Generalizing Rules via Algebraic Constraints
(Extended Abstract)

Alexander V. Gheorghiu David J. Pym

University College London, London WC1E 6BT, United Kingdom

Logics are often characterized by proof systems that are composed of rules. These rules give meaning to the logic — see, for example, proof-theoretic semantics [10]. We [6] propose a framework called *generalizing rules via algebraic constraints* (GRvAC), within which a rule may be decomposed into another rule together with some constraints over an algebra. The effect on the logic as a whole is more easily understood in the other direction: one enriches a logic $\mathfrak{L}$ with an algebra $\mathcal{A}$ to form a presentation of another logic $\mathfrak{L}'$. In short, we make precise the meaning of equations of the following form:

$$\text{Proof in } \mathfrak{L}' = \text{Proof in } \mathfrak{L} + \text{Algebra of Constraints } \mathcal{A}$$

By doing reasoning in $\mathfrak{L}$ enriched by $\mathcal{A}$, one recovers reasoning in $\mathfrak{L}'$ through a transformation that is parameterized by solutions to the algebraic constraints. Consequently, $\mathfrak{L}$ is thought of as more *general* than $\mathfrak{L}'$. More precisely, one begins by labling the syntax of $\mathfrak{L}$ by (a syntax for) $\mathcal{A}$ so that assignments I of the variables of $\mathcal{A}$ determine valuations ν_I mapping the syntax of $\mathfrak{L}$ enriched by $\mathcal{A}$ to the syntax of $\mathfrak{L}'$. A rule of $\mathfrak{L}'$ is *generalized* when a rule of $\mathfrak{L}$ (taken over the enriched language) with constraints (i.e., equations) over $\mathcal{A}$ is used to express it.

As an example, consider the *resource-distribution via boolean constraints* (RDvBC) mechanism introduced by Harland and Pym [8], of which GRvAC framework is an abstraction. The RDvBC mechanism was introduced for the study of proof-search in the presence of multiplicative (or intensional) connectives, such as for proof-search in linear logic (LL). One labels the formulas of LL with a syntax for boolean algebra $\mathcal{B}$ (e.g., one has formulas $\phi \cdot x, \psi \cdot \bar{x}$ in which ϕ and ψ are formulas of LL, x is a boolean variable, and $\bar{x}$ is its negation) such that assignments I determine valuations ν_I that keep formulas labelled by variables that I map to 1 and delete formulas labelled by variables that I map to 0 (e.g., if $I(x) = 0$, then $\nu_I(\{\phi \cdot x, \psi \cdot \bar{x}\}) = \{\phi\}$). This setup allows multiplicative rules to be *generalized* to additive rules; for example,

$$\frac{\Gamma \vdash \phi \quad \Delta \vdash \psi}{\Gamma, \Delta \vdash \phi \otimes \psi} \quad \text{generalizes to} \quad \frac{\Gamma \cdot x, \Delta \cdot \bar{x} \vdash \phi \quad \Gamma \cdot x, \Delta \cdot \bar{x} \vdash \psi}{\Gamma, \Delta \vdash \phi \otimes \psi}$$

In terms of GRvAC, this witnesses the following equation in which LL is a proof system for linear logic and LK is a proof system for classical logic: $\mathsf{LL} = \mathsf{LK} + \mathcal{B}$. Other examples of GRvAC are present in the literature too; for example, algebraic constraints may be used for unification in logic programming, which can be understood as saying that propositional logic is more general than predicate logic — see [5] for details.

Though generalization allows one to relate two logics, the idea of algebraic constraints is useful in itself and present elsewhere in the literature — see, for example, work by Negri [9] on relational calculi. Indeed, the concept of enrichment here is strongly related to the framework of *Labelled Deductive Systems* introduced by Gabbay [4].

The GRvAC framework is useful both for the theory and practice of logic. In theory, it is a technology that allows one to express formally relationships *between* logics; for example, it supports the folkore that classical logic (CL) is a combinatorial core of logics (i.e., CL generalizes most logics). It also allows one to study metatheory for particular logics; for example, the GRvAC framework allows one to translate between nested systems, tableaux systems, and relational calculi for normal modal logics, thereby proving soundness and completeness of all by proving it for one [6] (i.e., if there is a proof witnessing a sequent in one system, then immediately there is a proof witnessing the sequent in the other systems).

By understanding how a logic arises from CL by means of an algebra, GRvAC allows one to derive model-theoretic semantics for the logic; and, conversely, it allows one to generate sound and complete proof systems for a logic from a frame semantics. The semantic uses of GRvAC are prefigured by Docherty [3]. An example of the effectiveness of the GRvAC framework for metatheory is captured by a case study on intuitionistic logic (IL). Here, GRvAC allow one to construct from a single-conclusioned calculus a multiple-conclusioned sequent calculus, which witnesses that CL is the combinatorial core of IL. By studying the new calculus' relationship to CL using GRvAC, one can *derive* a model-theoretic semantics of IL. The derivations provides a new technique for proving soundness and completeness that proceeds by showing the equivalence of proof-search of the two logics relative to the constraints captured by the algebra — see [7] for further discussion.

The practical uses of GRvAC are in proof-search (including algorthmic). This claim is justified by the examples above; that is, RDvBC concerns the context-management problem during proof-search in substructural logics, the multiple-conclusion system for IL is a powerful tool for doing proof-search with backtracking. In general, GRvAC allows one to separate the combinatorial aspects of a logic from the internal choices made during proof-search; that is, the combinatorial aspects of proof-search in $\mathfrak{L}'$ can be understood by proof-search in $\mathfrak{L}$ with controls governed by constraints over $\mathcal{A}$. Among other things, therefore, GRvAC allows one to capture certain amount of global reasoning during proof-search, which can be interpreted as capturing a certain amount of backtracking within a proof system.

To elucidate the usefulness of GRvAC in proof-search, we illustrate its application to quantifiers. This captures earlier work by Wallen [11], Andrews [1], and Bibel [2]. Consider the putative conclusion $\exists x \forall y Pxy \vdash \forall u \exists v Puv$ in classical first-order logic (FOL). Two proof-search attempts are as follows:

$$
\dfrac{\dfrac{P(a,b) \vdash \forall u \exists v P(u,v)}{\forall y P(a,y) \vdash \forall u \exists v P(u,v)}\ \forall_{\mathsf{L}}}{\exists x \forall y P(x,y) \vdash \forall u \exists v P(u,v)}\ \exists_{\mathsf{L}}
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{Pab \vdash Pab}{Pab \vdash \exists u Pvb}\ \exists_{\mathsf{R}}}{\forall y Pay \vdash \exists u Pvb}\ \forall_{\mathsf{L}}}{\forall y Pay \vdash \forall u \exists v Pvu}\ \forall_{\mathsf{R}}}{\exists x \forall y Pxy \vdash \forall u \exists v Pvu}\ \exists_{\mathsf{L}}
$$

The first proof-search fails and the second succeeds. Why does the first fail? The GRvAC framework may be used to understand these proof-searches. One can generalize the quantifier rules so as not to commit to a substitution, but rather track that some substitution needs to be made, together with its conditions; for example, one has a computation of the following form:

$$
\frac{\dfrac{\dfrac{\bigparr_{A\in\Gamma}\bigparr_{B\in\Delta}\ell(A)=\ell(B)}{\Gamma\vdash\Delta}}{\Gamma,\phi[x\mapsto x\cdot n]\vdash\Delta\quad\kappa}}{\Gamma,\exists x\phi\vdash\Delta}
\qquad
\frac{\Gamma\vdash\phi[x\mapsto x\cdot n],\Delta\quad\kappa}{\Gamma\vdash\forall x\phi,\Delta}
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{n=a\quad m=b}{P(x\cdot n,b)\vdash P(a,u\cdot m)}}{P(x\cdot n,b)\vdash\exists vP(v,u\cdot m)}\quad m\neq n,b}{P(x\cdot n,b)\vdash\exists vP(v,u\cdot m)}}{\forall yP(x\cdot n,y)\vdash\forall u\exists vP(v,u)}}{\exists x\forall yP(x,y)\vdash\forall u\exists vP(v,u)}
$$

— the constraint κ expresses that n is not any term or label that appears in Γ,ϕ or Δ, the notation $\bigparr_{x\in X}\kappa_x$ denotes a meta-disjunction over constraints κ_x, for each $x\in X$, and $\ell(\phi)$ is a list of the labels occurring in ϕ.

The insolubility of the constraints $n=a$, $m\neq n,b$ and $m=b$ means that there is no interpretation of the proof structure as a proof. Nonetheless, the constraints give information about why the reduction fails that may be leveraged through some *global* reasoning to yield a successful proof-search. There is no purpose in permuting the rules producing m and n with each other as in either case one would have $m\neq n$, but the substitutions producing a and b are free of constraints, hence one can permute the rule producing b with the rule producing m, thereby eliminating the constraint $m\neq b$. The result is a coherent set of constraints whose solution determines the successful proof-search attempt.

The GRvAC framework allows one to express complex rules as simple rules together with algebraic constraints that recover the former from the latter by means of transformations parameterized by solutions to equations over the algebra. It is useful for intra-logic metatheory (i.e., proof theory and semantics), for inter-logic metatheory (i.e., connexions between logics), and in applied logic tasks involving proof-search. Though we have outlined it conceptionally, substantial work remains in developing the space of examples and using it to develop uniform approaches to metatheory. Moreover, on the question of proof-search, GRvAC may be used to give a general mathematical theory of control, which is currently lacking, and relate the control problems of proof-search to other aspects of the logic (e.g., the clauses of its semantics).

References

[1] Andrews, P. B. Theorem Proving via General Matings. *Journal of the ACM 28*, 2 (1981), 193–214.

[2] Bibel, W. *Automated Theorem Proving.* Springer, 1982.

[3] Docherty, S., and Pym, D. J. Modular Tableaux Calculi for Separation Theories. In *Foundations of Software Science and Computation Structures — FOSSACS* (2018), C. Baier and U. Dal Lago, Eds., pp. 441–458.

[4] Gabbay, D. M. *Labelled Deductive Systems.* Oxford University Press, 1996.

[5] Gheorghiu, A. V., Docherty, S., and Pym, D. J. Reductive Logic, Coalgebra, and Proof-search: A Perspective from Resource Semantics. In *Samson Abramsky on Logic and Structure in Computer Science and Beyond*, A. Palmigiano and M. Sadrzadeh, Eds., Springer Outstanding Contributions to Logic Series. Springer, 2021. In press, 2022.

[6] Gheorghiu, A. V., and Pym, D. J. Generalizing Rules via Algebraic Constraints. `http://www0.cs.ucl.ac.uk/staff/D.Pym/grvac.pdf`, 2022. Submitted.

[7] Gheorghiu, A. V., and Pym, D. J. Semantics ex Proof and Refutation. `http://www0.cs.ucl.ac.uk/staff/D.Pym/semantics-ex-proof-ref.pdf`, 2022. Presented at *Bilateralism & Proof-Theoretic Semantics*, Ruhr University Bochum, March 17-18, 2022. `https://sites.google.com/view/conferencebilateralism2022/home`

[8] Harland, J., and Pym, D. J. Resource-distribution via Boolean Constraints. *ACM Transactions on Computational Logic 4* (2003), 56–90.

[9] Negri, S. Proof Analysis in Modal Logic. *Journal of Philosophical Logic 34*, 5 (2005), 507–544.

[10] Schroeder-Heister, P. Proof-Theoretic Semantics. In *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2018.

[11] Wallen, L. A. *Automated Proof Search in non-Classical Logics: Efficient Matrix Proof Methods for Modal and Intuitionistic Logics.* PhD thesis, The University of Edinburgh, 1987.