2. We can assume without loss of generality that the set $\mathbf{D}_0$ contains a finite set of variables only, namely those, which occur in terms α, β and in terms of E.

At this stage we are interested in deciding in finite time whether a term t is in the set $\mathbf{D}^E_{\alpha\beta}$. In general this set is infinite; its definition is inductive and rather complex. If we constructed this set using its definition, we would have to start with the initial set $\mathbf{D}_0$ and then we would construct consecutive sets $\mathbf{D}_i$. We shall adopt another solution here by defining new structures within $\mathbf{D}^E_{\alpha\beta}$: these will make clearer the

relations between terms in this set and it will bring to evidence how and when new terms are added to its construction.

The first partition of $\mathbf{D}^E_{\alpha\beta}$ is naturally implied by its definition:

Definition 4.3. A *k-level* $\mathbf{L}_k$ is the set of new terms in $\mathbf{D}_k$ i.e.,

$$\mathbf{L}_0 = \mathbf{D}_0 \text{ and } \mathbf{L}_{k+1} = \mathbf{D}_{k+1} \setminus \mathbf{D}_k.$$

It is easy to see that if $\mathbf{L}_k = \emptyset$, then $\mathbf{D}^E_{\alpha\beta} = \mathbf{D}_{k-1}$

Example 1. Assume

$$\begin{aligned} \alpha &= f(t(p(x))), \\ \beta &= g(h(y)) \text{ and} \\ E &= \{t(p(x)) = g(h(y)), p(x) = h(y)\}. \end{aligned}$$

(We shall use this example until Ch. 8 to illustrate the notions introduced in the first part of the paper).

For such terms α and β and the set E we may represent the levels of the set $\mathbf{D}_2$ as follows:

$\mathbf{L}_0$:	$f(t(p(x)))$	$t(p(x))$	$p(x)$	x	$g(h(y))$	$h(y)$	y
$\mathbf{L}_1$:	$f(g(h(y)))$	$t(h(y))$			$g(p(x))$		
$\mathbf{L}_2$:	$f(t(h(y)))$	$f(g(p(x)))$					

The level $\mathbf{L}_0$ coincides with the set $\mathbf{D}_0$. The next level $\mathbf{L}_1$ contains the terms $f(g(h(y)))$, $t(h(y))$, $g(p(x))$. How did they appear in this construction? The equation $t(p(x)) \approx g(h(y)) \in E$ is an equation between the main subterm of the term α and a term from the set $\mathbf{D}_0$ (in this case it is the term β), so according to the definition of $\mathbf{D}^E_{\alpha\beta}$, we can substitute the term β for this main subterm of α. This yields the term $f(g(h(y)))$, which falls into $\mathbf{D}_1$.

In $\mathbf{L}_2$ we find two terms: $f(t(h(y)))$ and $f(g(p(x)))$. Let's have a closer look at the first. According to rule (iv), the equation $p(x) \approx h(y)$ being in E, we can deduce that $t(p(x)) \approx t(h(y))$ is in $\mathbf{E}$. Both terms in the latter equation are in $\mathbf{D}_1$, since $\mathbf{D}_1 = \mathbf{L}_0 \cup \mathbf{L}_1$. The term $t(p(x))$ is the main subterm of a term in $\mathbf{D}_0$, while the term $t(h(y))$ is itself in $\mathbf{D}_1$. Therefore, by the inductive definition of $\mathbf{D}^E_{\alpha\beta}$, the term $f(t(h(y)))$ will appear in $\mathbf{D}_2$. Similarly, starting with the equation $g(p(x)) \approx g(h(y)) \in \mathbf{E}$, we get the term $f(g(p(x))) \in \mathbf{D}_2$.

In our example $\mathbf{L}_3$ is empty and therefore $\mathbf{D}^E_{\alpha\beta} = \mathbf{D}_2 = \mathbf{L}_0 \cup \mathbf{L}_1 \cup \mathbf{L}_2$.

We shall now define the most important structure within $\mathbf{D}^E_{\alpha\beta}$:

Definition 4.4. A *braid* starting in $t \in \mathbf{D}_0$ (denoted by $\mathbf{T}(t)$) is defined as follows: Let $\mathbf{T}_0(t) = \{t\}$

$$\mathbf{T}_{k+1}(t) = \mathbf{T}_k(t) \cup \bigcup_{p \in \mathbf{F}} \{p(r) : \text{ there is an } s \in \mathbf{P} \text{ such that } p(s) \in \mathbf{T}_k(t), \\ r \in \mathbf{D}_k \text{ and } s \approx r \in \mathbf{E}\}.$$

Then

$$\mathbf{T}(t) = \bigcup_{k \in \omega} \mathbf{T}_k(t).$$

The term t is called the *root* of $\mathbf{T}(t)$. Clearly the number of braids is constant for given $\alpha, \beta \in \mathbf{P}$ and a set of equations E and it is equal to the number of terms in $\mathbf{D}_0$. Notice that each term $t \in \mathbf{D}_0$ sets its braid and that all terms in a braid $\mathbf{T}(t)$ have a common external operation symbol, which is the external operation symbol of t.

Our intention being the use of braids for the construction of $\mathbf{D}^E_{\alpha\beta}$, we shall mostly use *partial* braids $\mathbf{T}_k(t)$, the structure of which is similar to the structure of the whole braid.

Example 2. For terms α, β and the set E as in the previous example, we find three braids containing more than one element: $\mathbf{T}(f(t(p(x))))$, $\mathbf{T}(t(p(x)))$ and $\mathbf{T}(g(h(y)))$. Let's analyse the first of these:

$$\begin{array}{cc} f(t(p(x))) & \\ f(g(h(y))) & \\ f(t(h(y))) \qquad & f(g(p(x))) \end{array}$$

We start with a partial braid $\mathbf{T}_0(f(t(p(x))))$, which has one element only—the root. The equation $t(p(x)) \approx g(h(y)) \in \mathbf{E}$ induces a new term $f(g(h(y)))$, which belongs thus to $\mathbf{T}_1(f(t(p(x))))$. Then, using appropriate equations from $\mathbf{E}$, we get the term $f(t(h(y)))$ from $f(t(p(x)))$ and $f(g(p(x)))$ from $f(g(h(y)))$. This yields a four-element braid $\mathbf{T}(f(t(p(x))))$. The other two braids contain two elements each.

Lemma 4.5. *If $g \in \mathbf{D}_0$, then for any $t \in \mathbf{T}(g)$ there exists a sequence of terms $p_1, \ldots, p_n \in \mathbf{T}(g)$ for some $n \in \omega$ such that*

$$g \approx p_1, p_1 \approx p_2, \ldots, p_{n-1} \approx p_n, p_n \approx t \in \mathbf{E}.$$

Proof. We proceed by induction on the structure of the braid $\mathbf{T}(g)$. Let $t = f(r)$ for some $f \in \mathbf{F}$ and some $r \in \mathbf{P}$. If $t \in \mathbf{T}_0(g)$, then t is identically equal

to g and the lemma holds. If $t \in \mathbf{T}_{n+1}(g)$, then according to the definition of $\mathbf{T}(g)$ there is a term $r \in \mathbf{D}_n$ and a term $f(s) \in \mathbf{T}_n(g)$ for some $s \in \mathbf{P}$ with $r \approx s \in \mathbf{E}$. By rule (iv) we have $f(r) \approx f(s) \in \mathbf{E}$ and by inductive assumption for $f(s)$ there is a sequence

$$p_1, \ldots, p_n \in \mathbf{T}(g)$$

such that

$$g \approx p_1, p_1 \approx p_2, \ldots, p_{n-1} \approx p_n, p_n \approx f(s) \in \mathbf{E}.$$

Then $p_1, \ldots, p_n, f(s) \in \mathbf{T}(g)$ is the required sequence of terms. □

Observe that a simple modification of the proof above (substituting everywhere $\mathbf{T}_k(g)$ for $\mathbf{T}(g)$) shows that the lemma also holds true for any partial braid $\mathbf{T}_k(g)$ i.e., any term in a partial braid $\mathbf{T}_k(g)$ can be linked with the root by a sequence of appropriate terms belonging to $\mathbf{T}_k(g)$.

Lemma 4.6. *If $g \in \mathbf{D}_0$, then for any terms $p, t \in \mathbf{T}(g)$ there exists a sequence of terms $p_1, \ldots, p_n \in \mathbf{T}(g)$ for some $n \in \omega$ such that*

$$p \approx p_1,\ p_1 \approx p_2,\ \ldots,\ p_{n-1} \approx p_n,\ p_n \approx t \in \mathbf{E}.$$

Proof. Let $s_1, \ldots, s_l$ and $r_1, \ldots, r_m \in \mathbf{T}(g)$ be sequences of terms which exist for p and t, respectively, by the previous lemma, i.e.,

$$g \approx s_1,\ s_1 \approx s_2,\ \ldots,\ s_{l-1} \approx s_l,\ s_l \approx p \in \mathbf{E}$$

and

$$g \approx r_1,\ r_1 \approx r_2,\ \ldots,\ r_{m-1} \approx r_m,\ r_m \approx t \in \mathbf{E}.$$

Then $s_l, s_{l-1}, \ldots, s_1, g, r_1, r_2, \ldots, r_m \in \mathbf{T}(g)$ is the required sequence of terms. □

Observe that here, too, the lemma holds true for any partial braid $\mathbf{T}_k(g)$. Note also that in some cases there may be a sequence satisfying the previous lemma, the length of which is smaller than $l+1+m$; the sequence indicated in the proof, however, always exists.

Definition 4.7. Given arbitrary terms $p, t \in \mathbf{T}(g)$ (or $p, t \in \mathbf{T}_k(g)$ for some $k \in \omega$), a *chain* $\mathbf{F}_g(p, t)$ is a sequence of terms $p_1, \ldots, p_n \in \mathbf{T}(g)$ (or $p_1, \ldots, p_n \in \mathbf{T}_k(g)$, correspondingly), which exists for p and t by Lemma 4.6.

Example 3. In our example the chain $\mathbf{F}_g\big(f(t(p(x))), f(g(p(x)))\big)$ in the braid $\mathbf{T}(f(t(p(x))))$ consists of the following terms:

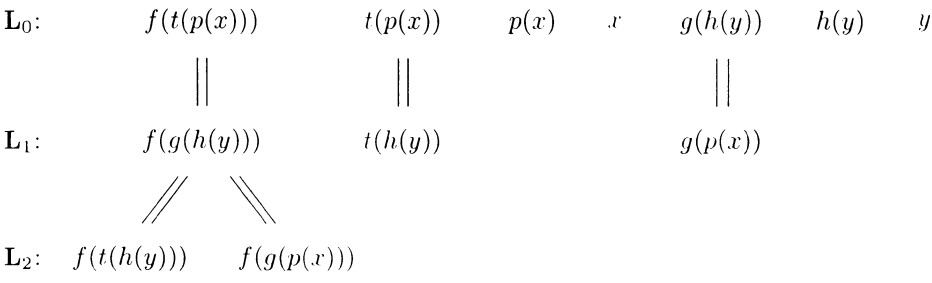
$$f(t(p(x))), \quad f(g(h(y))), \quad f(g(p(x))).$$

since the equations

$$f(t(p(x))) \approx f(g(h(y))), \quad f(g(h(y))) \approx f(g(p(x)))$$

are in $\mathbf{E}$.

We can now represent the braids in $\mathbf{D}_2$, indicating the equations within the braids:



It can be seen in the example above that braids cover the whole set $\mathbf{D}^E_{\alpha\beta}$. This important property is generalized in the following

Lemma 4.8.

$$\mathbf{D}^E_{\alpha\beta} = \bigcup_{p\in\mathbf{D}_0} \mathbf{T}(p).$$

Proof. The inclusion $\supseteq$ follows from the definition of a braid. The other inclusion will be proved inductively with respect to the index of the set $\mathbf{D}_n$, to which a term $t \in \mathbf{D}^E_{\alpha\beta}$ belongs. If $t \in \mathbf{D}_0$, then t is the root of a braid and therefore $t \in \bigcup_{p\in\mathbf{D}_0} \mathbf{T}(p)$. Assume $t \in \mathbf{D}_{n+1}$. According to the definition of $\mathbf{D}^E_{\alpha\beta}$, there exist terms $r, f(s) \in \mathbf{D}_n$ such that $t = f(r)$ and $r \approx s \in \mathbf{E}$. By inductive assumption for terms in $\mathbf{D}_n$, $f(s) \in \mathbf{T}(g)$ for some $g \in \mathbf{D}_0$. If $f(s) \in \mathbf{T}_k(g)$ for some $k \in \omega$, then by the definition of a braid, $t = f(r) \in \mathbf{T}_{k+1}(g) \subseteq \mathbf{T}(g) \subseteq \bigcup_{p\in\mathbf{D}_0} \mathbf{T}(p)$. □

The definition of $\mathbf{D}^E_{\alpha\beta}$ indicates that to construct $\mathbf{D}_{n+1}$ from the previous sets, we use equations from the set $\mathbf{E}$ such that both terms in an equation belong to $\mathbf{D}_n$. We know now, that these terms belong to some braids and thus what we are actually looking for are equations between partial braids. This suggests a new notion, which will prove to be as important for our purposes as that of a braid.

Definition 4.9. A *bridge* between two partial braids $\mathbf{T}_k(t)$ and $\mathbf{T}_l(p)$ for some $k, l \in \omega$ is any equation $g \approx s \in \mathbf{E}$ such that g is a main subterm of a term in $\mathbf{T}_k(t)$ and s is a term in $\mathbf{T}_l(p)$.

By extension, we shall call an equation $g \approx s \in \mathbf{E}$ a *bridge* between $\mathbf{T}(t)$ and $\mathbf{T}(p)$ or between $\mathbf{T}_k(t)$ and $\mathbf{T}(p)$ or between $\mathbf{T}(t)$ and $\mathbf{T}_l(p)$ when g is a main subterm of a term in the first (possibly partial) braid and s is in the second.

Example 4. In our example the equation $t(p(x)) \approx g(h(y)) \in \mathbf{E}$ is a bridge between $\mathbf{T}(f(t(p(x))))$ and $\mathbf{T}(g(h(y)))$, the equation $p(x) \approx h(y)$ is a bridge between $\mathbf{T}(t(p(x)))$ and $\mathbf{T}(h(y))$. Each equation which yields a new term in the inductive construction of $\mathbf{D}^E_{\alpha\beta}$ is in fact a bridge between appropriate braids. This suggests that finding a new term in the construction of a braid amounts to finding a bridge between this braid and other braids in $\mathbf{D}^E_{\alpha\beta}$. In fact, as the following theorem shows, a bridge is a very powerful tool in the construction of braids.

Theorem 4.10. (Duplication Theorem.) *For any terms $t, r \in \mathbf{D}_0$, $h, p \in \mathbf{P}$ and $g \in \mathbf{F}$, if $g(h) \in \mathbf{T}_k(t)$ for some $k \in \omega$, $p \in \mathbf{T}_m(r)$ for some $m \in \omega$ and $h \approx p \in \mathbf{E}$, then $g(\mathbf{T}(r)) \subseteq \mathbf{T}(t)$, where $g(\mathbf{T}(r)) = \{g(s)\colon s \in \mathbf{T}(r)\}$.*

The theorem states that if there is a bridge between two partial braids $\mathbf{T}_k(t)$ and $\mathbf{T}_m(r)$, then the whole braid $\mathbf{T}(r)$ will be copied into $\mathbf{T}(t)$; more precisely, all the terms in $\mathbf{T}(r)$ will occur as main subterms in $\mathbf{T}(t)$. What is crucial here is that one bridge is sufficient to copy the whole braid—this property will prove essential for establishing decidability of weak equational theories.

Proof. Let $t, r \in \mathbf{D}_0$, $h, p \in \mathbf{P}$ and $g \in \mathbf{F}$ satisfy the assumptions of the theorem. Since $h \approx p \in \mathbf{E}$, the term $g(p)$ is in $\mathbf{T}_{k+1}(t)$. For p and any term $q \in \mathbf{T}_m(r)$ there exists by lemma 4.6 a chain $\mathbf{F}_r(p, q)$, i.e., a sequence of terms $p_1, \ldots, p_n \in \mathbf{T}(g)$ and a corresponding sequence of equations

$$p \approx p_1,\ \ldots,\ p_{n-1} \approx p_n,\ p_n \approx q \in \mathbf{E}.$$

Now, due to these equations, we can consecutively "add" to the braid $\mathbf{T}(t)$ the terms $g(p_1), \ldots, g(p_{n-1}), g(p_n)$ and finally also $g(q)$. The term q being an arbitrary term in $\mathbf{T}_m(r)$, we see that all the terms in $\mathbf{T}_m(r)$ will appear in $\mathbf{T}(t)$ as main subterms. It is now sufficient to observe that $p \in \mathbf{T}_m(r) \subseteq \mathbf{T}_i(r)$ for all $i \geqslant m$ and $\mathbf{T}_j(r) \subseteq \mathbf{T}_m(r)$ for all $j \leqslant m$. Thus for any term $q \in \mathbf{T}(r)$ the term $g(q)$ is in $\mathbf{T}(t)$. □

The theorem shows that one equation $h \approx p$ with $p \in \mathbf{T}(r)$ will lead to any term in this braid, independently of whether the term has already occurred at the m -th step of the construction of $\mathbf{T}(r)$ or it will appear later. To find a common-life analogy for this situation, imagine a bridge constructed to join two highways. Once you cross it

going from the first of these highways to the second, you can reach any point on the second. Moreover, if the second highway is extended in some future, you can still reach the new points due to the same connection.

Returning to our main topic, we shall say that a term q or a braid $\mathbf{T}(r)$ has been *copied* into a braid $\mathbf{T}(t)$ when q or, respectively, all the terms of $\mathbf{T}(r)$ appear in $\mathbf{T}(t)$ as main subterms.

5. The graph of $D^E_{\alpha\beta}$

Let α, β be arbitrary terms and let E be a finite set of equations. We introduce here a graph $\mathbf{G}$ representing the set $\mathbf{D}^E_{\alpha\beta}$.

Definition 5.1. Let $\mathbf{G} = (V, K)$ be a directed graph without multiple edges such that:

(a) $V = \mathbf{D}_0$.

(b) For any $s, t \in V$, there is an edge in K from s to t iff there is a bridge between $\mathbf{T}(s)$ and $\mathbf{T}(t)$ in $\mathbf{D}^E_{\alpha\beta}$.

Then $\mathbf{G}$ is called *the graph of* $\mathbf{D}^E_{\alpha\beta}$.

It is clear from this definition that the graph of $\mathbf{D}^E_{\alpha\beta}$ is finite, the number of vertices being equal to the number of terms in $\mathbf{D}_0$. The edges in the graph are induced by bridges and they connect the roots of braids linked by a bridge. An edge from a term s to a term t (corresponding to a bridge between $\mathbf{T}(s)$ and $\mathbf{T}(t)$) will be denoted by $K(s,t)$ and it will be labelled by the external operation symbol in the braid $\mathbf{T}(s)$ (recall that all terms in a braid have the same external operation symbol).

The graph $\mathbf{G}$ will turn out to be a finite representation of the corresponding set $\mathbf{D}^E_{\alpha\beta}$ and thus also of all its braids. We shall use the graph to read the set of all the terms which belong to a given braid and we shall check for each term t whether it is in $\mathbf{D}^E_{\alpha\beta}$. We shall see the exact rules for reading terms from the graph after an example. The main idea consists in the possibility of identifying a braid with the set of all terms which can be read from the graph starting from the root of the braid.

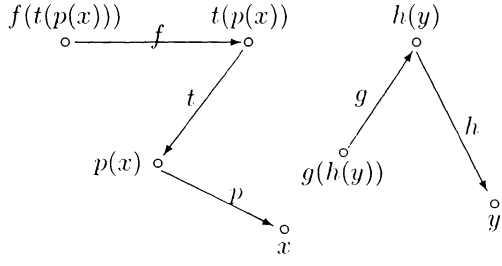
Observe that due to the properties of the set $\mathbf{D}^E_{\alpha\beta}$ (see [2]) it is closed with respect to subterms; moreover, by Birkhoff's rule (i) all equations of the form $t \approx t$ for an arbitrary term t are in $\mathbf{E}$. Thus if $f(p) \in \mathbf{D}_0$, then also $p \in \mathbf{D}_0$ and $p \approx p \in \mathbf{E}$. The term p is then on one hand the main subterm of the term $f(p)$ and on the other, it is itself in $\mathbf{D}_0$. Hence, according to the definition of the graph $\mathbf{G}$, there should be an edge $K(f(p), p)$.

Now, instead of constructing the set D, we shall construct the graph G. Whenever we find a new bridge (which has not been represented in the graph yet), we add an

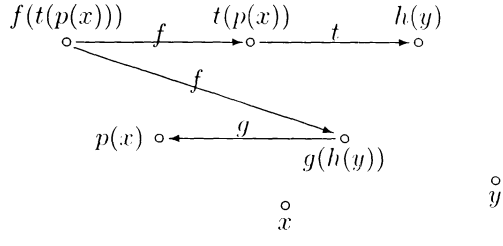
edge between the corresponding vertices i.e., between the braids connected by the bridge. The first step of the construction consists in drawing all th edges which are induced by Borkhoff's rule (i), independently of the set E (this will be clearly seen in the example below). The construction of the graph will be completed, when no new bridge is found. Then the set of terms included in the graph will be the set D, which will be formally proved later.

Example 5. Let us see first, how would the graph $\mathbf{G}$ look like after the first step in its construction, i.e., after introducing only those edges $K(f(p), p)$ which correspond to $f(p) \in \mathbf{D}_0$. Recall that

$\alpha = f(t(p(x)))$,
$\beta = g(h(y))$,
$E = \{t(p(x)) \approx g(h(y)), p(x) \approx h(y)\}$.



Here all the edges lead from a term to its main subterm. We shall now draw the complete graph with all the edges—except for those term—main subterm edges that have been already introduced above (the presence of the edge from $f(t(p(x)))$ to $t(p(x))$ will be explained later).



Here the only missing edges are:

$$K(t(p(x)),\ p(x)),\ K(p(x),x),\ K(g(h(y)),\ h(y)),\ K(h(y),y).$$

We shall return to them later. Let us analyze first the appearance of the edges indicated in the graph above.

The edge from $f(t(p(x)))$ to $g(h(y))$ is induced by the bridge $t(p(x)) \approx g(h(y))$. As explained above, this edge bears the label f. The bridge $p(x) \approx h(y)$ yields an edge from $t(p(x))$ to $h(y)$ and this edge is labelled by t. The equation $h(y) \approx p(x)$, which by symmetry is in $\mathbf{E}$, is a bridge between $\mathbf{T}(g(h(y)))$ and $\mathbf{T}(p(x))$, which induces an edge from $g(h(y))$ to $p(x)$; this edge is labelled by g, which is the external operation symbol in $\mathbf{T}(g(h(y)))$.

Note that we have found another equation in $\mathbf{E}$, which lead to the inclusion of the term $f(g(p(x)))$ in $\mathbf{D}^E_{\alpha\beta}$; this was $g(h(y)) \approx g(p(x))$. This equation is a bridge between $\mathbf{T}(\alpha)$ and $\mathbf{T}(\beta)$, but we already have an edge between α and β. There is no need to introduce a second edge between these two vertices since, as we shall see soon, the term $f(g(p(x)))$ can be recovered from the graph due to the existing edges.

We expect of $\mathbf{G}$ to represent the set $\mathbf{D}^E_{\alpha\beta}$ completely: all information about this set should be readable from the graph itself. In particular, we shall be often lead to decide whether a given term is in $\mathbf{D}^E_{\alpha\beta}$. We shall now introduce rules for reading terms from the graph and we shall prove later that the set of terms read from the graph is equal to $\mathbf{D}^E_{\alpha\beta}$.

Rules for reading terms from the graph $\mathbf{G}$

1. Reading can start at any vertice of $\mathbf{G}$.
2. We can stop reading at any vertice of $\mathbf{G}$.
3. Reading is performed along edges only, according to their directions.
4. Reading starts with an empty sequence of symbols; passing from one vertice to another, we add to the sequence of symbols the label of the corresponding edge.
5. When we stop reading at a vertice t, we add to the sequence of symbols the whole term t.

Example 6. Let us return to the edges missing in the second graph of the previous example:

$$K(t(p(x)),\ p(x)),\ K(p(x),x),\ K(g(h(y)),\ h(y)),\ K(h(y),y).$$

It can be observed that these edges introduce no new terms: note that any term which can be read from the first graph in the example is in the graph already—as one of its vertices. Thus for sake of clarity we shall omit such edges in the representation of graphs.

We shall now use the second graph in the previous example to find all the terms belonging to $\mathbf{D}^E_{\alpha\beta}$. Clearly we shall include all the terms at the vertices (since we can start reading at any vertice and then stop right there). These are terms from the set $\mathbf{D}_0$. Next, starting at the vertice $f(t(p(x)))$ and going to $g(h(y))$ we first write the symbol f, which labels the corresponding edge. If we stop here, we add the whole term at the final vertice to the sequence and we get $f(g(h(y)))$. If we proceed to the vertice $p(x)$, we add the symbol g to f and then the whole term $p(x)$, since there are no edges starting in this vertice. Thus we get the term $f(g(p(x)))$. It can be observed now that there is no need to indicate the bridge $g(h(y)) \approx g(p(x))$ on the graph, since the term induced by this equation can be already obtained due to the first bridge $t(p(x)) \approx g(h(y))$ between $\mathbf{T}(\alpha)$ and $\mathbf{T}(\beta)$. Finally, we can easily read the terms

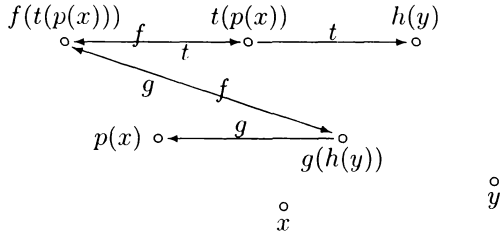
$$f(t(h(y))), g(p(x)), t(h(y)).$$

What new edges would we get if we include in E a new equation:

$$p(x) \approx f(t(h(y)))?$$

It is a bridge between $\mathbf{T}(g(h(y)))$ and $\mathbf{T}(\alpha)$, and between $\mathbf{T}(t(p(x)))$ and $\mathbf{T}(\alpha)$, since the term $p(x)$ is a main subterm in both braids $\mathbf{T}(g(h(y)))$ and $\mathbf{T}(t(p(x)))$ and $f(t(h(y)))$ is in $\mathbf{T}(\alpha)$. Our graph will now be as follows:

$\alpha = f(t(p(x)))$,
$\beta = g(h(y))$,
$E = \{t(p(x)) \approx g(h(y)), p(x) \approx h(y), p(x) \approx f(t(h(y)))\}$.



Now we can easily find all the terms in $\mathbf{D}^E_{\alpha\beta}$. Let us begin with the braid $\mathbf{T}(g(h(y)))$ and start at its root. Passing consecutively through edges labelled by g, f, g, f, t, f, g, f, t (note that different edges may have the same label) we get the term $g(f(g(f(t(f(g(f(t(h(y)\ldots)$. Taking another path: g, f, g, f, g, f, g, f, g, f we get

$g(f(g(f(g(f(g(f(t(p(x)\ldots)$, while the path $g, f, t, f, t, f, t, f, t, f, t, f$ yields $g(f(t(f(t(f(t(f(t(f(g(h(y)\ldots)$.

Clearly the braid $\mathbf{T}(g(h(y)))$ is infinite. This example shows that if there is a cycle in the graph which can be reached starting from a term t, then the braid $\mathbf{T}(t)$ is infinite. Thus we have three infinite braids in the set $\mathbf{D}^E_{\alpha\beta}$: $\mathbf{T}(\alpha), \mathbf{T}(\beta), \mathbf{T}(p(x))$.

The simplicity of the example leaves no doubt as to the fact that all terms read from the graph are indeed in $\mathbf{D}^E_{\alpha\beta}$. We shall now provide a formal proof.

Let $\mathbf{L}(t)$ be the set of all terms read from a graph $\mathbf{G}$ according to the rules, if we start from the vertice t.

Theorem 5.2. *For any term t in $\mathbf{D}_0$,*

$$\mathbf{T}(t) = \mathbf{L}(t).$$

P r o o f. We shall prove the inclusion of $\mathbf{L}(t)$ in $\mathbf{T}(t)$ by induction on the length k of the path required to read a term s from the graph $\mathbf{G}$ (starting in t). If $k = 1$, then the term s is the root of the braid and thus $s = t \in \mathbf{T}(t)$. Let $k = l + 1$ and assume that any term which can be read in not more than l steps is in $\mathbf{T}(t)$. Let $f_1, f_2, \ldots, f_l, f_{l+1}$ be the labels of the path which yields the term s and let q be the term corresponding to the path $f_2, \ldots, f_l, f_{l+1}$ (i.e., we start at the second vertice of the previous path; let this vertex be r). By inductive assumption, $q \in \mathbf{T}(r)$. Moreover, there is an edge f_1 leading from t to r: this edge represents a bridge, hence there is a bridge between $\mathbf{T}(t)$ and $\mathbf{T}(r)$. By the Duplication Theorem 4.10 we have that $f_1(\mathbf{T}(r)) \subseteq \mathbf{T}(t)$ and, in particular, $s = f_1(q) \in \mathbf{T}(t)$.

The other inclusion will be proved by induction of the index of the partial braid, to which a term $s \in \mathbf{T}(t)$ belongs. If $s \in \mathbf{T}_0(t)$, then $s = t$, i.e., s is the root of the braid and thus it is clearly in $\mathbf{L}(t)$. Assume $\mathbf{T}_k(t) \subseteq \mathbf{L}(t)$ for some k and let $s = p(h) \in \mathbf{T}_{k+1}(t)$ for some $h \in \mathbf{D}^E_{\alpha\beta}$. Then, according to the definition of a braid, there is a term $r \in \mathbf{P}$ such that

$$p(r) \in \mathbf{T}_k(t), \quad h \in \mathbf{D}_k, \quad r \approx h \in E.$$

Now, by Lemma 4.8 we deduce that if $h \in \mathbf{D}_k$, then there is a term $g \in \mathbf{D}_0$ such that $h \in \mathbf{T}_k(g)$ and hence, by the inductive assumption, $h \in \mathbf{L}(g)$. Moreover, the equation $r \approx h \in E$ is a bridge between $\mathbf{T}(t)$ and $\mathbf{T}(g)$, which implies the existence of an edge from t to g in the graph and this edge is labelled by p. Consider the path in $\mathbf{G}$ obtained by adding to the path which starts in g and yields the term h a new first edge, namely, the edge p leading from t to g. Clearly this new path starts in t and yields the term $s = p(h)$. Thus $s \in \mathbf{L}(t)$. □

We have seen the general construction rules for the graph $\mathbf{G}$ and its connections with the set $\mathbf{D}^E_{\alpha\beta}$. We shall now see how to use the graph $\mathbf{G}$ to construct the set $\mathbf{D}^E_{\alpha\beta}$.

In the construction of $\mathbf{G}$ we start with a discrete graph i.e., with a graph with no edges. After having found a bridge, we mark the corresponding edge on the graph until no new edges are obtained. Thus our procedure reduces to finding bridges. If m is the number of terms in $\mathbf{D}_0$, then there are at most m^2 bridges to be found, since a graph on m vertices can have no more than m^2 edges.

What we get in the construction is in fact a sequence of graphs (each consecutive graph has one edge more than the previous). We shall define this sequence more precisely now.

Let $\mathbf{G}_0$ be the initial graph i.e., the graph with no edges and the set of vertices of which is $\mathbf{D}_0$. If we have a graph $\mathbf{G}_k$ and we find a new bridge, we get the graph $\mathbf{G}_{k+1}$ by adding to $\mathbf{G}_k$ the edge corresponding to this bridge. As we have noted above, this sequence is finite and it has at most $m^2 + 1$ elements.

Let $\mathbf{L}_k(t)$ be the set of all terms, which can be read from the graph $\mathbf{G}_k$ starting from the vertice t.

Definition 5.3. Let

$$\mathbf{L}(\mathbf{G}_k) = \bigcup_{t \in \mathbf{D}_0} \mathbf{L}_k(t).$$

Thus $\mathbf{L}(\mathbf{G}_k)$ is the set of all terms which can be read from the graph $\mathbf{G}_k$.

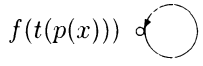
Theorem 5.4. *For any $k \in \{0, 1, \ldots, m^2\}$, if the graph $\mathbf{G}_k$ exists, then*

$$\mathbf{L}(\mathbf{G}_k) \subseteq \mathbf{D}^E_{\alpha\beta}.$$

P r o o f. We shall use Theorem 5.2 here. The set $\mathbf{L}(\mathbf{G}_k)$ is a union of the sets $\mathbf{L}_k(t)$ over all $t \in \mathbf{D}_0$, while $\mathbf{D}^E_{\alpha\beta}$ is a union of braids $\mathbf{T}(t)$ over all $t \in \mathbf{D}_0$, too. Thus it is sufficient to prove that for any $t \in \mathbf{D}_0$, $\mathbf{L}_k(t) \subseteq \mathbf{T}(t)$. This has been proved in the proof of Theorem 5.2. □

As we have stated before, we shall construct the set $\mathbf{D}^E_{\alpha\beta}$ using a sequence of graphs $\mathbf{G}_k$, obtained by a consecutive insertion of new edges corresponding to bridges. In the sequel we shall usually omit the subscript k when speaking of these graphs, since all the results presented in this section hold true for any k.

So, the construction of $\mathbf{G}$ consists in finding bridges. Any two vertices in the graph may be linked by an edge and, in particular, the graph may contain loops, e.g. when there is an equation in $\mathbf{E}$ relating a term with its main subterm.



The terms which can be read from this particular part of the graph are of the form

$$f(\ldots(f(t(p(x))))\ldots).$$

We shall first prove that given the graph $\mathbf{G}$, we can decide for any term t whether it is in $\mathbf{D}^E_{\alpha\beta}$ or not and we can do it in finite time. Moreover, we shall estimate the computational complexity of the procedures used and of the final algorithm. By *computational complexity* we mean here the order of the number of operations performed by the procedure or the algorithm in the worst case, depending on the number of initial data, which are the following:

terms α and β: these terms induce the initial set $\mathbf{D}_0$; let n be the number of terms in $\mathbf{D}_0$. By the definition of $\mathbf{D}^E_{\alpha\beta}$, this is just the number of occurrences of operation symbols in terms α and β.

the set E: let m be the number of equations in E and let l be the maximal number of occurrences of operation symbols in a term appearing in an equation in E. Algorithms scan the whole set, since any equation in it can induce an edge in the graph.

Observe that n is also the number of vertices of the graph, while n^2 is an upper bound on the total number of edges.

Definition 6.1. Let p be a vertice in the graph $\mathbf{G}$. A vertice t in $\mathbf{G}$ is a *neighbour* of p iff there is a (directed) edge from p to t. The set of all nieghbours of a vertice p will be denoted by $\mathbf{MS}(p)$.

Lemma 6.2. *Given the graph* $\mathbf{G}$ *of* $\mathbf{D}^E_{\alpha\beta}$, *it is effectively decidable for any* $t \in \mathbf{P}$ *whether* $t \in \mathbf{D}^E_{\alpha\beta}$.

Proof. A term t being in $\mathbf{D}^E_{\alpha\beta}$ is equivalent to t being readable from the graph $\mathbf{G}$. Let $|t| = k$ denote that a term t contains exactly k occurrences of operation symbols.

To read a term t from the graph of $\mathbf{D}^E_{\alpha\beta}$ we must find a path in the graph such that the labels of the consecutive edges in the path are equal to the corresponding occurrences of operation symbols of t, starting from the left (since reading a term from the graph we add consecutive labels on the right). Thus we are led to look for a path induced by the consecutive operation symbols of t.

Clearly, if the path is to start at a term p, then the first (external) operation symbol of p must be the same as the first operation symbol of t. Then some neighbour of p must have an external operation symbol equal to the second operation symbol in t, and so on. There are at most n edges starting in p and the total length of the path has to be k.

The simplest yet effective algorithm deciding whether there is such a path may have the following form (we assume that the occurrences of operation symbols in t are numbered by consecutive natural numbers, the most external symbol having index 1):

First we scan the whole graph, marking those vertices, which have the same external operation symbol as t. Each vertice can be either marked or unmarked—at each step of the algorithm. We start the algorithm with a set of marked vertices and at consecutive steps we determine new sets of such vertices.

Set $i := 2$ (i represents a variable, indicating consecutive operation symbols in t, starting from the left; its initial value is 2)

1. Search for all those vertices in the graph the external operation symbol of which is the same as the i -th operation symbol of t.
2. Search for all those vertices among those found in step 1 which are neighbours of marked vertices. If such vertices exist, they form the new set of marked vertices.
3. Set $i := i + 1$.
4. Repeat steps 1,2 and 3 while $i \leqslant k$. Finally, check whether there is an edge from a marked vertice to the variable occurring in t. If there is at least one such edge, then the term t can be read from the graph and therefore it is in $\mathbf{D}^E_{\alpha\beta}$

If no vertices are found in step 1 or 2 or if no edges are found in step 4, then the term t cannot be read from the graph $\mathbf{G}$ and the algorithm stops. After $k - 1$ iterations of the procedure, the value of the variable i will be k and the effect of step 1 will then be to check whether there are vertices the external operation symbol of

which is the same as the last (innermost) operation symbol in t. The performance of all the previous iterations yields a path consisting of exactly those operation symbols as the consecutive operation symbols in t; the last step yields the variable of t and thus the algorithm is completed. □

Lemma 6.3. *The complexity of reading a term t from the graph $\mathbf{G}$ is $\Theta(kn^2)$, where $|t| = k$.*

Proof. We shall compute the complexity of reading a term t from the graph $\mathbf{G}$ by estimating the cost of each step of the algorithm of lemma 6.2.

Before the first step we scan all the vertices of the graph, thus we must perform $\Theta(n)$ operations. In step 1 we must scan n vertices of the graph, thus the cost of this step is of the order n i.e., $\Theta(n)$. The cost of the second step is $\Theta(n^2)$, since we must check at most n^2 edges. Thus the total cost of steps 1,2 and 3 in one iteration is of the order of $n^2 + n$. We repeat the procedure $k - 1$ times, hence the cost of the whole algorithm is

$$\Theta((k-1)(n^2+n)) \approx \Theta(kn^2).$$

Thus the complexity of the algorithm is $\Theta(kn^2)$, where k is the number of occurrences of operation symbols and variables in t. □

Lemma 6.4. *For any braids $\mathbf{T}(p), \mathbf{T}(t) \subseteq \mathbf{D}^E_{\alpha\beta}$ it is effectively decidable on the graph $\mathbf{G}$ whether they have a common term, i.e., whether $\mathbf{T}(p) \cap \mathbf{T}(t) \neq \emptyset$.*

Proof. The lemma clearly holds true if the terms p and t are equal. If $p \neq t$, then both braids have a common term only if they have the same external operation symbols. Assume then that $p = f(r)$ and $t = f(s)$ for some operation symbol $f \in \mathbf{F}$ and some terms $r, s \in \mathbf{P}$. The following cases can occur:

1. The vertices $f(r)$ and $f(s)$ have a common neighbour in the graph $\mathbf{G}$.
2. The vertices $f(r)$ and $f(s)$ have neighbours with a common external symbol, say, q and then for $r = q(p_1)$ and $s = q(p_2)$ (for some terms p_1, p_2) we have again three possible cases to consider.
3. None of the above occurs and then the braids have no common term.

The consideration of these cases leads to a natural algorithm. It is based on a recursive scheme, which has no infinite loop. If the first or third case occurs, the algorithm stops. The problem is to estimate how many times can the second case occur. Looking for a common term for two braids $\mathbf{T}(p)$ and $\mathbf{T}(t)$ we can apply a modification of an algorithm for depth search in a graph. Thus we shall first introduce the original algorithm for depth search.

We shall use letters v, u to denote graph vertices, V will denote the set of all vertices of a graph, visit(v) represents an instruction, which the algorithm can perform at

each vertice of the graph, like e.g. comparing a value assigned to a vertice with some given value, NEW[v] represents an array with entries from the set $\{\mathbf{true}, \mathbf{false}\}$, where an entry indicates whether a vertice has been visited by the algorithm or not. At the beginning of the algorithm the whole array is initialized with all entries **true**, indicating that no vertice has been visited.
The algorithm:

1. begin
2. for $v \in V$ do NEW[v] := **true**; (initialization)
3. for $v \in V$ do
4. if NEW[v] then WG(v);
5. end;

where the procedure WG(v) has the following body:
procedure WG(v)

1. begin
2. visit(v); NEW[v] := **false**;
3. for $u \in$ {set of nieghbours of v} do
4. if NEW[u] then WG(u)
5. end; (the vertice v has been scanned)

This algorithm visits each vertice of the graph and each of them is visited exactly once. Looking for a vertice common to two braids $\mathbf{T}(p)$ and $\mathbf{T}(t)$ we shall follow the same algorithm, but we shall slightly modify some of its features:

1. Instead of vertices, we scan pairs of vertices with the same external operation symbol. This can be viewed as initializing the original algorithm simultaneously at two vertices, for which we are to find a common term. If the search for a common term has not been completed for a given pair of vertices, then we can go to a new pair of vertices which have the same external operation symbol. Observe that in the worst case we have to check n^2 neighbouring pairs
2. We do not scan all possible pairs with the same external operation symbol as in line 3 of our program above, but we start searching from one fixed pair of vertices: the one for which we want to find a common term.
3. the operation visit consists in checking whether we can stop the search at the pair currently scanned. For this purpose we have to check whether there exist edges going out of these vertices to a common vertice, i.e. whether the two vertices have a common neighbour. If this is so, then there is a term which can be read both from the vertice p and the vertice t.
4. As in the original algorithm, whenever we scan a pair of vertices for the first time, we record the fact in order to avoid visiting the same pair more than once (this could imply an infinite loop). Observe that we consider pairs of vertices

and not single vertices: it could well happen that one of the vertices or even both have been previously visited, but in different pairs.

The algorithm stops when during the search procedure we find a common neighbour for a pair of vertices and then the two braids have a common term. They do not have such a term, if we find a common neighbour for no pair of vertices before the algorithm stops.

After these informal considerations we can present a more formal form of the algorithm. Let us first introduce some handy notation:

V—the set of all vertices of the graph $\mathbf{G}$,

$\tilde{V}$—the set of all pairs of vertices which have the same external operation symbol and are not of the form (v, v), i.e.

$\tilde{V}=\{(v_1, v_2)\colon (v_1, v_2) \in V \times V, v_1 \neq v_2, v_1 = f(p_1), v_2 = f(p_2)$ for some operation symbol $f \in \mathbf{F}$ and some terms $p_1, p_2 \in \mathbf{P}\}$.

To abbreviate our notation we shall denote a pair (u_1, u_2) by $\tilde{u}$.

$Z(\tilde{u}) = \{(v_1, v_2)\colon v_1$ is a neighbour of u_1 in the graph $\mathbf{G}$, v_2 is a neighbour of u_2 in $\mathbf{G}$ and $(v_1, v_2) \in \tilde{V}\}$

The algorithm starts at a pair $\tilde{v}$:

1. begin
2. for $\tilde{w} \in \tilde{V}$ do NEW[$\tilde{v}$] := **true**; (initialization)
3. WG($\tilde{v}$);
4. end;

where the procedure WG($\tilde{u}$) has the following body:

procedure WG($\tilde{u}$)

1. begin
2. visit($\tilde{u}$); NEW[$\tilde{u}$] := **false**;
3. for $\tilde{w} \in Z(\tilde{u})$ do
4. if NEW[$\tilde{w}$] then WG($\tilde{w}$);
5. end; (the pair $\tilde{w}$ has been scanned)

The construction of the algorithm and its analogy to the original one imply that each pair will be visited exactly once—this is sufficient to guarantee that all pairs of edges going out from these vertices will be checked (if only the vertices have the same external operation symbols—and only such pairs of vertices are of our concern) and that a common term will be found, if it exists. Thus, if the braids $\mathbf{T}(p), \mathbf{T}(t) \in \mathbf{D}^E_{\alpha\beta}$ have a common term, then the algorithm will find it and the answer will be positive; otherwise the algorithm stops after performing the search and the answer is negative. □

Lemma 6.5. *The complexity of the search for a term common to two braids in* $\mathbf{G}$ *is* $\Theta(n^4)$.

Proof. To estimate the complexity of the problem we shall estimate the complexity of the algorithm introduced in Lemma 6.4. Observe first that the complexity of the original algorithm for depth search in a graph is $\Theta(n+m)$, where n is the number of vertices of the graph and m is the number of its edges.

Let us now estimate the complexity of our algorithm. The performance of the visit operation applied to a pair of vertices requires $\Theta(n^2)$ checks, since if there are n edges going out from each of the vertices, then there are n^2 pairs of edges which have to be checked. Such a check has to be performed at each visited pair of vertices, so the cost of checking the conditions of the algorithm is

$$n^2\Theta(n^2) \approx \Theta(n^4).$$

The complexity of our algorithm is not greater than the complexity of the original depth-search algorithm, i.e. the order of the sum of the number of vertices and the number of edges. In our algorithm the vertices are replaced by pairs of vertices with the same external operation symbols and there are at most n^2 such objects. The edges are now replaced by pairs of edges going out from vertices with the same external operation symbol and there are at most n^4 such pairs. Therefore the cost of the whole algorithm is

$$\Theta(n^4+n^2)+\Theta(n^4) \approx \Theta(n^4).$$

□

Observe that the proof (with natural modifications) remains valid if we consider partial braids, i.e. $\mathbf{T}_k(p)$ instead of $\mathbf{T}(p)$ and/or $\mathbf{T}_l(t)$ instead of $\mathbf{T}(t)$.

We shall now design algorithms for finding bridges between two arbitrary braids. For a fixed pair of braids we shall consider one by one each of Birkhoff rules to check whether one of them could lead to a bridge. We shall apply this procedure to all possible equations in E.

Let us enumerate all possible cases, which may lead to $p \approx t \in \mathbf{E}$.

1. $p = t$.
2. $p \approx t \in E$.
3. $t \approx p \in \mathbf{E}$.
4. $p' \approx t' \in \mathbf{E}$ and the terms p and t are obtained from p' and t' by replacing all occurrences of some variable by a fixed term.
5. $s \approx r \in \mathbf{E}$ and $p = f(s), t = f(r)$ for some $f \in \mathbf{F}$.

Symmetry being easy to verify, we can write E instead of $\mathbf{E}$ in 3. Also in 4. we can replace $\mathbf{E}$ by E, since substitution and extensionality permute and we can always apply rule 5. after all the others.

Observe that the equation $p \approx t$ could have been generated by substitution (rule 4.) only if there is an equation in E, the terms of which have the same external operation symbols as the terms p and t, correspondingly, or if E contains an equation of the form $h \approx x$—a projection—such that the term h has the same external operation symbol as either p or t.

Theorem 6.6. (Bridge Theorem.) *For any $\mathbf{T}_k(p), \mathbf{T}(t) \subseteq \mathbf{D}^E_{\alpha\beta}$ it is effectively decidable whether there exists a bridge between $\mathbf{T}_k(p)$ and $\mathbf{T}(t)$ in $\mathbf{E}$ for any fixed $k \in \omega$.*

P r o o f. To find a bridge between $\mathbf{T}_k(p)$ and $\mathbf{T}(t)$ we shall search $\mathbf{E}$ for equations between braids with roots in $\mathbf{MS}(p)$, i.e., braids of main subterms, and terms from the braid $\mathbf{T}(t)$. Thus our algorithm has to check all pairs of braids in which the root of the first braid lies in $\mathbf{MS}(p)$. Choose any term g in this set. Now we have to answer the following question: is there an equation in $\mathbf{E}$ such that one of its terms belongs to the braid $\mathbf{T}(g)$ and the second is in the braid $\mathbf{T}(t)$. We shall consider cases enumerated above. The algorithm will be introduced in the proof and, as we shall see, it will not depend on the fact whether the braid with root t is partial or not. The algorithm will be the same in both cases, but for convenience we shall use the notation $\mathbf{T}(g)$.

1. The question whether there exists a bridge between $\mathbf{T}_k(p)$ and $\mathbf{T}(t)$ in case 1. is answered in finite time by Lemma 6.3.
2. Case 2. consists in checking whether there exist terms in the braids $\mathbf{T}(g)$ and $\mathbf{T}(t)$ which occur in an equation in E.
3. Symmetry has been considered in the previous case already. Note that this case has to be checked together with cases 2., 4. and 5., i.e. whenever we check whether an equation has been generated by $s \approx r \in \mathbf{E}$, we must also check whether it is generated by $r \approx s$.

Thus we are left with two last cases. For convenience we shall distinguish two situations, depending on whether the terms g and t have the same external operation symbol or not.

I. The terms g and t have different external operation symbols.

In this case only 4. can occur. Assume first that there is no equation of the form $h \approx x$ in E, i.e. there is no projection. If $s \approx r \in E$ and s and r have the same external operation symbols as g and t, respectively, then $s \approx r$ is a good candidate to induce a bridge. If a bridge has been induced by this equation, then in $\mathbf{G}$ there

must be a sequence of edges starting in g such that the labels of these edges are the same as the operation symbols occuring in the term s. Let

$$f_1, f_2, \ldots, f_n$$

be the sequence of operation symbols occurring in s and—at the same time—the sequence of labels of a path in $\mathbf{G}$ starting in g. Thus in particular an edge labelled by f_1 goes out from the vertice g. Let

$$p_1, p_2, \ldots, p_m$$

be the sequence of symbols occurring in r taken, as above, in the proper order, starting with the external operation symbol. It is also the sequence of labels of edges forming a path starting at the vertice t. If the bridge has been induced by the equation $s \approx r \in E$, then both sequences must have been introduced into the graph by now. We shall now follow the paths corresponding to these two sequences of labels, one of them starting at g, the other starting at t, in order to check whether we can read from the graph two terms, the sequences of operation symbols of which—from the most external symbol up to some place in the sequence—are the same as the sequences of operation symbols in s and r, respectively. If so, then we check whether we can read a common term starting from the vertices reached after completing the paths labelled by $f_1, f_2, \ldots, f_n$ and $p_1, p_2, \ldots, p_m$. A positive answer means that a bridge occurred due to a substitution of a term for some variable in the equation $s \approx r \in \mathbf{E}$.

We shall use Lemma 6.2 to check whether we can read from $\mathbf{G}$ terms which have the same sequence of symbols as terms s and r. Thus we shall follow the paths labelled by $f_1, f_2, \ldots, f_n$ and $p_1, p_2, \ldots, p_m$, starting in g and t, correspondingly. We stop after exhausting both sequences. If we did complete both paths, then—if s and r have different variables—there exists a bridge between $\mathbf{T}_k(p)$ and $\mathbf{T}(t)$, since for different variables it is not important what terms can be read from this point. If, however, s and r have the same variable, then we must check whether a common term can be read from the vertices reached at this point. The existence of such a term means that it has been substituted for a variable in the equation, so a bridge exists. To find such a common term we can apply the algorithm of Lemma 6.4. We start the search for a common term at the vertices which have been reached after completing the paths labelled by $f_1, f_2, \ldots, f_n$ and $p_1, p_2, \ldots, p_m$.

If there is no projection in $\mathbf{E}$, then we are done with case **I**. Otherwise, a bridge can be induced by the equation $s \approx r$ where r or s is a variable only if the external operation symbol of the term in the projection is the same as the external operation symbol in g or t. If this is so, then the pocedure for checking whether a bridge has

been induced by the projection is analogous to the one described above with the only difference that the variable (one of the terms in the projection) has an empty sequence of operation symbols, i.e. it induces an empty path; hence we follow only the path induced by the second term and after completing it, we search again for a common term (which could have been substituted for the variable).

II. g and t have the same external operation symbol.

In this case a bridge can arise due to 4) or 5) and therefore we should first check whether case 4) has occurred, applying the previous algorithm (in **I**) without any changes: it could have happened that the equation is the result of substituting terms for some variables in an equation with the same external operation symbols on both sides. If no bridge was found while checking case 4), it remains to check whether a bridge could have been generated because of case 5). Assume first, that the vertices g and t have no single loops, i.e., there is no bridge between $\mathbf{T}(g)$ and $\mathbf{T}(g)$ nor between $\mathbf{T}(t)$ and $\mathbf{T}(t)$. With this assumption we have to look for equations between braids of main subterms of the braids $\mathbf{T}(g)$ and $\mathbf{T}(t)$, i.e., between braids with roots in $\mathbf{MS}(g)$ and $\mathbf{MS}(t)$. Thus we omit all occurrences of the external operation symbol in a possible equation and we check whether there is in $\mathbf{E}$ an equation without these symbols. If so, then also after adding the same operation symbol to both sides we get an equation in $\mathbf{E}$. This is in fact a backwards verification of Birkhoff's rule (iv). This check yields a recursive procedure, since looking for bridges between braids of main subterms requires considering all cases 1)–5) enumerated above. All possible pairs of braids of main subterms have to be checked, e.g. fixing a term $p \in \mathbf{MS}(g)$ we have to check whether there is a bridge between $\mathbf{T}(p)$ and any vertice in the set $\mathbf{MS}(t)$; this should be repeated for each term in $\mathbf{MS}(g)$.

During this search it can happen that two vertices have the same external operation symbol, i.e. a situation analoguous to that of Lemma 6.4. Therefore the same algorithm can be applied—the modified depth-search algorithm in a graph—with a new visiting operation.

As in the algorithm of Lemma 6.4, we scan all pairs of vertices with the same external operation symbols, starting with the vertices g and t. As we have seen before, each of the cases 1)–4) can occur for a given pair of vertices. Moreover, there exists a bridge between the braid of g and the braid of t iff one of the cases 1)–4) occurs for a pair of main subterms in the braids $\mathbf{T}(g)$ and $\mathbf{T}(t)$. Only when none of 1)–4) occurs neither for the braids $\mathbf{T}(g)$ and $\mathbf{T}(t)$ nor for any pair of braids of main subterms of these braids can we turn to another pair of vertices with the same external operation symbol. Thus the whole algorithm in case **II** will be identical to the algorithm of Lemma 6.4; however, the operation visit must check the cases 1)–4) not only for the pair of vertices visited, but also for all pairs of braids of main subterms of this pair.

In the worst case we shall have to scan n^2 pairs of vertices with the same external operation symbol, which induce n^4 pairs of braids of main subterms. Observe however, that there is no need to repeat the operation visit for all n^4 pairs. Indeed, if a pair of vertices p_1, p_2 is a pair of main subterms for two different pairs of braids (with equal external operation symbols), say g_1, g_2 and t_1, t_2, then if we state that none of the cases 1)–4) occurs for p_1, p_2 while visiting the pair g_1, g_2, then we need not check the pair p_1, p_2 again while visiting t_1 and t_2. Thus the operation visit must be performed just once for each pair of vertices in G. This shortcut can be solved in the algorithm by an additional table with n^2 entries, where each visit to a pair of vertices would be marked. One special case, when two braids with the same external operation symbol have single loops is considered below. Such a situation reduces to checking whether case 4) occurs, taking into account a larger number of equations in E. Thus we have a recursive algorithm again.

At each iteration we omit the external operation symbol of terms in vertices, i.e., in the possible equation we are looking for. Let us now consider what happens if there is a single loop in one of the braids, i.e., if there is a bridge from, say, $\mathbf{T}(g)$ to $\mathbf{T}(g)$. In this case the same vertice is simultaneously a braid of its main subterms and the whole algorithm remains unchanged. There is no danger of looping, since in the second vertice scanned we omit one symbol at each step and we always advance. The situation will be slightly different when both braids have a single loop. In this case we have to check by the algorithm of case **I** those equations in E in which one term only has the same external opeation symbol as the terms in our braids. This checking should be performed for the braid $\mathbf{T}(g)$ and all the braids of main subterms of the term t and viceversa, i.e., for all braids of main subterms of the term g and the braid $\mathbf{T}(t)$, scanning all the possible equations in $\mathbf{E}$. Due to this checking we will find e.g. the equation

$$f(f(f(p(x)))) \approx f(f(h(y)))$$

if the following equation was in E

$$f(p(x)) \approx h(y).$$

The external operation symbol here is f. If we omit all the occurrences of f here, i.e., if we went to check the braids of main subterms, then we would be checking whether the equation $p(x) \approx h(y)$ is in E. This equation, however, is not implied by the equation $f(p(x)) \approx h(y)$ and we may not be able to find a bridge between the braids considered. Only after this verification we can proceed with subsequent recursive calls of our algorithm for the braids of main subterms (in the example above this means omitting all the occurrences of f in both vertices). This is the only special case, which should be considered separately when two braids have the same external

operation symbol. It remains to prove that the algorithm yields an answer in finite time: that we shall never be induced to call the algorithm endlessly. The danger of such a looping can only occur in case **II**. Is it possible that after each calling of the procedure we find new barids of subterms with the same external operation symbols? The answer is: no. The argument is the same as in the proof of Lemma 6.4. □

Theorem 6.7. *The complexity of finding a bridge in the graph* $\mathbf{G}$ *is* $\Theta(2n^6 + 7kln^4 + 4mln^2)$.

Proof. We shall compute the complexity of the problem by estimating the consecutive steps of the algorithm introduced in Theorem 6.6.

1. The complexity of this step is $\Theta(n^4)$, the same as in Lemma 6.5.
2. By Lemma 6.4 the complexity of checking this case is $\Theta(4lmn^2)$. The reading of one term has complexity $\Theta(ln^2)$, since the longest term occurring in equations in E has length l and each equation requires 4 attempts to read a term from the graph: the equation itself has two terms and at each verification we also check the symmetric equation. The number of equations in E is m, so the cost $\Theta(4ln^2)$ of checking one equation has to multiplied by m. Thus the total complexity of this step is $\Theta(4mln^2)$.
3. It is clear that checking symmetry doubles the cost of computations. In cases 1. and 2. symmetry has been taken care of, while in the remaining cases we have to multiply the global cost by 2, thus taking into account symmetry.
4. The cost of the algorithm presented in **I** consists of the cost of reading two terms from the graph multiplied by the number of equations in the set E, which potentially may induce a bridge and of the cost of finding a common term for a given pair of braids. The cost will be the same for equations which are projections and for those which are not. The global cost of checking case 4) is therefore
$$\Theta(2kmn^2) + \Theta(n^4) \approx \Theta(n^4)$$
where k is the length of the longer term being read and m is the number of equations in the set E, $\Theta(2kn^2)$ is the cost of reading two terms of length k from the graph $\mathbf{G}$ and $\Theta(n^4)$ is the cost of checking whether two braids have a common term. These values are taken from the proofs of lemmas 6.3 and 6.5.
5. The cost of one visit operation is the sum of the costs 1-4 and thus is equal to
$$\Theta(4kln^2) + \Theta(n^4) + \Theta(n^4) \approx \Theta(2n^4 + 4kln^2).$$
In the worst case we shall have to visit all pairs of vertices, so all the visits will cost
$$n^2\Theta(2n^4 + 4kln^2) \approx \Theta(2n^6 + 4kln^4).$$

The cost of the algorithm itself (without the costs of the visit operation) is $\Theta(n^4)$, as in Lemma 6.4. Thus the total cost of checking case 5) is

$$\Theta(2n^6 + 4kln^4) + \Theta(n^4) \approx \Theta(2n^6 + 5kln^4).$$

The described above cases when one or two braids have single loops are impoprtant for the algorithm, but they do not influence its cost. In each case the order of computational complexity is the same, i.e.,

$$\Theta(2n^6 + 5kln^4).$$

We can now estimate the total cost of deciding whether there exists an equation inducing a new edge in the graph. It is

$$\Theta(2n^6 + 5kln^4) + \Theta(2n^4) + \Theta(4mln^2) \approx \Theta(2n^6 + 7kln^4 + 4mln^2),$$

which proves the theorem. □

7. Main Theorems

Theorem 7.1. *For any $\alpha, \beta \in \mathbf{P}$ and for any finite set of equations E the sequence $\mathbf{G}_k$ of graphs is finite and all its elements can be effectively constructed.*

Proof. Recall that the graph $\mathbf{G}_{k+1}$ is obtained from $\mathbf{G}_k$ by adding a new edge due to a newly found bridge. By Theorem 6.6 we can find a bridge in finite time. There are at most n^2 edges in a graph corresponding to a unary signature. The construction of the graph consists in choosing one pair of vertices and checking whether it can be linked by an edge. We must, however, perform more than n^2 verifications, since after finding a new bridge e.g. between $\mathbf{T}(p)$ and $\mathbf{T}(t)$, the set of terms belonging to $\mathbf{T}(p)$ becomes larger. Also the sets of terms in all other braids from which we can reach the vertice p, become larger. Thus, in order not to loose any edge we have to check—after finding a new bridge—the possibility of adding edges to and from all the vertices mentioned above. There is, however, a finite number of them and we shall be able to find all these edges in finite time. □

Theorem 7.2. *The complexity of constructing all the graphs $\mathbf{G}_k$ of the sequence is $\Theta(cn^{10})$, for some constant c.*

Proof. The complexity of finding a bridge in the graph $\mathbf{G}$ has been estimated in Theorem 6.6. Thus we need at most n^4 verifications to find all edges in the graph.

This estimation is very coarse, nut it shows, that even in such an "uneconomic" case the total cost of constructing the graph $\mathbf{G}$ is polynomial with respect to the dimension of input data. At least n^2 verifications have to be performed, for this is the possible number of all edges, and no more than n^4 such verifications are needed, as can be easily seen.

Hence the complexity of constructing the graph $\mathbf{G}$ is

$$n^4\Theta(2n^6 + 7kln^4 + 4mln^2) \approx \Theta(2n^{10} + 7kln^8 + 4mln^6).$$

This cost can be estimated as $\Theta(cn^{10})$ for some constant c. □

Let the last element in the sequence $\mathbf{G}_k$ of graphs be $\underline{\mathbf{G}}$.

Theorem 7.3. *For any $\alpha, \beta \in \mathbf{P}$ and for any finite set of equations E,*

$$\mathbf{L}(\underline{\mathbf{G}}) = \mathbf{D}^E_{\alpha\beta}.$$

Proof. The inclusion $\subseteq$ holds due to Theorem 5.4, since the graph $\underline{\mathbf{G}}$ is $\mathbf{G}_k$ for some $k \leqslant m^2 + 1$.

On the other hand, the inverse inclusion follows from Theorem 5.2, since the set $\mathbf{D}^E_{\alpha\beta}$ is the union of all braids with roots in $\mathbf{D}_0$, while $\mathbf{L}(\underline{\mathbf{G}})$ is the union of the sets $\mathbf{L}(t)$, again for all $t \in \mathbf{D}_0$. □

An equation $r \approx s$ will be called *an equation between braids* $\mathbf{T}(p)$ *and* $\mathbf{T}(q)$ for some $p, q \in \mathbf{P}$ iff r is an arbitrary term in the braid $\mathbf{T}(p)$ and s is an arbitrary term in $\mathbf{T}(q)$.

Remark 7.4. The verification of the rule (v′) will be based on the following property of braids: for any two terms p and q in a braid there exists for some n a sequence of terms $p_1, p_2, \ldots, p_n$ such that

$$p \approx p_1,\ p_1 \approx p_2,\ \ldots,\ p_{n-1} \approx p_n,\ p_n \approx q \in \mathbf{E}.$$

This property has been proved in Lemma 4.6. Due to this property we can operate in a more general setting: instead of looking for an equation between terms, we shall be looking for an equation between braids. This is an essetnial simplification, since there can be infinitely many terms in $\mathbf{D}^E_{\alpha\beta}$, but there is always a finite number of braids only (and it is the cardinality of the set $\mathbf{D}_0$).

We shall now describe the basic ideas underlying an algorithm for deciding whether a given equation has been generated by the rule of weak transitivity.

Let $t_1 \in \mathbf{T}(t)$ and $g_1 \in \mathbf{T}(g)$ and assume $t_1 \approx g_1 \in \mathbf{E}$; hence there is in $\mathbf{E}$ an equation between the braids $\mathbf{T}(t)$ and $\mathbf{T}(g)$. If $\alpha \in \mathbf{T}(t)$ and $\beta \in \mathbf{T}(g)$, then the following hold true:

1. for $t_1, \alpha \in \mathbf{T}(t)$ there exists for some n a sequence of terms $r_1, \ldots, r_n \in \mathbf{D}^E_{\alpha\beta}$ such that
$$\alpha \approx r_1,\ r_1 \approx r_2,\ \ldots,\ r_{n-1} \approx r_n,\ r_n \approx t_1 \in \mathbf{E}.$$
2. for $g_1, \beta \in \mathbf{T}(g)$ there exists for some m a sequence of terms $s_1, \ldots, s_m \in \mathbf{D}^E_{\alpha\beta}$ such that
$$g_1 \approx s_1,\ s_1 \approx s_2,\ \ldots,\ s_{m-1} \approx s_m,\ s_m \approx \beta \in \mathbf{E}.$$

Hence for the sequence

$$r_1,\ \ldots,\ r_n,\ t_1,\ g_1,\ s_1,\ \ldots,\ s_m$$

we have

$$\alpha \approx r_1,\ \ldots,\ r_n \approx t_1,\ t_1 \approx g_1,\ g_1 \approx s_1,\ \ldots,\ s_m \approx \beta \in \mathbf{E},$$

which imply

$$\alpha \approx \beta \in \mathbf{E}.$$

This means that for $\alpha \in \mathbf{T}(t)$ and $\beta \in \mathbf{T}(g)$, the existence of an equation between the braids $\mathbf{T}(t)$ and $\mathbf{T}(g)$ is sufficient to establish that $\alpha \approx \beta \in \mathbf{E}$. Moreover, this can also be deduced if there are some intermediate braids between $\mathbf{T}(t)$ and $\mathbf{T}(g)$, i.e., if for some $p_1, \ldots, p_n \in \mathbf{D}_0$ there are equations between $\mathbf{T}(t)$ and $\mathbf{T}(p_1)$, between $\mathbf{T}(p_1)$ and $\mathbf{T}(p_2)$ and so on, and between $\mathbf{T}(p_n)$ and $\mathbf{T}(g)$.

We can now describe the algorithm, which sums up our previous efforts and yields the main result of the paper.

Theorem 7.5. *For any $\alpha, \beta \in \mathbf{P}$ and for any finite set of equations E it is effectively decidable whether $E \models_w \alpha \approx \beta$ holds.*

Proof. We must consider the rules (i)–(iv) and (v′); this last rule can be checked after all others have been. The verification whether the equation $\alpha \approx \beta$ has been generated by one of the rules (i)–(iv) is algorithmically easy. Procedures checking these cases have been written in Pascal and they are used in a program constructing the set $\mathbf{D}^E_{\alpha\beta}$ from its definition (see the Appendix). The complexity of these algorithms is linear—of the order $\Theta(n)$. The verification whether the equation $\alpha \approx \beta$ has been generated by rule (v′) will use the following algorithm:

1. Construct the graph $\mathbf{G}$.

2. If $\beta \in \mathbf{T}(\alpha)$, then $\alpha \approx \beta \in \mathbf{E}$ and the algorithm stops.
3. We find all braids $\mathbf{T}(p_i) \subseteq \mathbf{D}^E_{\alpha\beta}$ such that there is in $\mathbf{E}$ an equation between the braids $\mathbf{T}(\alpha)$ and $\mathbf{T}(p_i)$, i.e., such that for some terms $t \in \mathbf{T}(\alpha)$, $g \in \mathbf{T}(p_i)$, the equation $t \approx g$ is in $\mathbf{E}$.
4. Among the braids $\mathbf{T}(p_i)$ we select those braids $\mathbf{T}(q_i)$, for which we have not checked whether $\beta \in \mathbf{T}(q_i)$.
5. for each i we check whether $\beta \in \mathbf{T}(q_i)$. The existence of such a braid implies that $\alpha \approx \beta \in \mathbf{E}$ and the algorithm stops.
6. If there is no braid containing β, we repeat the algorithm from 3. onwards, treating each consecutive $\mathbf{T}(q_i)$ as $\mathbf{T}(\alpha)$.

The algorithm stops when we find the term β in one of the braids and then we have $\alpha \approx \beta$ or when we find no unchecked braid and then $\alpha \approx \beta \notin \mathbf{E}$. We need not check twice one and the same braid: this would imply turning around in loops.

In step 2. the algorithm stops after establishing that $\beta \in \mathbf{T}(\alpha)$, since according to Remark 7.4 this information is sufficient to deduce that $\alpha \approx \beta$. If the algorithm did not stop at step 2., then in step 3. we find all the braids $\mathbf{T}(p_i)$ such that there is an equation between $\mathbf{T}(\alpha)$ and $\mathbf{T}(p_i)$ in $\mathbf{E}$. To end this we shall apply n times the algorithm of Theorem 6.6, since each vertice can be the root of an adequate braid. In step 5. we check whether the term β is in one of these braids. If the answer is positive, then repeating the argument of Remark 7.4 we establish that $\alpha \approx \beta \in \mathbf{E}$. If, however, we discover the term β in one of the braids $\mathbf{T}(p_i)$ after several iterations of the algorithm, then there exists a sequence of "intermediate" braids (see Remark 7.4), which still yields the conclusion that $\alpha \approx \beta \in \mathbf{E}$.

We shall make explicit now why each step of the algorithm is finite:

step 1: Theorem 7.1 states that the graph $\mathbf{G}$ can be constructed in finite time.

step 2: Here we need to read a term from a braid and the algorithm in lemma 6.2 shows that this can be done in finite time.

step 3: We can apply the algorithm of Theorem 6.6 to find the braids $\mathbf{T}(p_i)$, since we are in fact concerned with finding an equation between some terms in two braids. Thus Theorem 6.6 guarantees that this can be done in finite time.

step 4: Here we must verify the status: "checked" or "not checked" for a finite number of braids.

step 5: This step reduces to an iterated finite repetition of step 2., so again it can be done in finite time.

As can be seen, each step of the algorithm can be performed in finite time. The graph $\mathbf{G}$ has a finite number of vertices, so we have a finite number of braids to check. It can be easily verified that the algorithm (without step 1) will be performed not more than n times. □

Theorem 7.6. *The complexity of the deciding algorithm for weak equational theories is* $\Theta(dn^{10})$, *for some constant* d.

Proof. Let us again estimate the complexity of each step of the algorithm:
step 1: By Theorem 7.2, the cost of this step is $\Theta(cn^{10})$.
step 2: By lemma 6.3, the cost of this step is $\Theta(kn^2)$.
step 3: By Theorem 7.6, the cost of this step is $\Theta(2n^6 + 7kln^4 + 4mln^2)$.
step 4: The cost is $\Theta(n)$.
step 5: The cost is $\Theta(jkn^2)$ –the cost of step 2. applied to j braids for some finite j.

Thus the cost of all steps except step 1. is

$$\Theta(2n^6 + 7kln^4 + 4mln^2) + \Theta(kn^2) + \Theta(n) \leqslant \Theta(bn^6)$$

for some constant b. As we have seen, the algorithm (without step 1.) will run at most n times, since each vertice of the graph $\mathbf{G}$ is checked at most once. Thus the total cost of the algorithm is

$$n\Theta(bn^6) + \Theta(cn^{10}) \approx \Theta(dn^{10})$$

for some constant d. □

8. Arbitrary signatures

We shall now return to the general case, considering arbitrary signatures. To avoid repetitions we omit proofs, since modifications of the previous proofs to include non-unary signatures are natural and can be easily done by the reader.

If $t = f(p_1, \ldots, p_{\delta(f)})$ is a term, then the term p_i will be called its *i-th coordinate*.

The idea of the solution is essentially the same as for the unary case, we shall use the same structures and the same tools– adequately translated into a general language. Intuitively speaking, the main difference consists in the fact that whenever we do something with the main subterm of a unary term, we should repeat the action for each main subterm of a non-unary term, i.e., for each coordinate of the term. We shall again construct the set $\mathbf{D}^E_{\alpha\beta}$ using bridges and the graph $\underline{\mathbf{G}}$ and again the terms in the set $\mathbf{D}_0$ will be used as vertices in the graph.

The notions of *level*, *chain*, *braid* remain unchanged. A *bridge* between two braids will be an equation linking one of the main subterms of a term in the first braid and any term in the second. To make things more visible, we shall colour bridges. A bridge involving the first coordinate of a term will be coloured with colour b_1, the second coordinate—with colour b_2 etc.

In the unary case we only needed one bridge between two braids. Now we need only one bridge *of each colour*. Whenever we find a b_i-bridge between braids $\mathbf{T}(p)$ and $\mathbf{T}(t)$, we know, that the braid $\mathbf{T}(p)$ will duplicate all the terms in $\mathbf{T}(t)$ *on the corresponding coordinate*; a second bridge of the same colour brings no new information. Thus in the general case the number of bridges between $\mathbf{T}(p)$ and $\mathbf{T}(t)$ shall be equal to the arity of the external operation symbol in the term p. Whenever we find a new bridge, we add to the graph a new edge of the corresponding colour between roots of the braids involved.

Again, the graph $\underline{\mathbf{G}}$ contains all the information about the set $\mathbf{D}^E_{\alpha\beta}$. Terms will be read from the graph and we shall use the graph to check whether a term is in $\mathbf{D}^E_{\alpha\beta}$. The rules for reading terms from the graph remain the same, but now they should be interpreted as concerning each coordinate separately and that all the coordinates should be checked. It can be easily seen that this procedure will be recursively applied to each main subterm in each visited vertice of the graph. Thus reading a term yields a tree and not just a path, as was the case for a unary signature.

The number of all bridges in the graph $\underline{\mathbf{G}}$ is still constant and is equal to $n \cdot m$, where n is the number of all terms in $\mathbf{D}_0$ and m is the total number of all occurrences of main subterms in terms belonging to $\mathbf{D}_0$.

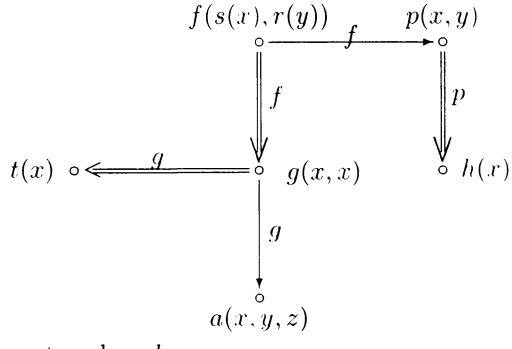
The braids are somewhat more complex, hence we shall classify the terms in braids:

Definition 8.1. We define the *rank* of a term as follows:

rank 0: A term p has rank 0 iff $p \in \mathbf{D}_0$.

rank $k+1$: A term p has rank $k+1$ iff it arises from a term t of rank k by extending by one step a tree used to get the term t.

Example 7. Observe the following fragment of a graph $\mathbf{G}_k$ for some k, some terms α, β and some set of equations E:



$\longrightarrow$ represents colour b_1

$\Longrightarrow$ represents colour b_2

Let us enumerate some terms of different ranks:

0 Terms of rank 0 are explicited in the graph.

1 Terms of rank 1:

$$f(p(x,y),r(y));\ f(s(x),g(x,x));\ f(p(x,y),g(x,x)$$

2 Terms of rank 2, e.g.:

$$f(p(x,h(x)),r(y));\ f(s(x),g(a(x,y,z),x));\ f(s(x),g(x,p(y)))$$

As before, we shall use $\underline{\mathbf{G}}$ to denote the graph obtained after finding all the bridges and drawing all the corresponding edges. The set of all terms which can be read from the graph $\underline{\mathbf{G}}$ will be denoted by $\mathbf{GT}$.

We shall now prove a theorem which shows that we can indeed use the graph $\underline{\mathbf{G}}$ to get all the terms in an arbitrary signature.

Theorem 8.2.

$$\mathbf{GT} = \mathbf{D}^E_{\alpha\beta}.$$

Proof. To prove $\supseteq$ we shall use induction on the structure of the set $\mathbf{D}^E_{\alpha\beta}$. Let $t \in \mathbf{D}_0$. Then clearly $t \in \mathbf{GT}$, since t is a vertice in the graph. Assume now $t = f(r_1,\ldots,r_{\delta(f)}) \in \mathbf{D}_{k+1}$. Then by the definition of the set $\mathbf{D}^E_{\alpha\beta}$ there exist terms $s_1,\ldots,s_{\delta(f)} \in \mathbf{P}$ such that

$$f(s_1,\ldots,s_{\delta(f)}),r_1,\ldots,r_{\delta(f)} \in \mathbf{D}_k$$

and $s_i \approx r_i \in \mathbf{E}$ for $i = 1,2,\ldots,\delta(f)$. By the inductive assumption we have

$$f(s_1,\ldots,s_{\delta(f)}),r_1,\ldots,r_{\delta(f)} \in \mathbf{GT}.$$

Now, the equations $s_i \approx r_i \in \mathbf{E}$ represent bridges in the graph. Let us have a closer look at them. The terms $r_1,\ldots,r_{\delta(f)}$ are main subterms of the term t, while the terms $s_1,\ldots,s_{\delta(f)}$ are some terms in the set $\mathbf{D}_k$ and therefore they are in some braids: let $s_i \in \mathbf{T}(s'_i)$ for $i = 1,2,\ldots,\delta(f)$. Hence the equations $s_i \approx r_i$ represent bridges between the braid $\mathbf{T}(t)$ and the braids $\mathbf{T}(s'_i)$. These bridges are represented by edges in the graph; going from the vertice t to each of the vertices s'_i and following the rules for reading terms from the graph, we first read the symbol f, since each edge leaving the vertice t is labelled by this symbol. Next, from the vertices $s'_1,\ldots,s'_{\delta(f)}$ we read the terms $s_1,\ldots,s_{\delta(f)}$, which are in the braids $\mathbf{T}(s'_1),\ldots,\mathbf{T}(s'_{\delta(f)})$ correspondingly. Thus we get the term t, which proves that $t \in \mathbf{GT}$.

The second inclusion will be proved by induction on the rank of a term $t \in \mathbf{GT}$. If the rank of t is 0, then $t \in \mathbf{D}_0$. Let now $t = f(r_1, \ldots, r_{\delta(f)}) \in \mathbf{GT}$ be of rank $k+1$. Consider the paths which led to the term t in the graph. They start at some vertice $p \in \mathbf{D}_0$, while the terms $r_1, \ldots, r_{\delta(f)}$ belong to some braids $\mathbf{T}(r'_1), \ldots, \mathbf{T}(r'_{\delta(f)})$ correspondingly. Reading the term t we get a tree of such paths. If in this tree we omit all the edges leaving the vertice p, we get subtrees yielding the main subterms of the term t i.e., the terms $r_1, \ldots, r_{\delta(f)}$. These terms have rank k and thus by inductive assumption belong to $\mathbf{D}^E_{\alpha\beta}$. The omitted edges represent briges between the braid $\mathbf{T}(p)$ and each of the braids $\mathbf{T}(r'_1), \ldots, \mathbf{T}(r'_{\delta(f)})$, so by the Duplication Theorem, the braid $\mathbf{T}(p)$ duplicates all these braids on the corresponding coordinates. Clearly the external operation symbol in p has to be f, so $t = f(r_1, \ldots, r_{\delta(f)}) \in \mathbf{D}^E_{\alpha\beta}$, which completes the proof of the Theorem. □

Now, is it possible to find a bridge between two braids in this general case, using the graph $\mathbf{G}$? The answer is positive: the algorithm for finding bridges in theorem 6.6—and Theorem 7.6—carry over without changes. Only the rules for reading terms from the graph are different, but they are natural generalizations of those for the unary case. Thus we can affirm that the problem of finding bridges and hence also the construction of the graph $\underline{\mathbf{G}}$ is solvable in finite time.

We can therefore formulate the following general theorem:

Theorem 8.3. *For any $\alpha, \beta \in \mathbf{P}$ and for any finite set of equations E it is effectively decidable whether $E \models_w \alpha \approx \beta$ holds.*

The whole proof together with the deciding algorithm is the same as for the unary case (see Theorem 7.5).

9. Complexity for arbitrary signatures

Observe first, that the computational complexity is most heavily influenced by the computational complexity of the algorithm for reading a term from the graph $\mathbf{G}$ (Lemma 6.3) and of the algorithm for checking whether two braids have a common term (Lemma 6.5). In the main theorem (the bridge theorem) we use in 1.2 and **I** algorithms previously defined and estimated, while the algorithm used in **II** is in its essence the same as the algorithm of Lemma 6.4 i.e., the algorithm searching for a term common to two braids. Thus, in order to estimate the complexity of all the algorithms presented above we can just estimate the complexity of these two basic algorithms.

Let j be the maximal arity of terms in $\mathbf{D}_0$, let k be, as before, the length of the term read. Recall that n is the number of all terms in $\mathbf{D}_0$.

Lemma 9.1. *For any signature, the cost of reading a term t from the graph $\mathbf{G}$ is $\Theta(kn^2)$ (i.e. the same as for a unary signature).*

Proof. The algorithm, as we have stated above, is basically the same as for a unary signature, suitably extended to all main subterms of the term t. The algorithm for the unary case was recursive with recursion depth equal to the number of operation symbols in t (i.e., k). Now we still have k operation symbols to be checked, but the checking for the main subterms is performed in the same step of the procedure. Thus the whole algorithm will have to perform k checkings again. Recursion is less deep here, though in turn each step of the procedure is more complex. We are still searching for k operation symbols with the corresponding edges, however. □

Lemma 9.2. *For any signature, the cost of finding a common term for two braids is $\Theta(j^2n^6)$.*

Proof. The algorithm solving this question is a natural extension of the algorithm for the unary case. Recall that for the latter we used an algorithm for searching the graph in depth, applied to all pairs of vertices with the same external operation symbol. The operation visit consisted in searching a common neighbour for a given pair. The problem is quite similar now. For each visited pair of terms we must check whether they have a common neighbour for each edge colour admissible for this operation symbol. Both terms of the visited pair have the same external operation symbol, so they also have the same arity and, consequently, the same colours of outgoing edges. To each pair of main subterms without common neighbour the recursion step is applied—namely, a search for a common neighbour is performed for each pair of terms (with the same external operation symbol) which are at one-edge distance from the given pair of terms. Finding a term common to two braids means that one and the same term can be read from the graph starting from two different vertices. In the worst case, this requires an application of the graph depth-search procedure to each visited pair of terms.

Let's estimate the cost of this algorithm. There are n^2 pairs of vertices and there are jn^2 edges (j and n are as above). Thus there are j^2n^4 pairs of edges. An application of our graph depth-search procedure to one edge colour and to one pair of vertices represents a cost of $\Theta(n^2 + n^4)$, just as for a unary signature. For each scanned pair we need to perform j such processes—one for each colour. The number of pairs of vertices being n^2, the cost of the whole algorithm is

$$jn^2\Theta(n^2 + n^4) \approx \Theta(jn^6).$$

The cost of the visit operation is $\Theta(jn^4)$ for each pair of vertices i.e., j times the cost in the unary case. This operation is performed at most n^2 times (for the same

reason as in the unary case), so the cost of all visit operations is $\Theta(j^2n^6)$. Therefore, the total cost of the algorithm is

$$\Theta(j^2n^6) + \Theta(jn^6) \approx \Theta(j^2n^6).$$

□

Knowing the cost of the algorithms, we can easily estimate the cost of constructing the whole graph $\mathbf{G}$ as well as the cost of the final deciding algorithm for the theory of weak equations. The cost of the algorithm of lemma 6.5 should now be replaced by the value obtained in the last lemma i.e., $\Theta(j^2n^6)$. It can be seen, then, that the cost of any of the other algorithms will increase by two orders of magnitude. Hence the total cost of the algorithm deciding whether $E \models_w p \approx q$ is of the order $\Theta(cn^{12})$ for some constant c.

All the algorithms above are only roughly described and feature one of the many possible ways of getting answers to questions we formulate. Also all the estimations of these algorithms are rather rough and in many cases excessive. It is important, however, that such algorithms exist and that their complexity is rather low and always polynomial. The reader will certainly be able to find other algorithms, perhaps more efficient, more subtle and much less expensive.

Appendix

For a better understanding of the problem I have written a program in Pascal, which yields the set $\mathbf{D}^E_{\alpha\beta}$ for arbitrary terms and an arbitrary finite set of equations introduced by the user. The signature is arbitrary. The construction of the set $\mathbf{D}^E_{\alpha\beta}$ follows its definition i.e., it is built level by level. The program is based on procedures for deciding for any pair $p, t \in \mathbf{P}$ of terms whether the equation $p \approx t$ is in $\mathbf{E}$, where, as before, $\mathbf{E}$ is the closure of the initial set of equations E with respect to Birkhoff rules (i)–(iv).

A copy of the program is available to anyone interested in the problem.

References

[1] *G. Grätzer*: Universal Algebra. van Nostrand Comp., 1968.

[2] *L. Rudak*: A completeness theorem for weak equational logic. Algebra Universalis *16* (1983), 331.

Author's address: 03-379, Warszawa, ul. Krasiczyńska 10, m. 148, Polska.