

Complex Analogies: Remarks on the Complexity of HDTP

Robert Robere¹ and Tarek Richard Besold²

¹ University of Toronto

robere@cs.toronto.edu

² Institute of Cognitive Science, University of Osnabrück

tbesold@uos.de

Abstract. After an introduction to Heuristic-Driven Theory Projection (HDTP) as framework for computational analogy-making, and a compact primer on parametrized complexity theory, we provide a complexity analysis of the key mechanisms underlying HDTP, together with a short discussion of and reflection on the obtained results. Amongst others, we show that restricted higher-order anti-unification as currently used in HDTP is $W[1]$ -hard (and thus NP-hard) already for quite simple cases. Also, we obtain $W[2]$ -hardness, and NP-completeness, for the original mechanism used for reducing second-order to first-order anti-unifications in the basic version of the HDTP system.

1 Introduction and Preliminaries

During the course of a day, we use different kinds of reasoning processes: We solve puzzles, play instruments, or discuss problems. Often we will find ourselves in situations in which we apply our knowledge of a familiar situation to the structurally similar novel one. Today it is undoubted that one of the basic elements of human cognition is the ability to see two a priori distinct domains as similar based on their shared relational structure (i.e., analogy-making). Key abilities within everyday life, such as communication, social interaction, tool use and the handling of previously unseen situations crucially rely on the use of analogy-based strategies and procedures. Relational matching, one of the key mechanisms underlying analogy-making, is also the basis of perception, language, learning, memory and thinking, i.e., the constituent elements of most conceptions of cognition [1].

Since the advent of computer systems, researchers in cognitive science and artificial intelligence have been trying to create computational models of analogy-making. This line of work has resulted in several different frameworks for computational analogical reasoning, starting with Reitman's ARGUS [2] and Evan's ANALOGY engine [3] and featuring systems as prominent as Hofstadter's Copycat [4] or the famous Structure-Mapping Engine (SME) [5] and MAC/FAC [6]. Whilst the latter two systems implement a version of Gentner's Structure-Mapping Theory (SMT) [7], more recently a different, generalization-based approach has been proposed: Heuristic-Driven Theory Projection (HDTP) [8,9]. Since its presentation, HDTP has continuously been refined and applied to different scenarios and domains, most prominently to mathematical reasoning [10] and conceptual blending in mathematics [11]. Still, except for a few accidental

observations, this far no formal analysis of the computational complexity of HDTP on a systematic basis has been provided. In this paper, we present several results concerning the computational complexity of HDTP and the corresponding anti-unification formalism for analogy-making.

The paper is structured as follows: Sect. 1.1 presents a short introduction to the Heuristic-Driven Theory Projection framework, followed by necessary preliminaries of parametrized complexity theory in Sect. 1.2. Then, Sect. 2 initially gives the definitions on the basic mechanisms of anti-unification, before Sect. 2.1 provides a complexity analysis of several forms of restricted higher-order anti-unification as applied in HDTP. Sect. 3 contains a complexity analysis of the use of equational theories in HDTP to simplify higher-order anti-unifications. Sect. 4 offers an interpretation of and reflection on the given complexity results, and Sect. 5 concludes the paper.

1.1 HDTP: Heuristic-Driven Theory Projection

The Heuristic-Driven Theory Projection (HDTP) framework [8] has been conceived as a mathematically sound framework for analogy-making. HDTP has been created for computing analogical relations and inferences for domains which are given in form of a many-sorted first-order logic representation [11]. Source and target of the analogy-making process are defined in terms of axiomatisations, i.e., given by a finite set of formulae. From there, HDTP tries to align pairs of formulae from the two domains by means of anti-unification (as firstly studied by Plotkin [12,13]): Anti-unification tries to solve the problem of generalizing terms in a meaningful way, yielding for each term an anti-instance, in which distinct subterms have been replaced by variables (which in turn would allow for a retrieval of the original terms by a substitution of the variables by appropriate subterms).¹ In its present form, HDTP extends Plotkin's classical first-order anti-unification to a restricted form of higher-order anti-unification, as mere first-order structures have shown to be too weak for the purpose of analogy-making [9].

Restricted higher-order anti-unification as presently used in HDTP was introduced in [14]. In order to restrain generalizations from becoming arbitrarily complex, a new notion of substitution is introduced. Classical first-order terms are extended by the introduction of variables which may take arguments (where classical first-order variables correspond to variables with arity 0), making a term either a first-order or a higher-order term. Then, anti-unification can be applied analogously to the original first-order case, yielding a generalization subsuming the specific terms. As already indicated by the naming, the class of substitutions which are applicable in HDTP is restricted to (compositions of) the following four cases: renamings, fixations, argument insertions, and permutations (cf. Sect. 2.1). Unfortunately, in the higher-order case, the least general generalization loses its uniqueness. Therefore, the current implementation of HDTP ranks generalizations according to a complexity measure on generalizations and chooses the least complex generalizations as preferred ones [9,15].

From a practical point of view, it is also necessary to anti-unify not only terms, but formulae. Therefore, HDTP extends the notion of generalization also to formulae

¹ In [12], Plotkin demonstrated that for a proper definition of generalization, for a given pair of terms there always is a generalization, and that there is exactly one least general generalization (up to renaming of variables).

by basically treating formulae in clause form and terms alike (as positive literals are structurally equal to function expressions, and complex clauses in normal form may be treated component wise). Furthermore, analogies do in general not only rely on an isolated pair of formulae from source and target, but on two sets of formulae. Here, heuristics are applied when iteratively selecting pairs of formulae to be generalized: Coherent mappings outmatch incoherent ones, i.e., mappings in which substitutions can be reused are preferred over isolated substitutions, as they are assumed to be better suited to induce the analogical relation. Once obtained, the generalized theory and the substitutions specify the analogical relation, and formulae of the source for which no correspondence in the target domain can be found may be transferred, constituting a process of analogical transfer between the domains.

The HDTP framework has successfully been tested in different application scenarios, and its use in several others has been proposed and theoretically grounded. [10] shows a way how HDTP can be applied to model analogical reasoning in mathematics by a case study on the inductive analogy-making process involved in establishing the fundamental concepts of arithmetic, [11] applies HDTP to conceptual blending in the mathematics domain by providing an account of a process by which different conceptualizations of number can be blended together to form new conceptualizations via recognition of common features, and judicious combination of distinctive ones. In [16], HDTP has been used in the context of solving geometric analogies. On the more theoretical side, [17] considers how the framework could fruitfully be applied to modeling human decision-making and rational behaviour, [18] elaborates on how to expand HDTP into a domain-independent framework for conceptual blending, and [19] provides considerations on the applicability of HDTP in a computational creativity context.

1.2 (Parameterized) Complexity Theory Preliminaries

As usual, P denotes the class of decision problems solvable in deterministic polynomial time, and NP denotes the class of decision problems solvable in non-deterministic polynomial time. We assume familiarity with the basic concepts of the theory of NP -completeness: NP -hardness, polynomial-time reductions, etc.

We briefly describe the basics of parameterized complexity theory. An instance of a *parameterized* decision problem $\mathcal{P}$ is a tuple (x, k) , where $x \in \{0, 1\}^*$ is a string describing the problem and $k \in \mathbb{Z}$, which is called the *parameter* of the problem (codifying other aspects of the problem besides n). We say a parameterized decision problem $\mathcal{P}$ is *fixed parameter tractable*, written $\mathcal{P} \in FPT$, if it is solvable in time bounded by $f(k) \cdot |x|^{O(1)}$, where $f(k)$ is some computable function of the parameters and $|x|^{O(1)}$ denotes a polynomial of the length of the input. As suggested by notation, we let FPT denote the class of all fixed parameter tractable problems.

Given two parameterized problems $\mathcal{P}, \mathcal{Q}$, a *parameterized reduction* is a function ϕ from $\mathcal{P}$ to $\mathcal{Q}$ such that the following holds, for an instance $(x, k) \in \mathcal{P}$:

1. $\phi(x)$ is computable in time $f(k) \cdot |x|^{O(1)}$, where f is a computable function of the parameter,
2. $x \in \mathcal{P}$ iff $\phi(x) \in \mathcal{Q}$, and
3. If k' is the parameter of $\phi(x)$, then $k' = g(k)$ for some function g .

Let the *weft* of a boolean circuit (containing only NOT gates, small AND and OR gates of fan-in ≤ 2 and large AND or OR gates of arbitrary finite fan-in) be the maximum number of large gates on any path from an input to the output, and let the *depth* be the maximum number of all gates on a path. Let $C[t, d]$ be the set of all circuits of weft at most t and depth at most d . Finally, define the (*Hamming*) *weight* of an assignment of truth values to the input variables of the circuit as the number of variables set to 1.

Problem 1. Weighted Circuit Satisfiability $[t, d]$

Input: A circuit C of depth d and weft t , a natural $k \in \mathbb{N}$.

Problem: Is there a satisfying assignment to C with weight k ?

We say a parameterized problem $\mathcal{P}$ is in $W[i]$ if it is reducible by a parameterized reduction to Weighted Circuit Satisfiability $[i, d]$ for some constant d , and is $W[i]$ -hard if every problem in $W[i]$ is reducible to $\mathcal{P}$ under a parameterized reduction. The assumption that $W[1] \neq \text{FPT}$ can be seen as analogous to the assumption that $P \neq NP$ in classical complexity, although it is indeed conjectured that $W[i] \subset W[j]$ for any $i < j$. For a more in-depth introduction to parameterized complexity, see for instance [20,21].

2 Anti-unification: Initial Considerations

We begin our considerations on the complexity of anti-unification with some requisite definitions. HDTP uses many-sorted term algebras to define the input conceptual domains. A term algebra requires two ingredients: a *signature* and a set of *variables*.

Definition 1. A many-sorted signature $\Sigma = \langle \text{Sort}, \text{Func} \rangle$ is a tuple containing a finite set *Sort* of sorts, and a finite set *Func* of function symbols. An n -ary function symbol $f \in \text{Func}$ is specified by $f : s_1 \times s_2 \times \dots \times s_n \rightarrow s$, where $s, s_1, \dots, s_n \in \text{Sort}$. We will consider function symbols of any non-negative arity, and we will use 0-ary function symbols to represent constants.

Definition 2. Let $\Sigma = \langle \text{Sort}, \text{Func} \rangle$ be a many-sorted signature, and let $\mathcal{V} = \{x_1 : s_1, x_2 : s_2, \dots\}$ be an infinite set of sorted variables, where the sorts are chosen from *Sort*. Associated with each variable $x_i : s_i$ is an arity, analogous to the arity of function symbols above. For any $i \geq 0$, we let $\mathcal{V}_i$ be the variables of arity i . The set $\text{Term}(\Sigma, \mathcal{V})$ and the function $\text{sort} : \text{Term}(\Sigma, \mathcal{V}) \rightarrow \text{Sort}$ are defined inductively as follows:

1. If $x : s \in \mathcal{V}$, then $x \in \text{Term}(\Sigma, \mathcal{V})$ and $\text{sort}(x) = s$.
2. If $f : s_1 \times s_2 \times \dots \times s_n \rightarrow s$ is a function symbol in Σ , and $t_1, \dots, t_n \in \text{Term}(\Sigma, \mathcal{V})$ with $\text{sort}(t_i) = s_i$ for each i , then $f(t_1, \dots, t_n) \in \text{Term}(\Sigma, \mathcal{V})$ with $\text{sort}(f(t_1, \dots, t_n)) = s$.

We refer to the structure $\langle \text{Term}(\Sigma, \mathcal{V}), \text{sort} \rangle$ as a term algebra, often suppressing *sort*. If $t \in \text{Term}(\Sigma, \mathcal{V})$ then we define $|t|$ to be the number of variables appearing in t .

From here on out, we will fix a term algebra $\text{Term}(\Sigma, \mathcal{V})$. In the next sections we will be considering several different forms of restricted higher-order anti-unification. First we introduce the *term substitutions* and *generalizations* allowed in first-order anti-unification.

Definition 3. A term substitution is a partial function $\sigma : \mathcal{V}_0 \rightarrow \text{Term}(\Sigma, \mathcal{V})$ mapping 0-ary variables to terms. An application of a term substitution σ on a term is defined inductively by:

1. $\text{apply}(x, \sigma) = \begin{cases} t & x \rightarrow t \in \sigma \\ x & \text{otherwise.} \end{cases}$
2. $\text{apply}(f(t_1, \dots, t_n), \sigma) = f(\text{apply}(t_1, \sigma), \dots, \text{apply}(t_n, \sigma)).$

Given two terms t, t' and a substitution σ such that $\text{apply}(t, \sigma) = t'$, then we call t' an instance of t and t an anti-instance of t' . We will often shorten $\text{apply}(t, \sigma) = t'$ to $t \xrightarrow{\sigma} t'$, or $t \rightarrow t'$ if the substitution is clear from context.

Let f, g be terms from a term algebra $\text{Term}(\Sigma, \mathcal{V})$. A generalization of f and g is a triple $\langle h, \sigma, \tau \rangle$ where $h \in \text{Term}(\Sigma, \mathcal{V})$ and σ, τ are substitutions such that $h \xrightarrow{\sigma} f$ and $h \xrightarrow{\tau} g$. The generalization $\langle h, \sigma, \tau \rangle$ is called the least general generalization (lgg) if for any generalization $\langle h', \sigma', \tau' \rangle$ of f, g , there exists a substitution ϕ such that $h' \xrightarrow{\phi} h$.

As mentioned earlier, the lgg is *unique* when considering only term substitutions. We can thus state the first-order anti-unification problem as follows:

Problem 2. First Order Anti-Unification

Input: Two terms f, g from a term algebra $\text{Term}(\Sigma, \mathcal{V})$

Problem: Find the least general generalization $\langle h, \sigma, \tau \rangle$ of f and g .

While this simple anti-unification problem has many efficient algorithms (see, for example, [12,22]), we will recall one algorithm which will form the basis of an efficient method for one of the forms of restricted higher-order anti-unification we consider. The algorithm will not operate on the terms themselves, per say, but rather rooted trees which can be naturally associated with each term.

Definition 4. Let f be a term in $\text{Term}(\Sigma, \mathcal{V})$. The tree form of f , denoted T_f , is a rooted labelled tree defined inductively as follows.

If f is a 0-ary (constant) term, then T_f consists of a single node r , labelled with $(f, 0)$. If f is an n -ary term, denote it as $f(t_1, \dots, t_n)$. Then the root of T_f is a node r , labelled with (f, n) , which has n ordered children $c_1, \dots, c_n$. The child c_i is the root of T_{t_i} , the tree form of the term t_i .

Algorithm 1. FO-AntiUnify. **Input:** The tree forms T_f, T_g of two terms f, g with respective roots r_f, r_g . **Output:** An anti-unifier h of f and g , with two substitutions σ, τ .

Let (f, n) be the label of r_f and (g, m) be the label of r_g .

if f or g is a 0-ary term or $n \neq m$ **then return** $h = X, \sigma = \{X \mapsto f\}, \tau = \{X \mapsto g\}$
end if

Let h be the empty term, and σ, τ be empty substitutions.

Let $a_1, \dots, a_n$ be the children of r_f , and $b_1, \dots, b_n$ be the children of r_g .

for $i = 1, \dots, n$ **do** $\langle y, \alpha, \beta \rangle = \text{FO-AntiUnify}(a_i, b_i)$; Merge $\langle y, \alpha, \beta \rangle$ with $\langle h, \sigma, \tau \rangle$.

end for; return $\langle h, \sigma, \tau \rangle$.

This algorithm performs a depth-first search of the two input trees, replacing subtrees with variables if they do not match structurally. It plainly runs in time linear in the size of the two input terms. We will return to this algorithm shortly, as it also efficiently solves a slightly more general anti-unification problem.

2.1 Restricted Higher-Order Anti-unification

In [14], a restricted form of higher-order anti-unification was presented for use in HDTP, defined as any composition of a certain set of unit substitutions.

Definition 5. *The following are the types of unit substitutions allowed in restricted higher-order anti-unification.*

1. A renaming $\rho(F, F')$ replaces a variable $F \in \mathcal{V}_n$ with another variable $F' \in \mathcal{V}_n$:

$$F(t_1, \dots, t_n) \xrightarrow{\rho(F, F')} F'(t_1, \dots, t_n).$$
2. A fixation $\phi(F, f)$ replaces a variable $F \in \mathcal{V}_n$ with a function symbol $f \in \mathcal{C}_n$:

$$F(t_1, \dots, t_n) \xrightarrow{\phi(F, f)} f(t_1, \dots, t_n).$$
3. An argument insertion $\imath(F, F', V, i)$ is defined as follows, where $F \in \mathcal{V}_n, F' \in \mathcal{V}_{n-k+1}, V \in \mathcal{V}_k, i \in [n]$:

$$F(t_1, \dots, t_n) \xrightarrow{\imath(F, F', V, i)} F'(t_1, \dots, t_{i-1}, V(t_i, \dots, t_{i+k}), t_{i+k+1}, \dots, t_n).$$

It “wraps” k of the subterms in a term using a k -ary variable, or can be used to insert a 0-ary variable.
4. A permutation $\pi(F, \tau)$ rearranges the arguments of a term, with $F \in \mathcal{V}_n, \tau : [n] \rightarrow [n]$ a bijection:

$$F(t_1, \dots, t_n) \xrightarrow{\pi(F, \tau)} F(t_{\pi(1)}, \dots, t_{\pi(n)}).$$

A restricted substitution is a substitution which results from the composition of any sequence of unit substitutions.

Clearly, restricted substitutions are strictly more general than term substitutions. By considering different combinations of restricted substitutions we can define several different forms of higher-order anti-unification. Unfortunately, the lgg is no longer necessarily unique [14], and so we will instead consider decision version of the problems parameterized by the number of substitutions, variables, and types of variables used.

Problem 3. F Anti-Unification

Input: Two terms f, g , and a natural $k \in \mathbb{N}$

Problem: Is there an anti-unifier h , containing at least k variables, using only renamings and fixations?

Problem 4. FP Anti-Unification

Input: Two terms f, g , and naturals $l, m, p \in \mathbb{N}$.

Problem: Is there an anti-unifier h , containing at least l 0-ary variables and at least m higher arity variables, and two substitutions σ, τ using only renamings, fixations, and at most p permutations such that $h \xrightarrow{\sigma} f$ and $h \xrightarrow{\tau} g$?

Problem 5. FPA Anti-Unification**Input:** Two terms f, g and naturals $l, m, p, a \in \mathbb{N}$.**Problem:** Is there an anti-unifier h , containing at least l 0-ary variables, at least m higher arity variables, and two substitutions σ, τ using renamings, fixations, at most p permutations, and at most a argument insertions such that $h \xrightarrow{\sigma} f$ and $h \xrightarrow{\tau} g$?

We summarize our (parameterized) complexity-theoretic results of higher-order anti-unification in the following theorem, which will be our next focus.

Theorem 1

1. *F Anti-Unification is solvable in polynomial time.*
2. *FP Anti-Unification is NP-complete and W[1]-hard w.r.t. parameter set $\{h, p\}$.*
3. *Let r be the maximum arity and s be the maximum number of subterms of the input terms. Then FP Anti-Unification is in FPT w.r.t. parameter set $\{r, p\}$.*
4. *FPA Anti-Unification is NP-complete and W[1]-hard w.r.t. parameter set $\{m, p, a\}$.*

F Anti-Unification. The F Anti-Unification problem can easily be seen to be solvable in linear time by a simple modification of Algorithm 1. When we consider the roots of two subtrees labelled with f, g , if the arities of the terms (i.e. the number of children) match (say, they are both n) we can add a new fixation using an n -ary variable. Moreover, up to renaming, this implies that when considering the F Anti-Unification problem the least general generalization is unique, just like First-Order Anti-Unification.

FP Anti-Unification

Theorem 2. *FP Anti-Unification is W[1]-hard w.r.t. parameter set $\{h, p\}$, and NP-complete.*

Proof. FP Anti-Unification is clearly in NP, by just non-deterministically guessing a generalization $\langle h, \sigma, \tau \rangle$ and applying the substitutions σ and τ to h . We will show W[1]-hardness and NP-completeness by reduction from the well-known W[1]-hard problem k -Clique.

Problem 6. Clique**Input:** A graph G and a natural $k \in \mathbb{N}$ **Problem:** Does G contain a clique, that is a complete induced subgraph, of size k ?

The theorem follows by the well-known reduction between Subgraph Isomorphism and Clique. Let G, k be an instance of Clique, and let $E = \{e_i = (u_i, v_i)\}_{i=1}^e$ be the set of edges in G . Set $n = k(k+1)/2$, the number of edges in a k -clique. Define the term g using G as follows. We will begin with an e -ary term gr , which contains terms $e_i(u_i, v_i)$, one for each edge in E .

We define f in the same way using K_k , the complete graph on k vertices. That is we start with the term gr , which contains a term $a_i(b_i, c_i)$ for each edge (b_i, c_i) in K_k . However, we will pad f using $m - n$ ternary terms $t_i(x_i, y_i, z_i)$, with unique symbols used for each term. Therefore, both terms will look like so:

$$g = gr(e_1(u_1, v_1), \dots, e_e(u_e, v_e))$$

$$f = gr(a_1(b_1, c_1), \dots, a_n(b_n, c_n), t_1(x_1, y_1, z_1), \dots, t_{e-n}(x_{e-n}, y_{e-n}, z_{e-n}))$$

Set $l = e - n + k, m = k, p = 1$.

Suppose that G contains a k -clique, denote $P \subseteq G$. Then clearly there exists an isomorphism ϕ from K_k to P , and so there exists a bijection between the edges of P and K_k . Using a single permutation, if necessary, move the edges in the terms f, g to occupy the first k arguments of gr , lining up the edges that are in correspondence under ϕ . We define the anti-unifier h of f and g like so:

$$h = gr(E_1(U_1, V_1), \dots, E_n(U_N, V_N), X_1, \dots, X_{m-n}),$$

using k 0-ary variables (corresponding to the *vertices*) for the terms inside the E_i variables, and using the E_i variables (corresponding to the *edges*), and define the fixation substitutions accordingly. Finally, use $n - k$ 0-ary variables to anti-unify the ternary terms at the end of f with the edges of g not occurring in the clique.

The reverse direction is simple, but technically tedious. One simply has to show that the term h described above is the only anti-unifier of f and g satisfying the values of the parameters (up to a renaming of the variables and a permutation of the subterms of gr). Once this is shown, it is simple to pull out a k -Clique in G , using the corresponding edge terms $E_i(U_i, V_i)$ occurring in the anti-unifier.

It seems that determining how to use permutations to order the subterms of f and g to maximize structural similarity is a quite difficult problem since even allowing *single* permutation implies computational hardness. On the tractability side, we can get a simple FPT result from restricting the arity and number of subterms of f and g .

Proposition 1. *FP Anti-Unification is in FPT w.r.t. parameter set $\{s, p, r\}$, where r is maximum arity and s is the maximum number of subterms of the input terms.*

Proof. Clearly, there are at most $\binom{s}{p}$ ways to apply permutations in any FP Anti Unification of f and g , and there will be at most $r!$ ways to apply each permutation. Running the algorithm for F Anti-Unification on each possible set of subterms gives an algorithm which runs in time at most $O(\binom{s}{p} r! \cdot n)$, where n is the number of symbols in the input.

FPA Anti-Unification We can get a parameterized hardness result for FPA Anti-Unification as a corollary of the above proof.

Theorem 3. *FPA Higher-Order Anti-Unification is $W[1]$ -hard with respect to parameter set $\{m, p, a\}$, and NP-complete.*

Proof. This result follows from the reduction above, where we can additionally set $a = 0$.

3 Equational Theories and HDTP

HDTP uses equational theories, which are conjunctions of first-order formulae with equality over a term algebra, to describe the input analogical domains. In certain cases, we can reduce a higher-order anti-unification to a first-order anti-unification by introducing new terms constructed from the terms in input theories. In this section we will show that finding such an expansion of the input theories is computationally intractable. We begin with some definitions.

Definition 6. Let $\text{Term}(\Sigma, \mathcal{V})$ be a term algebra. An equational theory E is a conjunction of statements of the form $t_1 = t_2$, where t_1, t_2 are first order formulae over terms from $\text{Term}(\Sigma, \mathcal{V})$. Given an equational theory E and an equation $t_1 = t_2$, we say that E implies $t_1 = t_2$, written $E \vdash t_1 = t_2$, if $t_1 = t_2$ can be proven from the input formulae, using any complete and sound proof system.

Given a term t we denote the set of all subterms of t as $st(t)$ and the set of variables in t as $var(t)$.

Definition 7. Let X be a set of terms. A set $\{t_1, \dots, t_n\} \subseteq X$ is called admissible relative to X if $\bigcup_{i=1}^n var(t_i) = \bigcup_{t \in X} var(t)$.

We can simplify some higher-order anti-unifications to first-order anti-unifications by introducing subterms built from the admissible sequences from the terms (for a description of how this reduces higher-order anti-unifications to first-order, see [8]).

Definition 8. Let $\text{Term}(\Sigma, \mathcal{V})$ be a term algebra, let E be an equational theory defined w.r.t. this algebra, and let $f(t_1, \dots, t_n) \in \text{Term}(\Sigma, \mathcal{V})$ be a term. An expansion of Σ and E relative to $f(t_1, \dots, t_n)$ is constructed as follows: for each admissible sequence $\{u_1, \dots, u_k\}$ relative to $st(f(t_1, \dots, t_n))$, a new function symbol h and an equation $h(u_1, \dots, u_k) = f(t_1, \dots, t_n)$ are added to Σ and E , respectively.

As shown in [8], the expanded equational theory can be used to reduce a selected class of higher-order anti-unifications to first-order anti-unifications. Now, confronted with the computational hardness results from Sect. 2.1, a naive attempt at dealing with the high complexity of higher-order anti-unification might be based on this seemingly simpler approach. Unfortunately, intuition here goes astray, and finding an admissible subset is actually quite a difficult problem by itself.

Problem 7. Function Admissible-Sequence

Input: A term $f(t_1, t_2, \dots, t_n) \in \text{Term}(\Sigma, \mathcal{V})$, a natural $k < |st(f(t_1, t_2, \dots, t_n))|$

Problem: Is there a set $X \subseteq st(f(t_1, t_2, \dots, t_n))$ such that $|X| \leq k$ and X is admissible relative to $st(f(t_1, t_2, \dots, t_n))$?

Theorem 4. Function Admissible-Sequence is $W[2]$ -Hard (and NP-Complete) w.r.t. parameter k .

Proof. Function Admissible Sequence is clearly in NP, by guessing a set $X \subseteq st(f(t_1, t_2, \dots, t_n))$, and checking in polynomial time if $\bigcup_{i=1}^n var(t'_i) = \bigcup_{t \in X} var(f(t_1, t_2, \dots, t_n))$. We will perform a reduction from the $W[2]$ -Hard problem Set Cover [23]:

Problem 8. Set Cover

Input: A finite family of sets $S = \{S_1, \dots, S_n\}$, a natural $k \in \mathbb{N}$.

Problem: Is there a subset $R \subseteq S$ with $|R| \leq k$ whose union is all elements in the union of S ?

Let $S = \{S_1, \dots, S_n\}$, be a family of sets and k' be a natural. Then it's straightforward to create a term $f(t_1, \dots, t_n)$, where each subterm t_i is a $|S_i|$ -ary term containing the elements of S_i as 0-ary terms. Finish by setting $k = k'$ in the definition of Function Admissible Sequence. In any solution of Set Cover or Function Admissible Sequence, the sets R and X are in direct correspondence with each other.

4 Interpretation of the Results

We want to provide some thoughts on the impact and the consequences of the complexity results from the previous sections, putting the obtained insights into a (cognitive) AI context. Presented with the complexity theoretic results, a critic may wonder why these results even matter for HDTP, for which undecidability follows quite straightforwardly from the undecidability of first-order logic. However it is important to note that HDTP is naturally split into two mechanisms: one which performs analogical matching of input theories, and one which performs re-representation of the input theories by deduction in first-order logic. It is the re-representational mechanism which is undecidable, where our above results apply solely to the analogical matching mechanism.

We focus on two results directly affecting HDTP. The $W[2]$ -hardness of the function admissible-sequence problem (cf. Theorem 4) deserves special attention, as this clearly shows that problems can already arise when only treating with reductions from higher-order to first-order anti-unification. Here it should be kept in mind that this was HDTP's original mechanism for anti-unification, and thus was frequently used on a basic level when, in order to reduce a higher-order anti-unification to a first order anti-unification, expanding the theory by including first-order projections of any higher-order terms. The result showing that FP higher-order anti-unification is $W[1]$ -hard (cf. Theorem 2) gives a hint at the difficulty introduced by the operations admissible within the restricted higher-order anti-unification on the complexity of the analogy-making process. Indeed, the only way that FP anti-unification can restructure the order of the terms is by argument permutations, and our results show that even allowing a *single* permutation is enough to imply computational hardness. If we contrast this result against the polynomial-time algorithm for F anti-unification, we have evidence that even a slight ability to restructure the input terms makes higher-order anti-unification a difficult problem to solve.

We now want to also present some remarks from a more integrated perspective. Mostly originating from within cognitive science, the recognition that human minds or brains are finite systems with limited resources for computation has led some researchers to advance the thesis that human cognitive capacities are constrained by computational tractability. This *Tractable Cognition Thesis* [24], if true, amongst others can serve researchers by limiting and narrowing down the space of computational-level theories of cognition. But this thesis can also have ramifications for work in AI, namely for the branch following the original artificial general intelligence (AGI) tradition in trying to create a system with human-like intelligence capacities. Adapting the *Tractable Cognition Thesis* to the new setting, converting it into a *Tractable AGI Thesis*, it seems recommendable to demand for computer models of artificial general intelligence to also be of the tractable type (i.e. polynomially-solvable) – as also all of the currently available computing systems are ultimately finite systems with limited resources, in this respect being very similar to the aforementioned human minds or brains.

Taking this stance, and additionally assuming that $P \neq NP$ (and $FPT \neq W[1]$) holds, the classical and parameterized hardness results presented in this paper cast a shadow on the suitability of the HDTP framework in its present state as basis for a general model for high-level cognitive capacities or a general cognitive architecture. Still, it should be

noticed that the mere fact that HDTP in its present state is basically intractable does not mean that future versions cannot be made tractable: One of the main questions for future theoretical research in relation to HDTP will have to address the question of how HDTP's version of computing generalizations via restricted higher-order anti-unification can be further constrained in a meaningful way as to obtain maximal expressivity and applicability whilst still staying within the domain of polynomial solvability. Also, more parametrized analysis will be needed, showing which are the factors that really impact complexity, and which are aspects of a problem that are not really harmful. Secondly, even when being intractable, HDTP can still be put to good use, serving as a testbed for possible uses of analogy-making engines. Coming back to the AGI setting, although it might not be held likely that an intractable system will grow into a general architecture providing artificial intelligence at a human level, it can still be used for exploring the possibilities and limitations of theories and paradigms incorporating analogy as a cognitive function (as, e.g., in the context of the already mentioned cognitive capacities conceptual blending, creativity, and rationality), and thus finally also contribute on a theoretical level to the development of a generally intelligent system. Even though HDTP in its current form might not scale to applications to real-world problems under real-world constraints, when relaxing these constraints the system might still fruitfully be applied to proof-of-concept scenarios.

5 Conclusion and Future Work

In this paper, after a compact introduction to the overall Heuristic-Driven Theory Projection framework for computational analogy-making, we provided a complexity analysis of the key mechanisms underlying HDTP, together with a short reflection on the obtained results. The obtained classical and parameterized complexity results do not affect the overall usefulness and functionality of HDTP as a computational tool for implementing an analogy-making mechanism, but currently do set limits to HDTP's status as a cognitive system: Although analogy-making as cognitive capacity can be modelled, tractability of the system can in general not be assumed.

With respect to future work, the main task is in searching for possibilities to reduce the complexity whilst still maintaining the functionality of HDTP. Besides further investigations into the complexity-relevant parameters within the setting, we see three possible lines of work. First, a study of the true power of argument insertion from a complexity theoretic perspective. Using argument insertion in its full generality, one can build any term incrementally by a sequence of argument insertions and fixations, which implies that it could be a source of much additional complexity. Next, a study searching possible restrictions on the representation language underlying HDTP, so as to limit complexity whilst still staying as expressive as possible (i.e. taking a step back from first-order logic into a less expressive but still suitable formalism). Finally, and perhaps most importantly, a direct address to the undecidability of the re-representational mechanism of the basic HDTP problem.

References

1. Schwering, A., Kühnberger, K.U., Kokinov, B.: Analogies: Integrating multiple cognitive abilities - guest editorial. *Journal of Cognitive Systems Research* 10 (2009)
2. Reitman, W.R., Grove, R.B., Shoup, R.G.: Argus: An information-processing model of thinking. *Behavioral Science* 9, 270–281 (1964)
3. Evans, T.G.: A heuristic program to solve geometric-analogy problems. In: *Proceedings of the Spring Joint Computer Conference, AFIPS 1964, April 21–23*, pp. 327–338. ACM, New York (Spring 1964)
4. Hofstadter, D., Mitchell, M.: The Copycat project: a model of mental fluidity and analogy-making. In: *Advances in Connectionist and Neural Computation Theory. Analogical Connections*, vol. 2, pp. 31–112. Ablex, New York (1994)
5. Falkenhainer, B., Forbus, K.D., Gentner, D.: The structure-mapping engine: Algorithm and examples. *Artificial Intelligence* 41, 1–63 (1989)
6. Gentner, D., Forbus, K.D.: MAC/FAC: A Model of Similarity-based Retrieval. *Cognitive Science* 19, 141–205 (1991)
7. Gentner, D.: Structure-mapping: A theoretical framework for analogy. *Cognitive Science* 7, 155–170 (1983)
8. Gust, H., Kühnberger, K.U., Schmid, U.: Metaphors and Heuristic-Driven Theory Projection (HDTP). *Theoretical Computer Science* 354, 98–117 (2006)
9. Schwering, A., Krumnack, U., Kühnberger, K.U., Gust, H.: Syntactic principles of Heuristic-Driven Theory Projection. *Journal of Cognitive Systems Research* 10(3), 251–269 (2009)
10. Guhe, M., Pease, A., Smaill, A., Schmidt, M., Gust, H., Kühnberger, K.U., Krumnack, U.: Mathematical reasoning with higher-order anti-unification. In: *Proceedings of the 32st Annual Conference of the Cognitive Science Society*, pp. 1992–1997 (2010)
11. Guhe, M., Pease, A., Smaill, A., Martinez, M., Schmidt, M., Gust, H., Kühnberger, K.U., Krumnack, U.: A computational account of conceptual blending in basic mathematics. A computational account of conceptual blending in basic mathematics. *Journal of Cognitive Systems Research* 12(3), 249–265 (2011)
12. Plotkin, G.D.: A note on inductive generalization. *Machine Intelligence* 5, 153–163 (1970)
13. Plotkin, G.D.: A further note in inductive generalization. *Machine Intelligence* 6, 101–124 (1971)
14. Krumnack, U., Schwering, A., Gust, H., Kühnberger, K.: Restricted Higher-Order Anti-Unification for Analogy Making. In: *Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI)*, vol. 4830, pp. 273–282. Springer, Heidelberg (2007)
15. Schmidt, M., Gust, H., Kühnberger, K.U., Krumnack, U.: Refinements of restricted higher-order anti-unification for heuristic-driven theory projection. In: *Proceedings 34th Annual German Conference on Artificial Intelligence*, Springer (2011)
16. Schwering, A., Gust, H., Kühnberger, K.U.: Solving geometric analogies with the analogy model hdt. In: *Taaten, N.A., van Rijn, H. (eds.) Proceedings of the 31st Annual Meeting of the Cognitive Science Society*, pp. 1780–1785. Cognitive Science Society, Austin (2009)
17. Besold, T.R., Gust, H., Krumnack, U., Abdel-Fattah, A., Schmidt, M., Kühnberger, K.U.: An argument for an analogical perspective on rationality & decision-making. In: *van Eijck, J., Verbrugge, R. (eds.) Proceedings of the Workshop on Reasoning About Other Minds: Logical and Cognitive Perspectives (RAOM 2011)*. CEUR Workshop Proceedings, Groningen, The Netherlands, vol. 751, pp. 20–31 (2011) CEUR-WS.org
18. Martinez, M., Besold, T.R., Abdel-Fattah, A., Kühnberger, K.U., Gust, H., Schmidt, M., Krumnack, U.: Towards a Domain-Independent Computational Framework for Theory Blending. In: *AAAI Technical Report of the AAAI Fall 2011 Symposium on Advances in Cognitive Systems*, pp. 210–217 (2011)

19. Martinez, M., Besold, T.R., Abdel-Fattah, A., Gust, H., Schmidt, M., Krumnack, U., Kühnberger, K.U.: Theory Blending as a Framework for Creativity in Systems for General Intelligence. In: Wang, P., Goertzel, B. (eds.) *Theoretical Foundations of AGI*. Atlantis Press (in press, 2012)
20. Jörg Flum, M.G.: *Parameterized Complexity Theory*. Springer (2006)
21. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer (1999)
22. Kuper, G.M., McAloon, K.W., Palem, K.V., Perry, K.J.: Efficient parallel algorithms for anti-unification and relative complement (1988)
23. Cesati, M.: *Compendium of parameterized problems*,
<http://www.sprg.uniroma2.it/home/cesati/research/compendium/>
24. van Rooij, I.: The tractable cognition thesis. *Cognitive Science* 32, 939–984 (2008)