Properties of Constrained Generalization Algorithms

Thierry Boy de La Tour

EPiC Series in Computing

Volume 50, 2017, Pages 64–77

GCAI 2017. 3rd Global Conference on Artificial Intelligence

Properties of Constrained Generalization Algorithms

Thierry Boy de la Tour

Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
thierry.boy-de-la-tour@imag.fr

Abstract

Two non deterministic algorithms for generalizing a solution of a constraint expressed in second order typed λ-calculus are presented. One algorithm derives from the proof of completeness of the higher order unification rules by D. C. Jensen and T. Pietrzykowski, the other is abstracted from an algorithm by N. Peltier and the author for generalizing proofs. A framework is developed in which such constrained generalization algorithms can be designed, allowing a uniform presentation for the two algorithms. Their relative strength at generalization is then analyzed through some properties of interest: their behaviour on valid and first order constraints, or whether they may be iterated or composed.

1 Introduction

In [2] a method for generalizing proofs in LK with equality is presented. It consists of three steps: by expressing inference rules, side conditions included, as second order equational constraints, the first step extracts from a given LK-proof a constraint expressing its syntactic validity. The second step lifts the proof by introducing as many variables as possible, yielding an *abstract proof* $\mathcal{P}$ that may not correspond to a valid LK-proof, an abstract constraint $\mathcal{X}$ that may not be valid, and a solution θ of $\mathcal{X}$, such that $\mathcal{P}\theta$ is the original proof. In the third step a *minimization* algorithm is applied to $\mathcal{X}, \theta$ in order to find a more general solution γ of $\mathcal{X}$, so that $\mathcal{P}\gamma$ is guaranteed to be a valid LK-proof, of which $\mathcal{P}\theta$ is therefore an instance.

As a very simple example consider the (trivial) proof of the sequent $P(a) \vdash P(a)$ by the LK axiom $\varphi \vdash \varphi$. We express this axiom as a constrained sequent $\varphi \vdash \psi \,|\, \varphi \doteq \psi$, where φ and ψ are first order variables. In this constrained system the proof translates as $P(a) \vdash P(a) \,|\, P(a) \doteq P(a)$, with a valid constraint. Note that $P(a) \vdash P(b) \,|\, P(a) \doteq P(b)$ is a valid proof in the constrained system, but since the constraint is unsatisfiable it does not correspond (translates) to a valid LK-proof. By introducing variables the proof is then lifted to the constrained proof $x \vdash y \,|\, x \doteq y$ (this is the proof $\mathcal{P}$ whith its constraint $\mathcal{X}$, which is not valid) together with the solution $\theta(x) = \theta(y) = P(a)$. The minimization algorithm, applied to $x \doteq y, \theta$ yields a solution $\gamma(x) = \gamma(y) = x$ more general than θ, and the proof $\mathcal{P}\gamma$ of $x \vdash x$ is an LK-proof more general than the initial one. Since the quantifier and equality rules of LK introduce second order variables, there is generally no most general proof such as this $\mathcal{P}\gamma$, see [2] for more elaborate examples.

This method is of course not restricted to LK-proofs in principle, it could be adapted to other proof systems as long as inference rules can be represented by second order constraints.

Yet the minimization algorithm presented in [2] depends in a non trivial way on peculiarities of LK with equality, especially the fact that the types that are admissible for variables (they stand not only for free and bound variables in first order formulæ but also for elements of *signatures*, i.e., function and predicate symbols) do not match the types of all logical symbols. Only the equality has the type of a predicate, and this strangely requires a special rule in the algorithm. Besides, this algorithm presents a complex behaviour and, being non deterministic, may output very different generalizations of θ, though in finite number. It is therefore liable to improvements, both in control (to find specific solutions) and extension (among a greater choice).

In order to design such algorithms it is therefore appropriate to adopt a simpler, more abstract setting, and the obvious choice is the simply typed λ-calculus (without constants) where the types of variables have no other restrictions than being of order at most 2. We thus assume a given pattern $\mathcal{P}$ and constraint $\mathcal{X}$ on the variables of $\mathcal{P}$, that defines the kind of objects we are interested in, namely the instances $\mathcal{P}\gamma$ of $\mathcal{P}$ where γ is a solution of $\mathcal{X}$. Assuming we know a special instance $\mathcal{P}\theta$ among these objects, we would like to discover possible generalizations of it, i.e., objects $\mathcal{P}\gamma$ of which $\mathcal{P}\theta$ is an instance. For this we need only search for generalizations γ of θ that satisfy $\mathcal{X}$, i.e., this does not depend on $\mathcal{P}$. A *constrained generalization algorithm (cga)* $\mathcal{R}$ defines, for all $\mathcal{X}$, a binary relation $\mathcal{R}_\mathcal{X}$ such that, for all solutions θ of $\mathcal{X}$ (input) there is a γ (output) with $\theta\ \mathcal{R}_\mathcal{X}\ \gamma$, and for all γ, $\theta\ \mathcal{R}_\mathcal{X}\ \gamma$ entails that γ is a solution of $\mathcal{X}$ more general than θ.

This approach to generalization is different from those commonly found in AI, where generalization is often considered as a way of inducing knowledge from "training" instances (see, e.g., [11]). But induction has long been criticized as a logically unsafe inference, notably by David Hume. One common way to circumvent this problem is to search for *least general generalizations (lgg)*, as in anti-unification which has been studied in many languages, see [7, 1]. In particular in [12] anti-unification in the Calculus of Constructions is a way of generalizing proofs. However, the lgg of two objects that have nothing in common necessarily exceeds the limits of sense. We believe that the only way generalization can be logically safe is to constrain it with what Mitchell calls *prior knowledge*[1] in [11]. The constraint $\mathcal{X}$ together with the pattern $\mathcal{P}$ can thus be seen as this prior knowledge that defines the reasonable limits that generalization is not allowed to cross.

As our constraints are equational this problem is closely related to second order unification, a well studied problem [4]. However, the fact that we are given a solution θ of $\mathcal{X}$ means that $\mathcal{X}$ is satisfiable. This is an important restriction since satisfiability of second order unification problems is undecidable [6]. Second order unification problems usually do not have a *most general unifier (mgu)*, which leaves room for many different generalization of a given unifier. This naturally leads to the unification algorithm in [9] which, unlike the one in [8], produces a *complete* set of unifiers of a higher order unification problem. Completeness here means that any unifier is an instance of one that can provably be produced by finite applications of their unification rules. This proof of completeness is constructive, and its algorithmic content is therefore a cga.

After more or less standard notations and definitions are fixed in Section 2, the rules for the two algorithms are presented in Section 3, together with a general framework in which their properties are obtained. The two algorithms are then compared, on the most simple forms of constraints in Section 4, and on general constraints in Section 5, where a difficult result of projectivity is obtained, though only for one algorithm.

[1] "Developing general methods for combining prior knowledge effectively with training data to constrain learning is a significant open problem".

2 Notations and Basic Definitions

Sequences of objects (variables, terms...) will be denoted as vectors: $\overrightarrow{x_n}$ is the sequence $x_1 \cdots x_n$. We may omit the length n, which may still be 0. We then write $\overline{x_n}$ for the set $\{x_1, \ldots, x_n\}$, and consistently $\overline{n}$ for $\{1, \ldots, n\}$. If $i \in \overline{n}$ then $\overrightarrow{x_{\setminus i}}$ denotes the sequence $x_1 \cdots x_{i-1} x_{i+1} \cdots x_n$. If necessary, elements of a sequence are (implicitly) separated by commas, e.g., $f(\overrightarrow{s_2})$ stands for $f(s_1, s_2)$.

We use the usual notion of simple types, and we inductively define $\overrightarrow{\tau_n} \to \rho$ as ρ if $n = 0$ and $\overrightarrow{\tau_{n-1}} \to (\tau_n \to \rho)$ if $n > 0$. Every type of order 1 or 2 can be written $\overrightarrow{\tau_n} \to \rho$ where ρ and the $\overline{\tau_n}$ are basic types; n is then the *arity* of this type. We assume a set $\mathcal{V}$ of *variables* such that each comes with a type of order 1 or 2, and there is an infinite supply of variables of each type. Well-typed terms are built as usual on variables with application $s(t)$ and abstraction $\lambda x.t.$, and $\mathrm{FV}(t)$ is the set of free variables occurring in t. The *arity* $\mathrm{a}(t)$ and order of a term t is that of its type; we consider terms of order at most 2. Assuming s has arity at least n and $\overrightarrow{t_n}$ have correct types, we write $s(t_1, \ldots, t_n)$ or simply $s(\overrightarrow{t_n})$ for s if $n = 0$ and for $s(\overrightarrow{t_{n-1}})(t_n)$ if $n > 0$. We write $\lambda\overrightarrow{x_n}.t.$ for t if $n = 0$ and for $\lambda\overrightarrow{x_{n-1}}.\lambda x_n.t.$. if $n > 0$. Every term can be written $\lambda\overrightarrow{x}.s(\overrightarrow{t}).$, where s is not an application; then $\overrightarrow{x}$ is the *prefix*, $s(\overrightarrow{t})$ the *matrix* and s the *head* of this term; if the head is some x_i then ($\overrightarrow{t}$ is empty) the term is a *projection*. We assume silent α-conversion, i.e., the prefix is not determined by the term but suitably chosen[2] depending on the context, and the matrix is determined by this choice.

Substitutions σ and their domains $\mathrm{Dom}(\sigma)$ and free variables $\mathrm{FV}(\sigma)$ are as usual. We write $[\,t_1/f_1, \ldots, t_n/f_n\,]$ for the substitution σ such that $\forall i \in \overline{n}, \sigma(f_i) = t_i$ and $\mathrm{Dom}(\sigma) \subseteq \{f_1, \ldots, f_n\}$ (hence $[\,]$ is the identity of $\mathcal{V}$). By *permutation* we will always mean a type-preserving permutation of $\mathcal{V}$, which is of course a substitution.

It is well known [5, 13] that β-reduction $\lambda x.s.(t) \to_\beta s[\,t/x\,]$ (resp. with η-reduction $\lambda x.t(x). \to_\eta t$ if $x \notin \mathrm{FV}(t)$, resp. with η-expansion, i.e., reversed η-reduction) are normalizing, yielding the β (resp. *short*, resp. *long*) normal form, or βnf (resp. *snf*, resp. *lnf*). We write $s \simeq t$ when s and t have the same snf (or lnf, equivalently), and similarly $\sigma \simeq \theta$ if $\sigma(f) \simeq \theta(f)$ for all $f \in \mathcal{V}$. The head of a βnf must be a variable. $t\sigma$ is defined as the βnf of the term obtained by replacing every $f \in \mathrm{FV}(t)$ by $\sigma(f)$, and similarly for $\theta\sigma$. Note that $\lambda x.t.\sigma = \lambda x.t\sigma.$ always holds since by silent α-conversion x is chosen such that $x \notin \mathrm{FV}(\sigma)$, making captures impossible. We assume that substitutions σ verify[3] $\mathrm{Dom}(\sigma) = \{f \in \mathcal{V} \mid f\sigma \neq f\}$. For any $V \subseteq \mathcal{V}$ we write $\sigma \simeq \theta[V]$ if $f\sigma \simeq f\theta$ for all $f \in V$. We write $s \lesssim t$ (resp. $\sigma \lesssim \theta$) if there is a substitution δ such that $s \simeq t\delta$ (resp. $\sigma \simeq \theta\delta$), and say that s (resp. σ) is *more general* than t (resp. θ) and that t is an *instance* of s. σ is a *variant* of θ if $\mathrm{Dom}(\sigma) = \mathrm{Dom}(\theta)$ and there is a permutation π such that $\sigma \simeq \theta\pi[\mathrm{Dom}(\sigma)]$.

An *equation* $\mathcal{E}$ is a directed pair $s \doteq t$ of terms of the same type, and $\mathcal{E} \equiv \top$ iff $s \simeq t$. Equations are implicitly converted to lnf. $\mathcal{E}\sigma$ is $s\sigma \doteq t\sigma$, a *constraint* $\mathcal{X}$ is a conjunction of equations (possibly the empty conjunction $\top$), and σ is a *solution* of $\mathcal{X}$ if $\mathcal{X}\sigma \equiv \top$.

Two terms r, r' have a *common image* if there exist terms $\overrightarrow{s}, \overrightarrow{t}$ such that $r(\overrightarrow{s}) \simeq r'(\overrightarrow{t})$. For any term s and $i \in \mathbb{N}$, let $\overrightarrow{u}$ be a prefix of its lnf and r the corresponding matrix, then $|s|_i$ is the number of occurrences of u_i in r (0 if $i > \mathrm{a}(s)$). Then s is *i-constant* if $i \in \bar{\mathrm{a}}(s)$ and $|s|_i = 0$, s is *free* if $|s|_j \geqslant 1$ for all $j \in \bar{\mathrm{a}}(s)$, and s is *affine* if $\forall j \in \bar{\mathrm{a}}(s)$, $|s|_j \leqslant 1$. Note that if s is i-constant than so is $s\sigma$.

[2] We may use de Bruijn indices [3] as a formal way of representing such terms, and then see bound variables as a convenient metanotation. We use distinct names for bound variables, e.g., u, v to mean that the chosen variables must be distinct, i.e., $u \neq v$. In contrast, free variables f, g may be equal.

[3] We may need to compute the snf of $f\sigma$ for all $f \in \mathrm{Dom}(\sigma)$ to ensure this, which we do implicitly. In fact, throughout the paper anytime we need a snf or a lnf we implicitly assume that it is computed as required.

3 Constrained Generalization Algorithms

The algorithms for generalizing θ w.r.t. a constraint $\mathcal{X}$ work by successive transformations of triples, starting with $\langle \mathcal{X}, [], \theta \rangle$ and refining the first two components until some $\langle \top, \sigma, \theta' \rangle$ is reached, where θ' is an extension of the initial θ to new variables, and σ is a new solution of $\mathcal{X}$ that is maintained more general than θ. A refinement step on a triple $\langle \mathcal{X}, \sigma, \theta \rangle$ selects an equation $\mathcal{E}$ in $\mathcal{X}$, and depending on $\mathcal{E}$ and θ applies a rule that yields a new constraint $\mathcal{Y}$ replacing $\mathcal{E}$ in $\mathcal{X}$ (by abuse of notation we denote $\mathcal{X}[\mathcal{Y}/\mathcal{E}]$ the result of this replacement), a substitution δ for refining both the constraint and σ, and a substitution ε that provides correct values for the new variables introduced by δ; the next triple is therefore $\langle \mathcal{X}[\mathcal{Y}/\mathcal{E}]\delta, \sigma\delta, \theta\varepsilon \rangle$. The relation between a triple and the next one is usually non deterministic, if only for the choice of $\mathcal{E}$ and of new variables. More formally:

Definition 1. For all $l \in \mathbb{N}$, an *l-run* of the algorithm $\mathcal{R}$ on $\mathcal{X}, \theta$ consists of a sequence of $l+1$ configurations $(\langle \mathcal{X}_k, \sigma_k, \theta_k \rangle)_{k=1}^{l+1}$ and a sequence $\overrightarrow{R_l}$ of rules of $\mathcal{R}$, verifying: $\langle \mathcal{X}_1, \sigma_1, \theta_1 \rangle = \langle \mathcal{X}, [], \theta \rangle$ and for all $k \in \overline{l}$, R_k is applied to some equation $\mathcal{E}_k$ in $\mathcal{X}_k$ and to θ_k, which yields new equations $\mathcal{Y}_k$ and substitutions δ_k and ε_k, and $\langle \mathcal{X}_{k+1}, \sigma_{k+1}, \theta_{k+1} \rangle = \langle \mathcal{X}_k[\mathcal{Y}_k/\mathcal{E}_k]\delta_k, \sigma_k\delta_k, \theta_k\varepsilon_k \rangle$. If $\mathcal{X}_{l+1} = \top$ the run is *complete* and its result is the restriction γ of σ_{l+1} to $\mathrm{FV}(\mathcal{X})$; then (and only if such a run exists) we say that the relation $\theta \, \mathcal{R}_{\mathcal{X}} \, \gamma$ holds.

Before we start defining the formal properties that these rules should obey for the relation $\mathcal{R}$ to be a cga, we first illustrate the previous notions by providing the rules of our two cgas and examples of runs. Each rule below is specified by a pattern for $\mathcal{E}$, a condition on the symbols extracted from $\mathcal{E}$ and on θ, and the resulting $\mathcal{Y}$, δ and ε. For sake of conciseness we only provide $\mathcal{Y}$ if it is different from $\mathcal{E}$, δ and ε if they are different from $[]$ (note that $\delta = []$ entails $\varepsilon = []$, hence if only one substitution is provided it must be δ). When the pattern together with the condition is not symmetric w.r.t. the left and right hand sides, i.e., if there exist s, t, θ such that the rule applies to $s \doteq t,\ \theta$ but not to $t \doteq s,\ \theta$, then we also assume (implicitly) a reversed version of the rule, and the rule is the union of the two asymmetric subcases, e.g., the imitation rule below applies if $\mathcal{E}$ matches $\lambda\overrightarrow{x}.f(\overrightarrow{s}) \doteq \lambda\overrightarrow{x}.g(\overrightarrow{t_m})$. or $\lambda\overrightarrow{x}.g(\overrightarrow{t_m}) \doteq \lambda\overrightarrow{x}.f(\overrightarrow{s})$..

All the variables in $\mathrm{FV}(\delta) \backslash \mathrm{FV}(\mathcal{E})$ are considered as *new* variables. We do not simply mean new w.r.t. the current triple $\langle \mathcal{X}, \sigma, \theta \rangle$ (i.e. disjoint from $\mathrm{FV}(\mathcal{X})$, $\mathrm{FV}(\sigma)$ and $\mathrm{FV}(\theta)$) but also w.r.t. the previous triples in the run so far *and* to previous runs. This profligacy is necessary for avoiding interactions when we consider successive runs (Section 5). The types of these new variables is obvious from those in $\mathcal{E}$ and θ.

The identification rule below requires notions that correspond to *agreement cap* and *opponent pairs* in [9], that we restrict to second order and to *simple comparison forms*, i.e., to affine terms. It is obvious that if two terms s, t have a common image and none is a projection then s and t have at least a common head. The following function separates the part that is common to s and t from their differences.

Definition 2. Let s, t be two affine terms in lnf with a common image, given fixed sequences of free variables $\overrightarrow{f}$, $\overrightarrow{g}$ of respective length $\mathrm{a}(t), \mathrm{a}(s)$, we define $\mathscr{D}(s,t)$ inductively as follows:

- $\mathscr{D}(\lambda\overrightarrow{u}.f(\overrightarrow{s_n}).,\lambda\overrightarrow{v}.f(\overrightarrow{t_n}).) = \langle f(\overrightarrow{r_n}), \prod_{i=1}^{n} \rho_i \rangle$ if $\forall i \in \overline{n}, \langle r_i, \rho_i \rangle = \mathscr{D}(\lambda\overrightarrow{u}.s_i., \lambda\overrightarrow{v}.t_i.)$,
- $\mathscr{D}(\lambda\overrightarrow{u}.u_i., t) = \langle u_i, [t/g_i] \rangle$ if $u_i \in \overline{u}$,
- $\mathscr{D}(s, \lambda\overrightarrow{v}.v_j.) = \langle v_j, [s/f_j] \rangle$ if $v_j \in \overline{v}$ and s is not a projection.

When computing $\langle r, \rho \rangle = \mathscr{D}(s,t)$ it is clear that each variable f_j or g_i enters ρ at most once per occurence of v_j or u_i, hence indeed at most once, which makes ρ a substitution.

Example 3. Let $s = \lambda u_1 u_2.f(g(u_2), u_1).$ and $t = \lambda v_1.f(v_1, g(x)).$ where $\mathrm{a}(x) = 0$, $\mathrm{a}(g) = 1$ and $\mathrm{a}(f) = 2$, hence s and t are in lnf and $\mathrm{a}(s) = 2$, $\mathrm{a}(t) = 1$, hence we are given new variables f_1, g_1 and g_2. They have a common image since $s(g(x), x) \simeq f(g(x), g(x)) \simeq t(g(x))$. Computing $\mathscr{D}(s,t)$ decomposes into computing $\mathscr{D}(\lambda u_1 u_2.g(u_2)., \lambda v_1.v_1.) = \langle v_1, [\,\lambda u_1 u_2.g(u_2)./f_1\,]\rangle$ and $\mathscr{D}(\lambda u_1 u_2.u_1., \lambda v_1.g(x).) = \langle u_1, [\,\lambda v_1.g(x)./g_1\,]\rangle$, hence the result is $\langle f(v_1, u_1), \rho\rangle$ where $\rho = [\,\lambda u_1 u_2.g(u_2)./f_1,\ \lambda v_1.g(x)./g_1\,]$. Now, if we let $\varepsilon = [\,\lambda u_1 u_2 v_1.f(v_1, u_1)./h\,]\rho$ then we see that $\lambda u_1 u_2.h(u_1, u_2, f_1(u_1, u_2)).\varepsilon \simeq s$ and $\lambda v_1.h(g_1(v_1), g_2(v_1), v_1).\varepsilon \simeq t$.

The first algorithm JP is defined by the six following rules. The five last rules correspond to the five unification rules in [9] and their conditions from the proof of completeness of these unification rules. Contrary to [9] we only apply these rules to the head symbols in the equations, hence we need a decomposition rule.

- Free decomposition rule: if $f\theta$ is free[4] then

$$\lambda\overrightarrow{x}.f(\overrightarrow{s_n}). \doteq \lambda\overrightarrow{x}.f(\overrightarrow{t_n}). \xrightarrow{\text{Fdcp}} \bigwedge_{i=1}^{n} \lambda\overrightarrow{x}.s_i. \doteq \lambda\overrightarrow{x}.t_i.$$

- Elimination rule: if $f\theta \simeq \lambda\overrightarrow{u}.r.$ is i-constant then

$$\lambda\overrightarrow{x}.f(\overrightarrow{s_n}). \doteq \lambda\overrightarrow{x}.f(\overrightarrow{t_n}). \xrightarrow{\text{Elim}} [\,\lambda\overrightarrow{u}.h(\overrightarrow{u_{\setminus i}})./f\,],\ [\,\lambda\overrightarrow{u_{\setminus i}}.r./h\,]$$

- Projection rule: if $f\theta$ is a projection then $\lambda\overrightarrow{x}.f(\overrightarrow{s}). \doteq t \xrightarrow{\text{Proj}} [\,f\theta/f\,]$
- Imitation rule: if $f \neq g \simeq g\theta$ and $f\theta \simeq \lambda\overrightarrow{u}.g(\overrightarrow{r_m}).$ then

$$\lambda\overrightarrow{x}.f(\overrightarrow{s}). \doteq \lambda\overrightarrow{x}.g(\overrightarrow{t_m}). \xrightarrow{\text{Imit}} [\,\lambda\overrightarrow{u}.g(f_1(\overrightarrow{u}), \ldots, f_m(\overrightarrow{u}))./f\,],\ \prod_{j=1}^{m} [\,\lambda\overrightarrow{u}.r_j./f_j\,]$$

- Repetition rule: if $f\theta \simeq \lambda\overrightarrow{u_n}.r.$, u_i occurs at least twice in r and r' is obtained from r by replacing one occurence of u_i by u_{n+1} then

$$\lambda\overrightarrow{x}.f(\overrightarrow{s}). \doteq t \xrightarrow{\text{Rept}} [\,\lambda\overrightarrow{u_n}.h(\overrightarrow{u_n}, u_i)./f\,],\ [\,\lambda\overrightarrow{u_{n+1}}.r'./h\,]$$

- Identification rule: if $f\theta \not\simeq f \neq g \not\simeq g\theta$, $f\theta$ and $g\theta$ are affine and non projective, and $\langle r, \rho\rangle = \mathscr{D}(f\theta, g\theta)$ then

$$\lambda\overrightarrow{x}.f(\overrightarrow{s_n}). \doteq \lambda\overrightarrow{x}.g(\overrightarrow{t_m}). \xrightarrow{\text{Idtf}} [\,\lambda\overrightarrow{u}.h(\overrightarrow{u}, f_1(\overrightarrow{u}), \ldots, f_m(\overrightarrow{u}))./f,\ \lambda\overrightarrow{v}.h(g_1(\overrightarrow{v}), \ldots, g_n(\overrightarrow{v}), \overrightarrow{v})./g\,],\ [\,\lambda\overrightarrow{u}\,\overrightarrow{v}.r./h\,]\rho$$

Example 4. We start with $\theta = [\,\lambda u.d(u,a)./f,\ \lambda u.d(u,a)./g,\ a/x,\ a/y\,]$ on the following constraint:

k	$\mathcal{X}_k$	R_k	δ_k	ε_k
1	$f(x) \doteq g(y)$	$\xrightarrow{\text{Idtf}}$	$[\,\lambda u.h_1(u, f_1(u))./f,\ \lambda v.h_1(g_1(v), v)./g\,]$	$[\,\lambda uv.d(u,a)./h_1,\ \lambda v.v./g_1\,]$
2	$h_1(x, f_1(x)) \doteq h_1(g_1(y), y)$	$\xrightarrow{\text{Elim}}$	$[\,\lambda u_1 u_2.h_2(u_1)./h_1\,]$	$[\,\lambda u.d(u,a)./h_2\,]$
3	$h_2(x) \doteq h_2(g_1(y))$	$\xrightarrow{\text{Fdcp}}$		
4	$x \doteq g_1(y)$	$\xrightarrow{\text{Proj}}$	$[\,\lambda v.v./g_1\,]$	
5	$x \doteq y$	$\xrightarrow{\text{Idtf}}$	$[\,z/x,\ z/y\,]$	$[\,a/z\,]$
6	$z \doteq z$	$\xrightarrow{\text{Fdcp}}$		
7	$\top$			

[4]Note that f may be bound, i.e., f be some x_i and then $f\theta = x_i$ is free, and $n = 0$.

Hence $\sigma_7 = \delta_1\delta_2\delta_4\delta_5$ and we obtain γ by computing $f\sigma_7$, $g\sigma_7$, $x\sigma_7$ and $y\sigma_7$, which yields $\gamma_1 = [\,\lambda u.h_2(u)./f,\ \lambda u.h_2(u)./g,\ z/x,\ z/y\,]$. It is easy to see that there is no other complete run than this one, hence that the result is unique up to variants, i.e., up to the chosen new variables. This is not always the case: both $\xrightarrow{\text{Fdcp}}$ and $\xrightarrow{\text{Proj}}$ apply to $f(x) \doteq f(x)$ when $f\theta = \lambda x.x.$, leading to non variant results (the rules above are not confluent w.r.t. variance).

Note that in a complete run of this algorithm the last rule R_l that produces $\top$ can only be $\xrightarrow{\text{Fdcp}}$, hence $\delta_l = \varepsilon_l = []$ and therefore $\sigma_{l+1} = \sigma_l$. We also see that, in all these rules if $f \in \mathrm{Dom}(\delta)$ then $f\theta \not\simeq f$, i.e., if we start with $f \notin \mathrm{Dom}(\theta)$ then we terminate with $f \notin \mathrm{Dom}(\sigma)$, which means that f behaves like a constant.

The second generalization algorithm BP is defined by the next four rules. It corresponds to the rules given in [2] though simplified mostly by the absence of constants and of restrictions on the types of new variables. The idea is to avoid using θ if possible, or to use either the equalities present in θ (though they may not be entailed by $\mathcal{X}$) or the values provided by θ (but only through shallow copies).

- Dependent decomposition rule[5]: if $f\theta \simeq g\theta$ and $\forall i \in \overline{n}, \lambda\overrightarrow{x}.s_i.\theta \simeq \lambda\overrightarrow{x}.t_i.\theta$ then

$$\lambda\overrightarrow{x}.f(\overrightarrow{s_n}). \doteq \lambda\overrightarrow{x}.g(\overrightarrow{t_n}). \xrightarrow{\text{Ddcp}} \textstyle\bigwedge_{i=1}^{n} \lambda\overrightarrow{x}.s_i. \doteq \lambda\overrightarrow{x}.t_i.,\ [\,f/g\,]$$

- Projection rule: as above.
- Replacement rule: if $f \notin \mathrm{FV}(\lambda\overrightarrow{x}.t.)$ then $\lambda\overrightarrow{x}.f(\overrightarrow{x}). \doteq \lambda\overrightarrow{x}.t. \xrightarrow{\text{Rplc}} \top,\ [\,\lambda\overrightarrow{x}.t./f\,]$
- Copy rule: if $f\theta \simeq \lambda\overrightarrow{u}.g(\overrightarrow{r_n}).$ is not a projection and $\overrightarrow{r_n} \neq \overrightarrow{u}$ then

$$\lambda\overrightarrow{x}.f(\overrightarrow{s}). \doteq t \xrightarrow{\text{Copy}} [\,\lambda\overrightarrow{u}.h(f_1(\overrightarrow{u}),\ldots,f_n(\overrightarrow{u}))./f\,],\ [\,g/h\,] \textstyle\prod_{i=1}^{n} [\,\lambda\overrightarrow{u}.r_i./f_i\,]$$

Example 5. We start with the same θ and constraint as in Example 4.

k	$\mathcal{X}_k$	R_k	δ_k	ε_k
1	$f(x) \doteq g(y)$	$\xrightarrow{\text{Copy}}$	$[\,\lambda u.h_1(f_1(u), f_2(u))./f\,]$	$[\,d/h_1,\ \lambda u.u./f_1,\ \lambda u.a./f_2\,]$
2	$h_1(f_1(x), f_2(x)) \doteq g(y)$	$\xrightarrow{\text{Copy}}$	$[\,\lambda u.h_2(g_1(u), g_2(u))./g\,]$	$[\,d/h_2,\ \lambda u.u./g_1,\ \lambda u.a./g_2\,]$
3	$h_1(f_1(x), f_2(x)) \doteq h_2(g_1(y), g_2(y))$	$\xrightarrow{\text{Ddcp}}$	$[\,h_1/h_2\,]$	
4	$f_1(x) \doteq g_1(y) \wedge f_2(x) \doteq g_2(y)$	$\xrightarrow{\text{Proj}}$	$[\,\lambda u.u./g_1\,]$	
5	$f_1(x) \doteq y \wedge f_2(x) \doteq g_2(y)$	$\xrightarrow{\text{Rplc}}$	$[\,f_1(x)/y\,]$	
6	$f_2(x) \doteq g_2(f_1(x))$	$\xrightarrow{\text{Copy}}$	$[\,\lambda u.z./f_2\,]$	$[\,a/z\,]$
7	$z \doteq g_2(f_1(x))$	$\xrightarrow{\text{Rplc}}$	$[\,g_2(f_1(x))/z\,]$	
8	$\top$			

Hence $\gamma_2 = [\,\lambda u.h_1(f_1(u), g_2(f_1(x)))./f,\ \lambda u.h_1(u, g_2(u))./g,\ f_1(x)/y\,]$. It is possible to apply $\xrightarrow{\text{Proj}}$ differently on $\mathcal{X}_4$ (with the projection $f_1\theta_4$), or to apply the rule $\xrightarrow{\text{Fdcp}}$ either on $f_1(x) \doteq g_1(y)$ (since $f_1\theta_4 = g_1\theta_4$ and $x\theta_4 = y\theta_4$) or to $f_2(x) \doteq g_2(y)$ (since $f_2\theta_4 = g_2\theta_4$) or at the start. The reader may check that these choices and others lead to many different generalizations.

We see that the two algorithms behave very differently, but before we start comparing their performances w.r.t. generalization, we need to make sure that they are indeed cgas. We first develop a general result in this direction.

[5] Again f or g may be bound, then $n = 0$ and $f = g$, hence $[\,f/g\,] = []$.

Definition 6. $\mathcal{Y}, \delta, \varepsilon$ are *well-formed w.r.t.* $\mathcal{E}$ if $\mathrm{FV}(\mathcal{Y}) \subseteq \mathrm{FV}(\mathcal{E})$, $\delta^2 = \delta$, $\mathrm{Dom}(\delta) \subseteq \mathrm{FV}(\mathcal{E})$ and $\mathrm{Dom}(\varepsilon) \subseteq \mathrm{FV}(\delta) \backslash \mathrm{FV}(\mathcal{E})$. A rule R is *correct* if, whenever $\mathcal{E}, \theta \xrightarrow{R} \mathcal{Y}, \delta, \varepsilon$ holds, then $\mathcal{Y}, \delta, \varepsilon$ are well-formed w.r.t. $\mathcal{E}$, $\mathcal{E}\theta \equiv \top$ entails both $\mathcal{Y}\theta \equiv \top$ and $\delta\theta\varepsilon \simeq \theta[\mathrm{Dom}(\delta)]$, and for all $\nu \gtrsim \delta$, $\mathcal{Y}\nu \equiv \top$ entails $\mathcal{E}\nu \equiv \top$. A set of rules is *complete* if every incomplete run can be extended.

Note that the conditions of correctness are only required if the rule applies, i.e., if the condition for applying the rule holds. However, the conditions of wellformedness are usually purely syntactic and, for all the rules above, do not depend on the condition for applying each rule. Remember that when a rule is applied the elements of $\mathrm{FV}(\delta) \backslash \mathrm{FV}(\mathcal{E})$ are always new.

Lemma 7. *If the rules of algorithm $\mathcal{R}$ are correct then for all $\mathcal{X}, \theta$ and $k \in \mathbb{N}$ such that $\mathcal{X}\theta \equiv \top$, every k-run of $\mathcal{R}$ on $\mathcal{X}, \theta$ verifies $\sigma_{k+1}\theta_{k+1} \simeq \theta[\mathrm{FV}(\mathcal{X})]$, $\mathcal{X}_{k+1}\theta_{k+1} \equiv \top$ and for all $\gamma \gtrsim \sigma_{k+1}$, $\mathcal{X}_{k+1}\gamma \equiv \top$ entails $\mathcal{X}\gamma \equiv \top$.*

Proof. By induction on k. For $k = 0$ this is trivial since $\sigma_1 = [\,]$, $\theta_1 = \theta$ and $\mathcal{X}_1 = \mathcal{X}$. Let $k \geqslant 1$ and assume as induction hypothesis (i.h.) that the property holds for $k-1$, hence $\mathcal{X}_k\theta_k \equiv \top$, and in particular $\mathcal{E}_k\theta_k \equiv \top$, which entails $\mathcal{Y}_k\theta_k \equiv \top$ and $\delta_k\theta_k\varepsilon_k \simeq \theta_k[\mathrm{Dom}(\delta_k)]$.

We first prove that $\varepsilon_k^2 = \varepsilon_k$. Let $f \in \mathrm{Dom}(\varepsilon_k)$, then f is a new variable and $f \in \mathrm{FV}(\delta_k)$ but $f \notin \mathrm{Dom}(\delta_k)$, hence there is a $g \in \mathrm{Dom}(\delta_k)$ such that $f \in \mathrm{FV}(g\delta_k)$. But $g\delta_k\theta_k\varepsilon_k \simeq g\theta_k$ and $f\theta_k\varepsilon_k = f\varepsilon_k$, hence $\mathrm{FV}(f\varepsilon_k) \subseteq \mathrm{FV}(g\delta_k\theta_k\varepsilon_k) = \mathrm{FV}(g\theta_k)$. Thus for all $f \in \mathrm{Dom}(\varepsilon_k)$, we have $\mathrm{FV}(f\varepsilon_k) \subseteq \mathrm{FV}(\theta_k)$, which is disjoint from $\mathrm{Dom}(\varepsilon_k)$, hence $\varepsilon_k^2 = \varepsilon_k$.

This brings $\delta_k\theta_k\varepsilon_k = \delta_k\theta_k\varepsilon_k^2 \simeq \theta_k\varepsilon_k[\mathrm{Dom}(\delta_k)]$, hence obviously $\delta_k\theta_k\varepsilon_k \simeq \theta_k\varepsilon_k$, and therefore $\sigma_{k+1}\theta_{k+1} = \sigma_k\delta_k\theta_k\varepsilon_k \simeq \sigma_k\theta_k\varepsilon_k$. But again $\mathrm{Dom}(\varepsilon_k)$ is disjoint from $\mathrm{FV}(\sigma_k)$, $\mathrm{FV}(\theta_k)$ and $\mathrm{FV}(\mathcal{X})$, hence $\sigma_k\theta_k\varepsilon_k = \sigma_k\theta_k[\mathrm{FV}(\mathcal{X})]$. By i.h. $\sigma_k\theta_k \simeq \theta[\mathrm{FV}(\mathcal{X})]$, hence $\sigma_{k+1}\theta_{k+1} \simeq \theta[\mathrm{FV}(\mathcal{X})]$.

Then we also have $\mathcal{X}_{k+1}\theta_{k+1} = \mathcal{X}_k[\,\mathcal{Y}_k/\mathcal{E}_k]\delta_k\theta_k\varepsilon_k \simeq \mathcal{X}_k[\,\mathcal{Y}_k/\mathcal{E}_k]\theta_k\varepsilon_k \equiv \top$.

To complete the last step, we need two preliminary results. We first prove that for all $j \in \overline{k}$, $\mathrm{FV}(\mathcal{X}_j) \cap \mathrm{Dom}(\sigma_j) = \varnothing$, by induction on j. It is obvious for $j = 1$ as $\sigma_1 = [\,]$. We now assume that $\mathrm{FV}(\mathcal{X}_j) \cap \mathrm{Dom}(\sigma_j) = \varnothing$. For all $f \in \mathcal{V}$, if $f \in \mathrm{Dom}(\delta_k)$, since $\delta_j^2 = \delta_j$ then $\mathrm{FV}(f\delta_j) \cap \mathrm{Dom}(\delta_j) = \varnothing$, which is also true if $f \notin \mathrm{Dom}(\delta_j)$. Let $\mathcal{X}' = \mathcal{X}_j[\,\mathcal{Y}_j/\mathcal{E}_j]$, as $\mathcal{X}_{j+1} = \mathcal{X}'\delta_j$ then $\mathrm{FV}(\mathcal{X}_{j+1}) \cap \mathrm{Dom}(\delta_j) = \varnothing$. Since $\mathrm{FV}(\mathcal{Y}_j) \subseteq \mathrm{FV}(\mathcal{E}_j)$ then $\mathrm{FV}(\mathcal{X}') \subseteq \mathrm{FV}(\mathcal{X}_j)$, hence $\mathrm{FV}(\mathcal{X}_{j+1}) \subseteq \mathrm{FV}(\mathcal{X}_j\delta_j) \subseteq \mathrm{FV}(\mathcal{X}_j) \cup \mathrm{FV}(\delta_j) = \mathrm{FV}(\mathcal{X}_j) \cup (\mathrm{FV}(\delta_j) \backslash \mathrm{FV}(\mathcal{E}_j))$ since $\mathrm{FV}(\mathcal{E}_j) \subseteq \mathrm{FV}(\mathcal{X}_j)$. This last set contains only new variables, hence none in common with $\mathrm{Dom}(\sigma_j)$, hence $\mathrm{FV}(\mathcal{X}_{j+1}) \cap \mathrm{Dom}(\sigma_j) = \varnothing$. But $\mathrm{Dom}(\sigma_{j+1}) \subseteq \mathrm{Dom}(\sigma_j) \cup \mathrm{Dom}(\delta_j)$, hence $\mathrm{FV}(\mathcal{X}_{j+1}) \cap \mathrm{Dom}(\sigma_{j+1}) = \varnothing$.

We next prove that $\delta_k\sigma_k\delta_k = \sigma_k\delta_k$. By the previous result we have $\mathrm{Dom}(\sigma_k) \cap \mathrm{FV}(\mathcal{X}_k) = \varnothing$. But $\mathrm{FV}(\delta_k) \cap \mathrm{FV}(\mathcal{E}_k) \subseteq \mathrm{FV}(\mathcal{X}_k)$ and $\mathrm{FV}(\delta_k) \backslash \mathrm{FV}(\mathcal{E}_k)$ is also disjoint from $\mathrm{Dom}(\sigma_k)$, hence so is $\mathrm{FV}(\delta_k)$. Now, for all $f \in \mathcal{V}$, if $f \in \mathrm{Dom}(\delta_k)$ then $f \notin \mathrm{Dom}(\sigma_k)$ hence $f\sigma_k\delta_k = f\delta_k$, and $\mathrm{FV}(f\delta_k) \subseteq \mathrm{FV}(\delta_k)$ hence $f\delta_k\sigma_k\delta_k = f\delta_k^2 = f\delta_k$. Of course if $f \notin \mathrm{Dom}(\delta_k)$ then $f\delta_k\sigma_k\delta_k = f\sigma_k\delta_k$, and we have proved that $\delta_k\sigma_k\delta_k = \sigma_k\delta_k$.

Finally, for all $\gamma \gtrsim \sigma_{k+1} = \sigma_k\delta_k$ such that $\top \equiv \mathcal{X}_{k+1}\gamma = \mathcal{X}_k[\,\mathcal{Y}_k/\mathcal{E}_k]\delta_k\gamma$, then in particular $\mathcal{Y}_k\delta_k\gamma \equiv \top$ hence $\mathcal{E}_k\delta_k\gamma \equiv \top$, and therefore $\mathcal{X}_k\delta_k\gamma \equiv \top$. But there exists a ρ such that $\gamma = \sigma_k\delta_k\rho$, hence $\mathcal{X}_k\delta_k\gamma = \mathcal{X}_k\delta_k\sigma_k\delta_k\rho = \mathcal{X}_k\sigma_k\delta_k\rho = \mathcal{X}_k\gamma$. Since obviously $\gamma \gtrsim \sigma_k$ then by i.h. we obtain $\mathcal{X}\gamma \equiv \top$, which completes the induction. □

The following general result will be useful to prove Lemma 17, a property specific to JP.

Corollary 8. For any l-run of $\mathcal{R}$, $k \in \overline{l}$ and $f \in \mathrm{FV}(\mathcal{X}_k)$, $f\delta_k \cdots \delta_l \lesssim f\theta_k$.

Proof. We have established above that $\delta_k\theta_k\varepsilon_k \simeq \theta_k\varepsilon_k$, thus $\delta_k\theta_k\varepsilon_k \cdots \varepsilon_l \simeq \theta_k\varepsilon_k \cdots \varepsilon_l$, i.e., $\delta_k\theta_{l+1} \simeq \theta_{l+1}$ for all $k \in \overline{l}$, hence $\delta_k \cdots \delta_l\theta_{l+1} \simeq \theta_{l+1} \simeq \theta_k[\mathrm{FV}(\mathcal{X}_k)]$. □

Theorem 9. *If the rules of $\mathcal{R}$ are correct, complete and terminating then $\mathcal{R}$ is a cga.*

Proof. Let θ such that $\mathcal{X}\theta \equiv \top$. By completeness and termination every run of $\mathcal{R}_\mathcal{X}$ on θ can be extended to a complete l-run for some $l \in \mathbb{N}$, hence there is a γ such that $\theta\ \mathcal{R}_\mathcal{X}\ \gamma$. Besides, for all such γ, by Definition 1 there is a complete l-run of $\mathcal{R}$ on $\mathcal{X}, \theta$ for some $l \in \mathbb{N}$, that ends in $\langle \mathcal{X}_{l+1}, \sigma_{l+1}, \theta_{l+1}\rangle$ where $\mathcal{X}_{l+1} = \top$ and $\gamma = \sigma_{l+1}[\mathrm{FV}(\mathcal{X})]$. It is obvious that $\sigma_{l+1} \lesssim \sigma_{l+1}$ and $\mathcal{X}_{l+1}\sigma_{l+1} \equiv \top$, which by Lemma 7 entails $\mathcal{X}\sigma_{l+1} \equiv \top$ and hence $\mathcal{X}\gamma \equiv \top$. We also have $\sigma_{l+1}\theta_{l+1} \simeq \theta[\mathrm{FV}(\mathcal{X})]$, hence $\gamma \lesssim \theta$. □

Note that this result is not restricted to second order languages. We can now apply it.

Theorem 10. JP *and* BP *are cgas.*

Proof. We leave it to the reader to check that all the rules presented above are correct, which is straightforward (for $\xrightarrow{\text{Idtf}}$, see Example 3).

We now prove that the rules of JP are complete. Assume $\mathcal{X} \neq \top$, hence there is an equation $\mathcal{E}$ in $\mathcal{X}$ of the form $\lambda\overrightarrow{x}.f(\overrightarrow{s_n}) \doteq \lambda\overrightarrow{x}.g(\overrightarrow{t_m})$.. If $f = g$ then $n = m$ and either $f\theta$ is free and $\xrightarrow{\text{Fdcp}}$ applies, or there is a i such that $f\theta$ is i-constant and $\xrightarrow{\text{Elim}}$ applies. Otherwise $f \neq g$, and if one of $f\theta, g\theta$ is a projection then $\xrightarrow{\text{Proj}}$ applies. If one of f, g, say g, is invariant by θ then since $\mathcal{E}\theta \equiv \top$ (by Lemma 7) the head of $f\theta$ must be g hence $\xrightarrow{\text{Imit}}$ applies. If one of $f\theta, g\theta$, say $f\theta$, is not affine then there is a i such that $|f\theta|_i \geqslant 2$ and $\xrightarrow{\text{Rept}}$ applies. Otherwise f, g are not invariant by θ, $f\theta, g\theta$ are affine and non projective hence rule $\xrightarrow{\text{Idtf}}$ applies.

Similarly we prove that the rules of BP are complete. Assume an equation $\mathcal{E}$ in $\mathcal{X}$ of the form $\lambda\overrightarrow{x}.f(\overrightarrow{s_n}) \doteq \lambda\overrightarrow{x}.g(\overrightarrow{t_m})$.. If one of $f\theta, g\theta$ is a projection then $\xrightarrow{\text{Proj}}$ applies. Otherwise if one on $f\theta, g\theta$'s snf is not a variable then $\xrightarrow{\text{Copy}}$ applies. Otherwise $f\theta$ and $g\theta$'s snf are variables, and since $\mathcal{E}\theta \equiv \top$ they must be the same variable, hence $f\theta \simeq g\theta$ and $n = m$. But variables are free terms, hence $\mathcal{E}\theta \equiv \top$ entails $\forall i \in \overline{n}, \lambda\overrightarrow{x}.s_i.\theta \simeq \lambda\overrightarrow{x}.t_i.\theta$, hence $\xrightarrow{\text{Ddcp}}$ applies[6].

To prove termination we define suitable well-orderings on the set of configurations $\langle \mathcal{X}, \sigma, \theta\rangle$. We write $\|t\|$ for the (standard) length of the matrix of t, $\|s \doteq t\|$ for $\|s\| + \|t\|$ and $\|\mathcal{X}\|$ for $\sum_{\mathcal{E}\in\mathcal{X}} \|\mathcal{E}\|$. For JP we need the lexicographic extension of $<$ on $\mathbb{N}$, with three components. The first is $\sum_{f\in\mathrm{FV}(\mathcal{X})} \|f\theta\|$, which strictly decreases by rules $\xrightarrow{\text{Proj}}$, $\xrightarrow{\text{Imit}}$, $\xrightarrow{\text{Idtf}}$ (because the common head of $f\theta, g\theta$ is counted only once in $h\varepsilon$) and does not increase by the other rules. The second is $\sum_{f\in\mathrm{FV}(\mathcal{X})} \sum_{j=1}^{\mathrm{a}(f)} (|f\theta|_j \dot{-} 1)$ (where $k \dot{-} 1 = \max(0, k-1)$), which strictly decreases by $\xrightarrow{\text{Rept}}$ (because a counted occurrence of u_i is replaced by an uncounted occurrence of u_{n+1}) and does not increase by $\xrightarrow{\text{Elim}}$ and $\xrightarrow{\text{Fdcp}}$. The last is $\|\mathcal{X}\|$ which strictly decreases by $\xrightarrow{\text{Elim}}$ and $\xrightarrow{\text{Fdcp}}$. Hence the set of rules of JP terminates.

For BP we need two components. The first is $|V| + 2\sum_{f\in\mathrm{FV}(\mathcal{X})\setminus V} \|f\theta\|$ where V is the set of $f \in \mathrm{FV}(\mathcal{X})$ such that the snf of $f\theta$ is a variable; it strictly decreases by rules $\xrightarrow{\text{Proj}}$, $\xrightarrow{\text{Rplc}}$, $\xrightarrow{\text{Copy}}$ (because the twice counted head g of $f\theta \notin \mathcal{V}$ is counted only once in $h\varepsilon = g \in \mathcal{V}$) and does not increase by $\xrightarrow{\text{Ddcp}}$ (because $\varepsilon = []$). The second is $\|\mathcal{X}\|$ which strictly decreases by $\xrightarrow{\text{Ddcp}}$, hence this set of rules terminates. □

[6] Note that $\xrightarrow{\text{Rplc}}$ has not been used. It is therefore easy to see that, by removing the conditions (on θ) and ε from $\xrightarrow{\text{Ddcp}}$, $\xrightarrow{\text{Proj}}$ and $\xrightarrow{\text{Copy}}$, we get three rules that are complete for second order unification. This somehow reverses the process by which we obtained the algorithm JP from the proof of completeness of the second order unification rules in [9].

4 Loose and First Order Generalizations

Definition 11. An algorithm $\mathcal{R}$ has the *loose generalization property* if for any constraint $\mathcal{X}$ such that $\mathcal{X} \equiv \top$ and any substitution θ, we have $\theta\ \mathcal{R}_{\mathcal{X}}\ []$.

Note that $[]$ is the most general solution of any valid constraint.

Theorem 12. *The algorithm* BP *has the loose generalization property.*

Proof. We start with $\sigma_1 = []$. If there is an equation $\mathcal{E}$ in $\mathcal{X} \equiv \top$, then $\mathcal{E} \equiv \top$ and as both sides are in lnf then $\mathcal{E}$ must be of the form $\lambda\overrightarrow{x}.f(\overrightarrow{s_n}). \doteq \lambda\overrightarrow{x}.f(\overrightarrow{s_n}).$, and for any θ the rule $\xrightarrow{\text{Ddcp}}$ applies and yields $\delta = []$ and a new constraint $\mathcal{X}[\bigwedge_{i=1}^{n} \lambda\overrightarrow{x}.s_i. \doteq \lambda\overrightarrow{x}.s_i./\mathcal{E}] \equiv \top$. Hence by a trivial induction the algorithm terminates with $\sigma_{l+1} = \delta_1 \cdots \delta_l = []$, and we get $\theta\ \mathsf{BP}_{\mathcal{X}}\ []$. □

In contrast, JP does not have the loose generalization property, as is easily proven by taking $\lambda\overrightarrow{x_2}.f(\overrightarrow{x_2}). \doteq \lambda\overrightarrow{x_2}.f(\overrightarrow{x_2}).$ for $\mathcal{X}$ and $\theta = [\lambda\overrightarrow{x_2}.x_1./f]$. As $f\theta$ is not free then we can only apply rule $\xrightarrow{\text{Elim}}$ or rule $\xrightarrow{\text{Proj}}$ to $\mathcal{X}$, and both yield a $\delta \neq []$.

It is not reasonable to expect that a generalization algorithm could reach a most general solution whenever there is one. However, we may think that it should be able to emulate the generalizing power of first order unification by finding a mgu in this case, which we first need to characterize.

Definition 13. A *first order constraint* is a constraint in which no abstraction occurs. A substitution θ is *first order* if $\mathrm{a}(\mathrm{Dom}(\theta)) \subseteq \{0\}$.

Theorem 14. *For any* $\mathcal{R} \in \{\mathsf{JP}, \mathsf{BP}\}$*, first order constraint* $\mathcal{X}$ *and first order solution* θ *of* $\mathcal{X}$*, there is a mgu* γ *of* $\mathcal{X}$ *such that* $\theta\ \mathcal{R}_{\mathcal{X}}\ \gamma$*.*

Proof. Obvious for $\mathsf{BP}_{\mathcal{X}}$; under our hypotheses the rule $\xrightarrow{\text{Ddcp}}$ subsumes the standard decomposition rule of first order unification $f(\overrightarrow{s_n}) \doteq f(\overrightarrow{s_n}) \to \bigwedge_{i=1}^{n} s_i \doteq t_i$ (it also applies to equations $x \doteq y$, yielding $\delta = [x/y]$, hence removes the equations $x \doteq x$) and the rule $\xrightarrow{\text{Rplc}}$ is the standard replacement rule $x \doteq t \to \top, [t/x]$ if $x \notin \mathrm{FV}(t)$, and these rules are known to produce a mgu.

The rule $\xrightarrow{\text{Fdcp}}$ corresponds to standard decomposition and to the elimination of equations $x \doteq x$. The rule $\xrightarrow{\text{Idtf}}$ reduces to $x \doteq y \to [z/x, y], [x\theta/z]$ if $x \neq y$, and the rule $\xrightarrow{\text{Imit}}$ to $x \doteq g(\overrightarrow{t_m}) \to [g(\overrightarrow{z_m})/x], [t_j\theta/z_j]_{j=1}^{m}$ (together with the symmetric version on $g(\overrightarrow{t_m}) \doteq x$). The standard replacement rule clearly derives from these rules (albeit with a harmless renaming of variables, see [10, Theorem 3.13]), hence $\mathsf{JP}_{\mathcal{X}}$ also produces a mgu. □

5 Weak Projectivity

Comparing the generalizing power of two cgas is a difficult task. Even if we were able to prove that a cga $\mathcal{R}$ always yields a more general solution than $\mathcal{R}'$, this would not rule out $\mathcal{R}'$ as useless since, the composition of two cgas (on the same constraint $\mathcal{X}$) being a cga, it may still be the case that ${\mathcal{R}'}^2_{\mathcal{X}}$ (or some other combination of the two cgas, e.g., $\mathcal{R}'_{\mathcal{X}} \circ \mathcal{R}_{\mathcal{X}}$) yields better solutions than $\mathcal{R}_{\mathcal{X}}$ or even $\mathcal{R}^2_{\mathcal{X}}$. On a first approach we may try to compare $\mathcal{R}^2_{\mathcal{X}}$ with $\mathcal{R}_{\mathcal{X}}$, independently of $\mathcal{R}'$. Of course this is made more complex by the non deterministic nature of cgas.

Definition 15. A relation $\mathcal{R}$ is *weakly projective* if for any θ, σ such that $\theta\ \mathcal{R}\ \sigma$ there is a variant σ' of σ such that $\sigma\ \mathcal{R}\ \sigma'$.

In general this will mean that if we apply a strategy to get a unique generalization, hence turning the non deterministic relation into a function, then this function is a projection (w.r.t. variants): one run of the algorithm exhausts its generalizing power. We now prove that this is indeed the case of $\mathsf{JP}_{\mathcal{X}}$ for any constraint $\mathcal{X}$. We therefore assume a complete l-run of the algorithm JP on $\mathcal{X}, \theta$ as in Definition 1. We first prove a lemma concerning the shape of the obtained generalization γ. We will use the equation:

$$|\lambda\overrightarrow{u}.f(\overrightarrow{s_m}).\sigma|_j = \sum_{i=1}^{m} |f\sigma|_i\, |\lambda\overrightarrow{u}.s_i.\sigma|_j \tag{1}$$

which is obvious since for each $i \in \overline{m}$ the bound variable u_j has $|\lambda\overrightarrow{u}.s_i.\sigma|_j$ occurrences in the lnf of $s_i\sigma$, which has $|f\sigma|_i$ occurrences in the lnf of $f(\overrightarrow{s_m})\sigma = f\sigma(s_1\sigma, \ldots, s_m\sigma)$.

Lemma 16. *For all $k \in \overline{l+1}$, $e \in \mathrm{FV}(\mathcal{X}_k)$ and $j \in \bar{\mathrm{a}}(e)$, $|e\delta_k \cdots \delta_l|_j \leqslant |e\theta_k|_j$*

Proof. We proceed by descending induction on k. The base case $k = l+1$ is trivial since $\mathrm{FV}(\mathcal{X}_{l+1}) = \varnothing$. We now assume that the property holds on $k+1$ for some $k \in \bar{l}$, we let $e \in \mathrm{FV}(\mathcal{X}_k)$, $\delta = \delta_{k+1} \cdots \delta_l$, $j \in \bar{\mathrm{a}}(e)$ and examine the different possibilities for R_k. First, we see that $e\theta_{k+1} = e\theta_k\varepsilon_k = e\theta_k$ since $\mathrm{Dom}(\varepsilon_k) \cap \mathrm{FV}(\mathcal{X}_k) = \varnothing$. For every rule but $\xrightarrow{\text{Fdcp}}$ we have $\mathcal{X}_{k+1} = \mathcal{X}_k\delta_k$, hence if $e \notin \mathrm{Dom}(\delta_k)$ then $e \in \mathrm{FV}(\mathcal{X}_{k+1})$ hence by i.h. $|e\delta_k\delta|_j = |e\delta|_j \leqslant |e\theta_{k+1}|_j = |e\theta_k|_j$. For these rules there only remains to examine the case $e \in \mathrm{Dom}(\delta_k)$.

- If R_k is $\xrightarrow{\text{Fdcp}}$ then $\mathcal{E}_k$ is some $\lambda\overrightarrow{x}.f(\overrightarrow{s}). \doteq \lambda\overrightarrow{x}.f(\overrightarrow{t})$. where $f\theta_k$ is free, hence $|f\theta_k|_j \geqslant 1$. If $e = f$ and $f \notin \mathrm{FV}(\mathcal{X}_{k+1})$ then $e \notin \mathrm{Dom}(\delta_k\delta)$ hence $|e\delta_k\delta|_j = |e|_j = 1 \leqslant |e\theta_k|_j$. Otherwise, either $e = f$ and $e \in \mathrm{FV}(\mathcal{X}_{k+1})$, or $e \neq f$ and since the terms $\overrightarrow{s}, \overrightarrow{t}$ occur in $\mathcal{Y}_k$, it is clear that $e \in \mathrm{FV}(\mathcal{X}_{k+1})$ again, and since $\mathrm{Dom}(\delta_k) = \varnothing$ then $e \notin \mathrm{Dom}(\delta_k)$ and we can proceed by i.h. as above.

- If R_k is $\xrightarrow{\text{Elim}}$, we assume $e \in \mathrm{Dom}(\delta_k)$ hence $\mathcal{E}_k$ is some $\lambda\overrightarrow{x}.e(\overrightarrow{s_n}). \doteq \lambda\overrightarrow{x}.e(\overrightarrow{t_n})$. where $e\theta_k \simeq \lambda\overrightarrow{u}.r.$ is i-constant, $\delta_k = [\,\lambda\overrightarrow{u}.h(\overrightarrow{u_{\setminus i}})./e\,]$, $\varepsilon_k = [\,\lambda\overrightarrow{u_{\setminus i}}.r./h\,]$ and $\mathcal{X}_{k+1} = \mathcal{X}_k\delta_k$. If $i = j$ then $e\delta_k = \lambda\overrightarrow{u}.h(\overrightarrow{u_{\setminus j}})$. is j-constant hence $|e\delta_k\delta|_j = 0$. If $j < i$ (resp. $i < j$) u_j is the j^{th} (resp. $(j-1)^{\text{th}}$) prefix variable of $\lambda\overrightarrow{u_{\setminus i}}.r.$ hence $|h\theta_{k+1}|_j = |\lambda\overrightarrow{u_{\setminus i}}.r.|_j = |\lambda\overrightarrow{u}.r.|_j = |e\theta_k|_j$ (resp. $|h\theta_{k+1}|_{j-1} = |e\theta_k|_j$) and since $f \in \mathrm{FV}(\mathcal{X}_k)$ then $h \in \mathrm{FV}(\mathcal{X}_{k+1})$, so by i.h. $|h\delta|_j \leqslant |h\theta_{k+1}|_j$ (resp. $|h\delta|_{j-1} \leqslant |h\theta_{k+1}|_{j-1}$). But u_j occurs only as the j^{th} (resp. $(j-1)^{\text{th}}$) argument of h in $h(\overrightarrow{u_{\setminus i}})$ hence $|e\delta_k\delta|_j = |\lambda\overrightarrow{u}.h(\overrightarrow{u_{\setminus i}}).\delta|_j = |h\delta|_j \leqslant |h\theta_{k+1}|_j = |e\theta_k|_j$ (resp. $|e\delta_k\delta|_j = |h\delta|_{j-1} \leqslant |h\theta_{k+1}|_{j-1} = |e\theta_k|_j$).

- If R_k is $\xrightarrow{\text{Imit}}$ then $\mathcal{E}_k$ is some $\lambda\overrightarrow{x}.e(\overrightarrow{s_n}). \doteq \lambda\overrightarrow{x}.g(\overrightarrow{t_m})$. where $e \neq g \simeq g\theta_k$ and $e\theta_k \simeq \lambda\overrightarrow{u}.g(\overrightarrow{r_m})$., $e\delta_k = \lambda\overrightarrow{u}.g(f_1(\overrightarrow{u}), \ldots, f_m(\overrightarrow{u}))$. and for all $i \in \overline{m}$, $f_i\theta_{k+1} = f_i\varepsilon_k = \lambda\overrightarrow{u}.r_i.$. Furthermore $e \in \mathrm{FV}(\mathcal{X}_k)$ entails $f_i \in \mathrm{FV}(\mathcal{X}_{k+1})$, hence by i.h. $|f_i\delta|_j \leqslant |f_i\theta_{k+1}|_j$. Now by (1) we get $|e\delta_k\delta|_j = \sum_{i=1}^{m} |g\delta|_i |\lambda\overrightarrow{u}.f_i(\overrightarrow{u}).\delta|_j$, but $|g\delta|_i = |g|_i = 1$ and again by (1)

  $$|\lambda\overrightarrow{u}.f_i(\overrightarrow{u}).\delta|_j = \sum_{q=1}^{n} |f_i\delta|_q |\lambda\overrightarrow{u}.u_q.\delta|_j = \sum_{q=1}^{n} |f_i\delta|_q |\lambda\overrightarrow{u}.u_q.|_j = |f_i\delta|_j,$$

  hence $|e\delta_k\delta|_j = \sum_{i=1}^{m} |f_i\delta|_j \leqslant \sum_{i=1}^{m} |f_i\theta_{k+1}|_j$. And yet again by (1)

  $$|e\theta_k|_j = |\lambda\overrightarrow{u}.g(\overrightarrow{r_m}).|_j = \sum_{i=1}^{m} |g|_i |\lambda\overrightarrow{u}.r_i.|_j = \sum_{i=1}^{m} |f_i\theta_{k+1}|_j \geqslant |e\delta_k\delta|_j.$$

The cases $\xrightarrow{\text{Proj}}$, $\xrightarrow{\text{Rept}}$ and $\xrightarrow{\text{Idtf}}$ are similar. □

Hence for all $f \in \text{FV}(\mathcal{X})$ we have $|f\gamma|_j \leqslant |f\theta|_j$, which reveals a limit to the generalizing power of JP. Example 5 shows that this property is not true of BP, as $|g\gamma_2|_1 = 2 > |g\theta|_1$.

Next we prove that the first run can be mimicked by a second, parallel one. The difficulty is to prove that the condition for applying a rule on an equation is still valid, although we have changed θ to γ. Another difficulty is that this second run also introduces new variables, and we have to shift (by means of a permutation) from the new variables of the first run to those of the second run. In particular, new variables that behave like constants, i.e., that are fixpoints of θ_l, should correspond in both runs since this has an influence on the conditions of rules $\xrightarrow{\text{Imit}}$ and $\xrightarrow{\text{Idtf}}$.

Lemma 17. *Let γ be the result of the first run of $\mathsf{JP}_\mathcal{X}$ on θ, then for all $k \in \overline{l+1}$ there is a $(k-1)$-run of $\mathsf{JP}_\mathcal{X}$ on γ that ends on $\langle \mathcal{X}_k\pi_k, \sigma'_k, \theta'_k\rangle$ where π_k is a permutation between the new variables of both runs so far, such that $\pi_k\sigma'_k = \sigma_k\pi_k$, $\pi_k\theta'_k \simeq \delta_k \cdots \delta_l[\text{FV}(\mathcal{X}_k) \cap \text{Dom}(\theta_k)]$ and $\text{FV}(\mathcal{X}_k) \cap \text{Dom}(\theta'_k)\pi_k^{-1} \subseteq \text{Dom}(\theta_k)$.*

Proof. By induction on k. For $k = 1$ we take $\pi_1 = []$, hence $\langle \mathcal{X}_1\pi_1, \sigma'_1, \theta'_1\rangle = \langle \mathcal{X}, [], \gamma\rangle$ is the end of a 0-run of $\mathsf{JP}_\mathcal{X}$ on γ, and the required properties are met by definition of γ and θ_1.

We now assume the property for $k \in \bar{l}$. We refer to the objects defined by this $(k-1)$-run by priming the metavariables of the first run, in particular we have $\sigma'_k = \delta'_1 \cdots \delta'_{k-1}$ and $\theta'_k = \gamma\varepsilon'_1 \cdots \varepsilon'_{k-1}$. $\mathcal{E}_k\pi_k$ is an equation in $\mathcal{X}_k\pi_k$; we first prove that the rule R_k applies to $\mathcal{E}_k\pi_k, \theta'_k$ which yields new equations $\mathcal{Y}_k\pi_k$ and substitutions $\delta'_k, \varepsilon'_k$, from which we define a suitable permutation π that commutes with π_k and such that together with $\pi_{k+1} = \pi_k\pi$ verify the following three properties: $\pi_{k+1}\delta'_k = \delta_k\pi_{k+1}$, $\pi\varepsilon'_k \simeq \delta_{k+1} \cdots \delta_l[\text{Dom}(\varepsilon_k)]$ and $\text{Dom}(\varepsilon_k)\pi = \text{Dom}(\varepsilon'_k)$. We let $\delta = \delta_{k+1} \cdots \delta_l$ and we examine the different possibilities for R_k.

- If R_k is $\xrightarrow{\text{Fdcp}}$ then $\mathcal{E}_k$ is some $\lambda\overrightarrow{x}.f(\overrightarrow{s}). \doteq \lambda\overrightarrow{x}.f(\overrightarrow{t}).$ where $f\theta_k$ is free, hence for all $i \in \bar{\text{a}}(f)$, $f\theta_k$ is not i-constant and since it is an instance of $f\delta_k\delta$ then $f\delta_k\delta$ is not i-constant. If $f \notin \text{Dom}(\theta_k)$ then by i.h. $f \notin \text{Dom}(\theta'_k)\pi_k^{-1}$, hence $f\pi_k\theta'_k = f\pi_k$ is free; otherwise by i.h. $f\pi_k\theta'_k \simeq f\delta_k\delta$ is also free. $\xrightarrow{\text{Fdcp}}$ can therefore be applied to $\mathcal{E}_k\pi_k, \theta'_k$, yielding $\delta'_k = [] = \delta_k$ and $\varepsilon'_k = [] = \varepsilon_k$; we then let $\pi = []$. Since $\text{Dom}(\varepsilon_k) = \text{Dom}(\varepsilon'_k) = \varnothing$, the three properties are trivial.

- If R_k is $\xrightarrow{\text{Elim}}$, then $\mathcal{E}_k$ is some $\lambda\overrightarrow{x}.f(\overrightarrow{s_n}). \doteq \lambda\overrightarrow{x}.f(\overrightarrow{t_n}).$ where $f\theta_k \simeq \lambda\overrightarrow{u}.r.$ is i-constant, $\delta_k = [\lambda\overrightarrow{u}.h(\overrightarrow{u_{\setminus i}})./f]$ and $\varepsilon_k = [\lambda\overrightarrow{u_{\setminus i}}.r./h]$. By i.h. $f\pi_k\theta'_k \simeq f\delta_k\delta = \lambda\overrightarrow{u}.h\delta(\overrightarrow{u_{\setminus i}}).$ which is obviously i-constant hence $\xrightarrow{\text{Elim}}$ can be applied on $\mathcal{E}_k\pi_k, \theta'_k$ yielding $\delta'_k = [\lambda\overrightarrow{u}.h'(\overrightarrow{u_{\setminus i}})./f\pi_k]$, and $\varepsilon'_k = [\lambda\overrightarrow{u_{\setminus i}}.h\delta(\overrightarrow{u_{\setminus i}})./h'] \simeq [h\delta/h']$, where h' is new also w.r.t. the first run, hence $\pi = (h\ h')$ commutes with π_k. We then easily check that $\delta_k\pi_{k+1} = \pi[\lambda\overrightarrow{u}.h'(\overrightarrow{u_{\setminus i}})./f]\pi_k = \pi\pi_k[\lambda\overrightarrow{u}.h'(\overrightarrow{u_{\setminus i}})./f\pi_k] = \pi_{k+1}\delta'_k$, that $\pi\varepsilon'_k \simeq [h\delta/h, h/h'] = \delta[\text{Dom}(\varepsilon_k)]$ and that $\text{Dom}(\varepsilon_k)\pi = \{h\}\pi = \{h'\} = \text{Dom}(\varepsilon'_k)$.

- If R_k is $\xrightarrow{\text{Proj}}$ then $\mathcal{E}_k$ is some $\lambda\overrightarrow{x}.f(\overrightarrow{s}). \doteq t$ where $f\theta_k$ is a projection $\lambda\overrightarrow{u}.u_i$. hence $\delta_k = [\lambda\overrightarrow{u}.u_i./f]$ and $\varepsilon_k = []$. By i.h. $f\pi_k\theta'_k \simeq f\delta_k\delta = \lambda\overrightarrow{u}.u_i.$ is a projection hence $\xrightarrow{\text{Proj}}$ can be applied to $\mathcal{E}_k\pi_k, \theta'_k$ and yields $\delta'_k = [\lambda\overrightarrow{u}.u_i./f\pi_k]$ and $\varepsilon'_k = []$. We see that $\delta_k\pi_k = \pi_k\delta'_k$ hence that the three properties hold with $\pi = []$.

- If R_k is $\xrightarrow{\text{Imit}}$ then $\mathcal{E}_k$ is some $\lambda\overrightarrow{x}.f(\overrightarrow{s_n}). \doteq \lambda\overrightarrow{x}.g(\overrightarrow{t_m}).$ where $f \neq g \simeq g\theta_k$ and $\delta_k = [\lambda\overrightarrow{u}.g(f_1(\overrightarrow{u}), \ldots, f_m(\overrightarrow{u}))./f]$, $\text{Dom}(\varepsilon_k) = \{f_1, \ldots, f_m\}$. Then $f\pi_k \neq g\pi_k$ and $g\pi_k\theta'_k =$

$g\pi_k$ since $g \notin \mathrm{Dom}(\theta_k)$ hence $g\pi_k \notin \mathrm{Dom}(\theta'_k)$. Thus $\xrightarrow{\text{Imit}}$ applies to $\mathcal{E}_k\pi_k, \theta'_k$ and yields $\delta'_k = [\,\lambda\overrightarrow{u}.g(f'_1(\overrightarrow{u}),\ldots,f'_m(\overrightarrow{u}))./f\pi_k\,]$ and $\varepsilon'_k = [\,\lambda\overrightarrow{u}.f_i\delta(\overrightarrow{u})./f'_i\,]_{i=1}^m \simeq [\,f_i\delta/f'_i\,]_{i=1}^m$ since by i.h. $f\pi_k\theta' \simeq f\delta_k\delta = \lambda\overrightarrow{u}.g(f_1\delta(\overrightarrow{u}),\ldots,f_m\delta(\overrightarrow{u}))$.. The f'_i's are new so that $\pi = (f_i\ f'_i)_{i=1}^m$ commutes with π_k, hence $\delta_k\pi_{k+1} = \pi[\,\lambda\overrightarrow{u}.g(f'_1(\overrightarrow{u}),\ldots,f'_m(\overrightarrow{u}))./f\,]\pi_k = \pi_{k+1}\delta'_k$, $\pi\varepsilon'_k \simeq [\,f_i\delta/f_i, f_i/f'_i\,]_{i=1}^m = \delta[\mathrm{Dom}(\varepsilon_k)]$ and $\mathrm{Dom}(\varepsilon_k)\pi = \{f'_1,\ldots,f'_m\} = \mathrm{Dom}(\varepsilon_k)$.

- If R_k is $\xrightarrow{\text{Rept}}$ then $\mathcal{E}_k$ is some $\lambda\overrightarrow{x}.f(\overrightarrow{s_n}). \doteq t$ where $f\theta_k \simeq \lambda\overrightarrow{u_n}.r.$ with $|f\theta_k|_i \geqslant 2$, and then $\delta_k = [\,\lambda\overrightarrow{u_n}.h(\overrightarrow{u_n},u_i)./f\,]$, $\mathrm{Dom}(\varepsilon_k) = \{h\}$ and $h\theta_{k+1} = \lambda\overrightarrow{u_{n+1}}.r'.$ as in the definition of the rule, so that $|h\theta_{k+1}|_i = |f\theta_k|_i - 1 \geqslant 1$ and $|h\theta_{k+1}|_{n+1} = 1$. Since $h\theta_{k+1}$ is an instance of $h\delta$ (by Corollary 8 and Theorem 9) then $|h\delta|_i \geqslant 1$ and $|h\delta|_{n+1} \geqslant 1$, but by i.h. and (1) $|f\pi_k\theta'_k|_i = |f\delta_k\delta|_i = \sum_{j=1}^n |h\delta|_j|\lambda\overrightarrow{u}.u_j.|_i + |h\delta|_{n+1}|\lambda\overrightarrow{u}.u_i.|_i = |h\delta|_i + |h\delta|_{n+1} \geqslant 2$ hence $\xrightarrow{\text{Rept}}$ can be applied to $\mathcal{E}_k\pi_k, \theta'_k$. We write $h\delta = \lambda\overrightarrow{u_{n+1}}.d'.$ and $d = d'[\,u_i/u_{n+1}\,]$, then $f\pi_k\theta'_k \simeq f\delta_k\delta = \lambda\overrightarrow{u_n}.h\delta(\overrightarrow{u_n},u_i). = \lambda\overrightarrow{u_n}.d.$ and d' can be obtained from d by replacing one occurence of u_i by u_{n+1}, hence an application of $\xrightarrow{\text{Rept}}$ on $\mathcal{E}_k\pi_k, \theta'_k$ does yield $\delta'_k = [\,\lambda\overrightarrow{u_n}.h'(\overrightarrow{u_n},u_i)./f\pi_k\,]$ and $\varepsilon'_k = [\,\lambda\overrightarrow{u_{n+1}}.d'./h'\,] \simeq [\,h\delta/h'\,]$. Let $\pi = (h\ h')$, as h and h' are new variables then π and π_k commute, $\delta_k\pi_{k+1} = [\,\lambda\overrightarrow{u_n}.h(\overrightarrow{u_n},u_i)./f\,]\pi\pi_k = \pi[\,\lambda\overrightarrow{u_n}.h'(\overrightarrow{u_n},u_i)./f\,]\pi_k = \pi\pi_k[\,\lambda\overrightarrow{u_n}.h'(\overrightarrow{u_n},u_i)./f\pi_k\,] = \pi_{k+1}\delta'_k$, $\pi\varepsilon'_k \simeq [\,h\delta/h, h/h'\,] = \delta[\mathrm{Dom}(\varepsilon_k)]$ and $\mathrm{Dom}(\varepsilon_k)\pi = \{h'\} = \mathrm{Dom}(\varepsilon'_k)$.

- If R_k is $\xrightarrow{\text{Idtf}}$ then $\mathcal{E}_k$ is some $\lambda\overrightarrow{x}.f(\overrightarrow{s_n}). \doteq \lambda\overrightarrow{x}.g(\overrightarrow{t_m}).$ where $f\theta_k \not\simeq f \neq g \not\simeq g\theta_k$, $f\theta_k$ and $g\theta_k$ are affine and non projective; let $s = \lambda\overrightarrow{u}.h(\overrightarrow{u}, f_1(\overrightarrow{u}),\ldots,f_m(\overrightarrow{u}))$. and $t = \lambda\overrightarrow{v}.h(g_1(\overrightarrow{v}),\ldots,g_n(\overrightarrow{v}),\overrightarrow{v})$. so that $\delta_k = [\,s/f, t/g\,]$, and we also have $h \in \mathrm{Dom}(\varepsilon_k) \subseteq \{h, f_1,\ldots,f_m,g_1,\ldots,g_n\}$. By i.h. $f\pi_k\theta'_k \simeq f\delta_k\delta$ which cannot be projective since its instance $f\theta_k$ is not projective. By definition $h\varepsilon_k = h\theta_{k+1}$ and the $f_j\theta_{k+1}$'s are non projective hence similarly $h\delta$ and the $f_j\delta$'s are non projective. The head e of $h\delta$ must therefore be introduced by some $\delta_{k'}$ for $k' > k$. If $R_{k'}$ is the imitation rule then $e\theta_{k'} = e$, otherwise e is a new symbol, hence in all cases we have $e \neq f$. As h is the head of $f\delta_k$ then e is the head of $f\delta_k\delta$ hence $f\delta_k\delta \not\simeq f$. Since $f\theta_k$ is affine then for all $i \in \bar{\mathrm{a}}(f)$ we have $|f\theta_k|_i \leqslant 1$, hence by Lemma 16 $|f\delta_k\delta|_i \leqslant 1$, i.e., $f\delta_k\delta$ is affine. Similarly we prove that $g\pi_k\theta'_k$ is non projective, different from g and affine, hence the rule $\xrightarrow{\text{Idtf}}$ can be applied on $\mathcal{E}_k\pi_k, \theta'_k$, yielding $\delta'_k = [\,s'/f\pi_k, t'/g\pi_k\,]$ where $s' = \lambda\overrightarrow{u}.h'(\overrightarrow{u}, f'_1(\overrightarrow{u}),\ldots,f'_m(\overrightarrow{u}))$. and $t' = \lambda\overrightarrow{v}.h'(g'_1(\overrightarrow{v}),\ldots,g'_n(\overrightarrow{v}),\overrightarrow{v})$.. Let $\pi = (h\ h')\prod_{j=1}^m(f_j\ f'_j)\prod_{i=1}^n(g_i\ g'_i)$ and $\varepsilon'_k = [\,\lambda\overrightarrow{u}\,\overrightarrow{v}.r'./h'\,]\rho$ where $\langle r', \rho\rangle = \mathscr{D}(f\pi_k\theta'_k, g\pi_k\theta'_k) = \mathscr{D}(s\delta, t\delta)$. As above π and π_k commute and $\delta_k\pi_{k+1} = [\,s/f, t/g\,]\pi\pi_k = \pi[\,s'/f, t'/g\,]\pi_k = \pi\pi_k[\,s'/f\pi_k, t'/g\pi_k\,] = \pi_{k+1}\delta'_k$.

  We now write $h\delta = \lambda\overrightarrow{u}\,\overrightarrow{v}.d.$ and prove that $r' = d$. It is clear that the matrices of $s\delta$ and $t\delta$ (the left and right arguments of $\mathscr{D}$) are both instances of d, and that they differ exactly at the positions of the bound variables in d since these are instantiated by different terms on each side, i.e., $u_i \neq g_i\delta(\overrightarrow{v})$ and $f_j\delta(\overrightarrow{u}) \neq v_j$ (we always assume $\overline{u} \cap \overline{v} = \varnothing$). If during the computation a variable u_i is encountered on the left, as the $f_j\delta$'s are non projective then it must be at the position of u_i in d, and then u_i enters r' and $[\,\lambda\overrightarrow{v}.g_i\delta(\overrightarrow{v})./g'_i\,] \simeq [\,g_i\delta/g'_i\,]$ enters ρ. Otherwise v_j is encountered on the right while the left is not u_i; hence it must be at the position of v_j in d and then v_j enters r' (since $f_j\delta$ is non projective) and $[\,f_j\delta/f'_j\,]$ enters ρ. Hence $r' = d$, but this also shows that u_i (resp. v_j) occurs in d iff $g'_i \in \mathrm{Dom}(\rho)$ (resp. $f'_j \in \mathrm{Dom}(\rho)$).

  For all $g_i \in \mathrm{Dom}(\varepsilon_k)$, we also know that $h\theta_{k+1}$ cannot be i-constant, but this is an instance of $h\delta$ hence u_i occurs in d and therefore $g'_i = g_i\pi \in \mathrm{Dom}(\varepsilon'_k)$. Conversely, if

$g_i \notin \mathrm{Dom}(\varepsilon_k)$ then $|h\theta_{k+1}|_i = 0$ and by Lemma 16 u_i does not occur in d and therefore $g'_i = g_i\pi \notin \mathrm{Dom}(\varepsilon'_k)$. Similarly we prove that $f_j \in \mathrm{Dom}(\varepsilon_k)$ iff $f_j\pi \in \mathrm{Dom}(\varepsilon'_k)$, and we have both $h \in \mathrm{Dom}(\varepsilon_k)$ and $h' \in \mathrm{Dom}(\varepsilon'_k)$, hence $\mathrm{Dom}(\varepsilon_k)\pi = \mathrm{Dom}(\varepsilon'_k)$.

We have $h\pi\varepsilon'_k = h'\varepsilon'_k = \lambda\vec{u}\,\vec{v}.r'. = h\delta$. For all $g_i \in \mathrm{Dom}(\varepsilon_k)$, $g_i\pi = g'_i \in \mathrm{Dom}(\varepsilon'_k)$ hence $g_i\pi\varepsilon'_k = g'_i\rho \simeq g_i\delta$ (as shown above), and similarly for the f_j's in $\mathrm{Dom}(\varepsilon_k)$, which proves that $\pi\varepsilon'_k \simeq \delta[\mathrm{Dom}(\varepsilon_k)]$.

In all cases we easily see that $\mathrm{Dom}(\pi) \cap \mathrm{FV}(\mathcal{X}_k) = \varnothing$, and hence that $\mathcal{X}_k\pi_k\,[\,\mathcal{Y}_k\pi_k\,/\mathcal{E}_k\pi_k\,]\,\delta'_k = \mathcal{X}_k\,[\,\mathcal{Y}_k\,/\mathcal{E}_k\,]\,\pi_k\delta'_k = \mathcal{X}_k\,[\,\mathcal{Y}_k\,/\mathcal{E}_k\,]\,\pi\pi_k\delta'_k = \mathcal{X}_k\,[\,\mathcal{Y}_k\,/\mathcal{E}_k\,]\,\delta_k\pi_{k+1} = \mathcal{X}_{k+1}\pi_{k+1}$. We let $\sigma'_{k+1} = \sigma'_k\delta'_k$ and $\theta'_{k+1} = \theta'_k\varepsilon'_k$, hence we have proved that there is a k-run of $\mathsf{JP}_\mathcal{X}$ from γ that ends on $\langle \mathcal{X}_{k+1}\pi_{k+1}, \sigma'_{k+1}, \theta'_{k+1}\rangle$. We also see that $\mathrm{FV}(\sigma'_k) = \mathrm{FV}(\delta'_1 \cdots \delta'_{k-1})$ only contain variables from $\mathrm{FV}(\mathcal{X})$ and the new variables of the second run before step k, which are kept separate from those of the first run, in particular from $\mathrm{FV}(\mathcal{X}_{k+1})$, hence it is disjoint from $\mathrm{Dom}(\pi)$ and therefore σ'_k and π commute. Then by i.h. $\pi_{k+1}\sigma'_{k+1} = \pi_k\pi\sigma'_k\delta'_k = \pi_k\sigma'_k\pi\delta'_k = \sigma_k\pi_k\pi\delta'_k = \sigma_k\pi_{k+1}\delta'_k = \sigma_k\delta_k\pi_{k+1} = \sigma_{k+1}\pi_{k+1}$.

For all f in $\mathrm{FV}(\mathcal{X}_{k+1}) \cap \mathrm{Dom}(\theta_{k+1})$ we have $f \notin \mathrm{Dom}(\delta_k)$ and either $f \in \mathrm{Dom}(\theta_k)$ or $f \in \mathrm{Dom}(\varepsilon_k)$. If $f \in \mathrm{FV}(\mathcal{X}_k)$ then $f \in \mathrm{Dom}(\theta_k)$ and therefore $f\pi_{k+1}\theta'_{k+1} = f\pi_k\theta'_k \simeq f\delta_k\delta$ by i.h., but $f\delta_k \simeq f$ hence $f\pi_{k+1}\theta'_{k+1} \simeq f\delta$. Otherwise $f \notin \mathrm{FV}(\mathcal{X}_k)$ and $f \in \mathrm{Dom}(\varepsilon_k)$, hence $f\pi_{k+1}\theta'_{k+1} = f\pi\varepsilon'_k \simeq f\delta$. We have proved that $\pi_{k+1}\theta'_{k+1} \simeq \delta[\mathrm{FV}(\mathcal{X}_{k+1}) \cap \mathrm{Dom}(\theta_{k+1})]$.

For all f in $\mathrm{FV}(\mathcal{X}_{k+1}) \cap \mathrm{Dom}(\theta'_{k+1})\pi^{-1}_{k+1}$ we have $f\pi_{k+1} \in \mathrm{Dom}(\theta'_{k+1}) = \mathrm{Dom}(\theta'_k) \uplus \mathrm{Dom}(\varepsilon'_k)$. If $f \in \mathrm{FV}(\mathcal{X}_k)$ then $f\pi_{k+1} = f\pi_k \in \mathrm{Dom}(\theta'_k)$ hence by i.h. $f \in \mathrm{Dom}(\theta_k) \subseteq \mathrm{Dom}(\theta_{k+1})$. Otherwise $f\pi_{k+1} = f\pi \in \mathrm{Dom}(\varepsilon'_k)$ hence $f \in \mathrm{Dom}(\varepsilon_k) \subseteq \mathrm{Dom}(\theta_{k+1})$. We have proved that $\mathrm{FV}(\mathcal{X}_{k+1}) \cap \mathrm{Dom}(\theta'_{k+1})\pi^{-1}_{k+1} \subseteq \mathrm{Dom}(\theta_{k+1})$, which completes the induction. □

Theorem 18. $\mathsf{JP}_\mathcal{X}$ *is weakly projective for any constraint* $\mathcal{X}$.

Proof. By Lemma 17 there is a l-run of $\mathsf{JP}_\mathcal{X}$ on γ that ends on $\langle \mathcal{X}_{l+1}\pi_{l+1}, \sigma'_{l+1}, \theta'_{l+1}\rangle$ where π_{l+1} is a permutation such that $\pi_{l+1}\sigma'_{l+1} = \sigma_{l+1}\pi_{l+1}$. This run is complete since $\mathcal{X}_{l+1} = \top$, hence we let γ' be the restriction of σ'_{l+1} to $\mathrm{FV}(\mathcal{X})$, so that $\gamma\ \mathsf{JP}_\mathcal{X}\ \gamma'$. We have $\gamma' = \pi_{l+1}\sigma'_{l+1}[\mathrm{FV}(\mathcal{X})] = \sigma_{l+1}\pi_{l+1}[\mathrm{FV}(\mathcal{X})] = \gamma\pi_{l+1}[\mathrm{FV}(\mathcal{X})]$, hence γ' is a variant of γ. □

It is easy to see that the run of $\mathsf{BP}_\mathcal{X}$ on θ in Example 5 cannot be reproduced on γ_2, hence the technique used in the previous proof does not extend to BP. Yet a longer run on γ_2 does yield a variant of γ_2, which leaves open the question whether $\mathsf{BP}_\mathcal{X}$ is weakly projective.

If we apply $\mathsf{BP}_\mathcal{X}$ after $\mathsf{JP}_\mathcal{X}$ with the $\mathcal{X}$ and θ of Example 4 we only get a variant of γ_1. If we rather apply $\mathsf{BP}_\mathcal{X} \circ \mathsf{JP}_\mathcal{X}$ we get a $\gamma_3 = [\,\lambda u.h_3(h_4(u),\, h_4(x)).\,/f, \lambda u.h_3(u,u).\,/g, h_4(x)\,/y]$, which is not a variant of γ_2, hence this option seems preferable.

6 Conclusion

We believe that the results of Section 3 provide a framework in which cgas can be better designed and understood. We could for instance mix the rules from the two algorithms, provided completeness and termination are preserved (the former is perserved by adding rules, the latter by removing them). The rules can also be modified, for instance the repetition rule is interesting for increasing the arity of the variables but needs to be restricted in some way to ensure termination: here the number of occurrences of bound variables is the limiting factor, but others are possible. A prolog implementation is under way.

The questions asked and partially answered in Sections 4 and 5 are natural and could be addressed for any cga. However, it would be more interesting to compare the two algorithms

w.r.t. the generalizing order $\lesssim$ rather than variance, if possible. Another problem is the control on the rules: it seems hardly possible to consider that some generalization is necessarily better than another one (if they are not comparable with $\lesssim$), their interest can only be compared with some purpose in mind. This raises the question of guiding the search for a generalization toward some specific goal, a task that could easily prove undecidable.

References

[1] Alexander Baumgartner, Temur Kutsia, Jordi Levy, and Mateu Villaret. Higher-order pattern anti-unification in linear time. *J. Autom. Reasoning*, 58(2):293–310, 2017.

[2] Thierry Boy de la Tour and Nicolas Peltier. Proof generalization in LK by second order unifier minimization. *Journal of Automated Reasoning*, 57(3):245–280, October 2016.

[3] N. G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indag Math.*, 34(5):381–392, 1972.

[4] Gilles Dowek. Higher-order unification and matching. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 16, pages 1009–1062. Elsevier Science, New York, 2001.

[5] R. O. Gandy. An early proof of normalization by A.M. Turing. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 453–455. Academic Press, 1980.

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

[7] Kouichi Hirata, Takeshi Ogawa, and Masateru Harao. Generalization algorithms for second-order terms. In Rui Camacho, Ross D. King, and Ashwin Srinivasan, editors, *Inductive Logic Programming, 14th International Conference*, volume 3194 of *Lecture Notes in Computer Science*, pages 147–163. Springer, 2004.

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

[9] D. C. Jensen and T. Pietrzykowski. Mechanizing ω-order type theory through unification. *Theoretical Computer Science*, 3(2):123–171, November 1976.

[10] J.-L. Lassez, M. Maher, and K. Marriot. Unification revisited. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 67–113. Morgan-Kaufman, 1988.

[11] T. M. Mitchell. Generalization as Search. *Artificial Intelligence*, 18:203–226, 1982.

[12] Frank Pfenning. Unification and anti-unification in the Calculus of Constructions. In *Sixth Annual IEEE Symposium on Logic in Computer Science*, pages 74–85, Amsterdam, The Netherlands, July 1991.

[13] Tomasz Pietrzykowski. A complete mechanization of second-order type theory. *J. ACM*, 20(2):333–364, 1973.