May I Borrow Your Logic?*

Maura Cerioli[1] and José Meseguer[2]

[1] DISI–Dipartimento di Informatica e Scienze dell'Informazione, Università di Genova, Viale Benedetto XV, 16132 Genova, Italy, e-mail: cerioli@disi.unige.it

[2] SRI International, Menlo Park, CA 94025, and Center for the Study of Language and Information, Stanford University, Stanford, CA 94305 e-mail: meseguer@csl.sri.com

Abstract. It can be very advantageous to borrow key components of a logic for use in another logic. The advantages may be not only conceptual; due to the existence of software systems supporting mechanized reasoning in a given logic, it may be possible to reuse a system developed for one logic—for example, a theorem-prover—to obtain a new system for another. Translations between logics by appropriate mappings provide a first way of reusing tools of one logic in another. This paper generalizes this idea to the case where entire components—for example, the proof theory—of one of the logics involved may be completely missing, so that the appropriate mapping could not even be defined. The idea then is to borrow the missing components (as well as their associated tools if they exist) from a logic that has them in order to create the full-fledged logic and tools that we desire. The relevant structure is transported using maps that only involve a limited aspect of the two logics in question—for example, their model theory. The constructions accomplishing this kind of borrowing of logical structure are very general and simple. They only depend upon a few abstract properties that hold under very general conditions given a pair of categories linked by adjoint functors.

1 Introduction

The use of logic in computer science is undergoing vigorous growth. Since the applications are many, there are increasingly stronger interactions between the two fields that are having a profound impact on both of them. New logics are frequently been proposed, and new variants or adaptations of existing logics for new purposes are widespread.

* Partially supported by Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of C.N.R. (Italy), MURST-40% Modelli e Specifiche di Sistemi Concorrenti, by the US Office of Naval Research under contracts N00014-90-C-0086 and N00014-92-C-0518, and by the Information Technology Promotion Agency, Japan, as a part of the R & D of Basic Technology for Future Industries "New Models for Software Architecture" sponsored by NEDO (New Energy and Industrial Technology Development Organization).

This proliferation of logics—although certainly a sign of vitality and intellectual creativity—brings with it important conceptual challenges. In a sense, each logic is a different language and, as in the case of natural languages, there is often a serious need to bridge the gap between different languages by means of appropriate translations, and the danger of serious confusion when translations are not correct. There is also a related need to understand the essential features shared by logics in general so that systematic methods can be developed to deal with these problems.

In computer science, the conceptual needs posed by the proliferation of logics were first addressed by Goguen and Burstall [8], who proposed their theory of *institutions* as a general framework for logics. The work on institutions has been further developed by their original proponents and by others [9, 10, 22, 23], and has influenced other notions proposed by different authors [15, 19, 7, 16, 13, 20, 5, 2]. Some of the notions proposed are closely related to institutions; however, in other cases the main intent is to substantially expand the primarily model-theoretic viewpoint provided by institutions to give an adequate treatment of proof-theoretic aspects such as entailment and proof structures.

Institutions arose out of work on the Clear specification language [3], in which the goal was to provide powerful modularity and parameterization mechanisms to structure and reuse formal specifications. Such reusability techniques have later been applied to a good number of specification and logical programming languages such as, for example, [12, 6, 21, 11, 17]. However, the need for reusability arises not only inside one logic—so that specifications or logical programs written in that logic can be reused—but also at the metalevel, in the sense that it can be greatly advantageous to reuse entire logics, or key components of such logics. The advantages may be not only conceptual, although of course this is important; due to the existence of software systems supporting mechanized reasoning in a given logic, it may be possible to reuse a system developed for one logic—for example, a theorem-prover—to obtain a new system for another.

Translations between logics by appropriate mappings—especially if they are *conservative* in the sense of [16]—may provide a first way of reusing tools of one logic in another, by translating the appropriate sentences or proofs and using the original tool on the translations. This paper generalizes this idea to the case where entire components—for example, the proof theory—of one of the logics involved may be completely missing, so that the appropriate mapping could not even be defined. The idea then is to borrow the missing components (as well as their associated tools if they exist) from a logic that has them in order to create, *ex nihilo* as it were, the full-fledged logic and tools that we desire. The relevant structure is transported using maps that only involve a limited aspect of the two logics in question—for example, their model theory.

The constructions accomplishing this kind of borrowing of logical structure are very general and simple. We show that they only depend upon a few abstract properties that hold under very general conditions given a pair of categories linked by adjoint functors. Therefore, the constructions capitalize on the fact that, as was shown in [16], the different components of a logic—entailment relation, model theory, and proof theory—are in a very precise technical sense *modular*, namely in that they can be added or deleted by means of constructions that are adjoint functors.

Specifically, we show that, given two institutions—i.e., the model-theoretic components of two logics—$\mathcal{I}$ and $\mathcal{I}'$ and a map of institutions $\mathcal{I} \longrightarrow \mathcal{I}'$, we can borrow the entailment relation or the proof theory (or both things) from $\mathcal{I}'$ if they exist to use them for $\mathcal{I}$, and that completeness, if it holds, is preserved by the borrowing. Similarly, given two entailment systems $\mathcal{E}$ and $\mathcal{E}'$—i.e., the provability relations of two logics—and a map of entailment systems $\mathcal{E} \longrightarrow \mathcal{E}'$, we show that we can borrow the proof theory from $\mathcal{E}'$ if it exists to endow $\mathcal{E}$ with a proof theory. Finally, given two logics $\mathcal{L}$ and $\mathcal{L}'$—i.e., the provability relations and the model-theoretic components of two logics—and a map of logics $\mathcal{L} \longrightarrow \mathcal{L}'$, we show that we can borrow the proof theory from $\mathcal{L}'$ if it exists to endow $\mathcal{L}$ with a proof theory. Due to space limitations neither proofs nor examples are given; they can be found in [4].

2 General Logics

Since the significant examples of application of the general categorical construction are all largely based on the concepts of *institution* [8] and *general logic* [16], this section is devoted to recall some basic definitions and results from these theories. For a detailed discussion of these concepts see [10, 16].

Institutions cover the semantic aspects of a logical framework, providing formal counterparts for the notions of *signature*, *sentences*, *models* and *validity*, while entailment systems deal with the deductive part. Putting together an institution and a compatible entailment system, i.e. an entailment system that is sound w.r.t. the institution, a *logic* is obtained, where tools to deal with provability and with model theoretic aspects are both at hand. But, since entailment systems disregard the computational aspects, the concept of (structured) proofs is not formalized; thus *proof calculi* are introduced to cover also this feature.

Definition 1. [16] An *entailment system* $\mathcal{E} = (\mathbf{Sign}, Sen, \vdash)$ consists of a category $\mathbf{Sign}$ of *signatures*, a functor $Sen\colon \mathbf{Sign} \to \mathbf{Set}$ giving the set of *sentences* over a given signature, and for each Σ in $\mathbf{Sign}$ a binary relation $\vdash_\Sigma \subseteq \wp(Sen(\Sigma)) \times Sen(\Sigma)$ called *Σ-entailment* satisfying the following properties:

reflexivity: for any $\phi \in Sen(\Sigma)$, $\{\phi\} \vdash_\Sigma \phi$;
monotonicity: if $\Gamma \vdash_\Sigma \phi$ and $\Gamma \subseteq \Gamma'$, then $\Gamma' \vdash_\Sigma \phi$;
transitivity: if $\Gamma \vdash_\Sigma \phi_i$ for all $i \in I$ and $\Gamma \cup \{\phi_i \mid i \in I\} \vdash_\Sigma \psi$, then $\Gamma \vdash_\Sigma \psi$;
$\vdash$-*translation*: if $\Gamma \vdash_\Sigma \phi$, then for any $\sigma : \Sigma \to \Sigma'$ in $\mathbf{Sign}$, $Sen(\sigma)(\Gamma) \vdash_{\Sigma'} Sen(\sigma)(\phi)$. □

Definition 2. [8] An *institution* $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ consists of a category $\mathbf{Sign}$ of *signatures*, a functor $Sen\colon \mathbf{Sign} \to \mathbf{Set}$ giving the set of *sentences* over a given signature, a functor $Mod\colon \mathbf{Sign}^{op} \to \mathbf{Cat}$ giving the category of *models* of a given signature, and for each Σ in $\mathbf{Sign}$ a *satisfaction* relation

$\models_\Sigma \subseteq |Mod(\Sigma)| \times Sen(\Sigma)$, such that for each morphism $\sigma: \Sigma \to \Sigma'$ in **Sign**, the *Satisfaction Condition*

$$M' \models_{\Sigma'} Sen(\sigma)(\xi) \iff Mod(\sigma)(M') \models_\Sigma \xi$$

holds for each $M' \in |Mod(\Sigma')|$ and each $\xi \in Sen(\Sigma)$. □

Given an entailment system (respectively, an institution) its category $\mathbf{Th}_0$ of *theories* has as objects pairs $T = (\Sigma, \Gamma)$ with Σ a signature and Γ a set of sentences on Σ, and as morphisms $\sigma: (\Sigma, \Gamma) \to (\Sigma', \Gamma')$ the signature morphisms $\sigma: \Sigma \to \Sigma'$ s.t. $Sen(\sigma)(\Gamma) \subseteq \Gamma'$. Since a set Γ of sentences on Σ determines the full subcategory of models that satisfy all the sentences Γ, it is easy to show that the functor Mod extends to a functor $Mod: \mathbf{Th}_0{}^{op} \to \mathbf{Cat}$. Similarly, by assigning to each theory $T = (\Sigma, \Gamma)$ the sentences on its signature we can extend the functor $Sen: \mathbf{Sign} \to \mathbf{Set}$ to a functor $Sen: \mathbf{Th}_0 \to \mathbf{Set}$.

Definition 3. [16] A *logic* is a 5-tuple $\mathcal{L} = (\mathbf{Sign}, Sen, Mod, \models, \vdash)$ such that $(\mathbf{Sign}, Sen, \vdash)$ is an entailment system, $(\mathbf{Sign}, Sen, Mod, \models)$ is an institution, and for each $\Sigma \in |\mathbf{Sign}|$, $\Gamma \subseteq Sen(\Sigma)$ and $\phi \in Sen(\Sigma)$, $\Gamma \vdash_\Sigma \phi \Rightarrow \Gamma \vdash_{\models_\Sigma} \phi$, where $\Gamma \vdash_{\models_\Sigma} \phi$ iff ($M \models_\Sigma \gamma$ for all $\gamma \in \Gamma$) implies $M \models_\Sigma \phi$. □

Definition 4. [16] A *proof calculus* is a 6-tuple $\mathcal{P} = (\mathbf{Sign}, Sen, \vdash, P, \mathrm{Pr}, \pi)$ such that: $(\mathbf{Sign}, Sen, \vdash)$ is an entailment system, $P: \mathbf{Th}_0 \to \mathbf{Struct}_\mathcal{P}$ is a functor (for each theory T, the object $P(T) \in \mathbf{Struct}_\mathcal{P}$ is called its *proof-theoretic structure*), $\mathrm{Pr}: \mathbf{Struct}_\mathcal{P} \to \mathbf{Set}$ is a functor (for each theory T, the set $\mathrm{Pr}(P(T))$ is called its *set of proofs*, and *proofs* denotes the composite functor $\mathrm{Pr} \cdot P: \mathbf{Th}_0 \to \mathbf{Set}$), $\pi : proofs \Rightarrow Sen$ is a natural transformation, such that for each theory $T = (\Sigma, \Gamma)$, the image of $\pi_T: proofs(T) \to Sen(T)$ is the set $\Gamma^\bullet$ of all sentences ϕ s.t. $\Gamma \vdash \phi$. □

Definition 5. [16] The 8-tuple $\mathcal{S} = (\mathbf{Sign}, Sen, Mod, \vdash, \models, P, \mathrm{Pr}, \pi)$ is a *logical system* if $(\mathbf{Sign}, Sen, Mod, \models, \vdash)$ is a logic, and $(\mathbf{Sign}, Sen, \vdash, P, \mathrm{Pr}, \pi)$ is a proof calculus. □

In [16] maps of entailment systems, institutions, logics, proof calculi, and logical systems are defined and are illustrated with examples; here the definitions are presented in summarized form.

Definition 6. [16] For $\mathcal{E} = (\mathbf{Sign}, Sen, \vdash)$ and $\mathcal{E}' = (\mathbf{Sign}', Sen', \vdash')$ entailment systems, a *map of entailment systems* $(\Phi, \alpha): \mathcal{E} \to \mathcal{E}'$ consists of a natural transformation $\alpha: Sen \Rightarrow Sen' \cdot \Phi$ and an α-sensible functor[3] $\Phi: \mathbf{Th}_0 \to \mathbf{Th'}_0$ satisfying the following property:

$$\Gamma \vdash_\Sigma \phi \;\Rightarrow\; \Gamma' \cup \alpha_\Sigma(\Gamma) \vdash'_{\Sigma'} \alpha_\Sigma(\phi),$$

where, by convention, $(\Sigma', \Gamma') = \Phi(\Sigma, \Gamma)$.

[3] We refer to [16] for the detailed definition of α-sensible functors.

For $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ and $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$ institutions, a *map of institutions* $(\Phi, \alpha, \beta) \colon \mathcal{I} \to \mathcal{I}'$ consists of an α-sensible functor $\Phi \colon \mathbf{Th}_0 \to \mathbf{Th}'_0$, and natural transformations $\alpha : Sen \Rightarrow Sen' \cdot \Phi$, and $\beta : Mod' \cdot \Phi \Rightarrow Mod$ such that for each $\Sigma \in |\mathbf{Sign}|$, each $\phi \in Sen(\Sigma)$, and each $M' \in |Mod'(\Phi(\Sigma, \emptyset))|$ the following property is satisfied:

$$M' \models'_{\Sigma'} \alpha_\Sigma(\phi) \;\Leftrightarrow\; \beta_{(\Sigma,\emptyset)}(M') \models_\Sigma \phi,$$

where Σ' is the signature of the theory $\Phi(\Sigma, \emptyset)$.

Given logics $\mathcal{L}$ and $\mathcal{L}'$, a *map of logics* $(\Phi, \alpha, \beta) \colon \mathcal{L} \to \mathcal{L}'$ is a map $(\Phi, \alpha, \beta) : inst(\mathcal{L}) \to inst(\mathcal{L}')$ of the underlying institutions s.t., in addition, $(\Phi, \alpha) : ent(\mathcal{L}) \to ent(\mathcal{L}')$ is a map of the underlying entailment systems.

Given proof calculi $\mathcal{P}$ and $\mathcal{P}'$, a *map of proof calculi* $(\Phi, \alpha, \gamma) \colon \mathcal{P} \to \mathcal{P}'$ consists of a map $(\Phi, \alpha) \colon ent(\mathcal{P}) \to ent(\mathcal{P}')$ of the underlying entailment systems together with a natural transformation $\gamma \colon proofs \Rightarrow proofs' \cdot \Phi$ such that $\pi'_\Phi \circ \gamma = \alpha \circ \pi$.

Given logical systems $\mathcal{S}$ and $\mathcal{S}'$, a *map of logical systems* $(\Phi, \alpha, \beta, \gamma) \colon \mathcal{S} \to \mathcal{S}'$ consists of a map of the underlying logics $(\Phi, \alpha, \beta) \colon log(\mathcal{S}) \to log(\mathcal{S}')$ and a map of the underlying proof calculi $(\Phi, \alpha, \gamma) \colon pcalc(\mathcal{S}) \to pcalc(\mathcal{S}')$. □

The relationships among the categories $\underline{Ent}$ of entailment systems, $\underline{Inst}$ of institutions, $\underline{Log}$ of logics, $\underline{PCalc}$ of proof calculi, and $\underline{LogSys}$ of logical systems can be illustrated by the following diagram, where all the arrows depicted are forgetful functors that "throw away" appropriate components of a logic. For example, the functor *inst* maps a logic $\mathcal{L} = (\mathbf{Sign}, Sen, Mod, \models, \vdash)$ to the institution $(\mathbf{Sign}, Sen, Mod, \models)$.

$$\begin{tikzcd}
 & & \underline{LogSys} \arrow[dl, "log"'] \arrow[dr, "pcalc"] & \\
 & \underline{Log} \arrow[dl, "inst"'] \arrow[dr, "ent_{Log}"] & & \underline{PCalc} \arrow[dl, "ent"] \\
\underline{Inst} & & \underline{Ent} &
\end{tikzcd}$$

Proposition 7. *[16]*

1. *The functor* $(\_)^+ \colon \underline{Inst} \to \underline{Log}$, *mapping* $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ *to* $\mathcal{I}^+ = (\mathbf{Sign}, Sen, Mod, \models, \vdash_\models)$, *where* $\Gamma \vdash_\models \phi$ *iff (*$M \models_\Sigma \gamma$ *for all* $\gamma \in \Gamma$*) implies* $M \models_\Sigma \phi$, *and* ρ *to* $\rho^+ = \rho$ *for any map* ρ, *is the right adjoint of the forgetful functor* $inst \colon \underline{Log} \to \underline{Inst}$. *The unit of the adjunction for a logic* $\mathcal{L} = (\mathbf{Sign}, Sen, Mod, \models, \vdash)$ *is the map* $(Id_{\mathbf{Sign}}, Id_{Sen}, Id_{Mod})$.
2. *The functor* $(\_)^\sharp \colon \underline{Ent} \to \underline{PCalc}$, *defined by* $\mathcal{E}^\sharp = (\mathbf{Sign}, Sen, \vdash, thm, Id_{\mathbf{Set}}, j)$ *(where* thm *is the functor sending each theory to the set of its theorems, described in Section 2.2 of [16], and* j *is the natural subfunctor inclusion*

thm $\subseteq$ *Sen*) *and mapping* (Φ, α) *to* $(\Phi, \alpha, \alpha_{|thm})$, *where* $\alpha_{|thm}$ *is the restriction of* α *to the theorems of the domain, is the right adjoint of the forgetful functor* $ent: \underline{PCalc} \to \underline{Ent}$. *The unit of the adjunction for a proof calculus* $\mathcal{P} = (\mathbf{Sign}, Sen, \vdash, P, \mathrm{Pr}, \pi)$ *is the map* $(Id_{\mathbf{Sign}}, Id_{Sen}, \pi)$, *where* π *is now viewed as a natural transformation* $\pi: proofs \Rightarrow thm$.

3. *The functor* $(\_)^{\diamond}: \underline{Log} \to \underline{LogSys}$, *mapping* $\mathcal{L} = (\mathbf{Sign}, Sen, Mod, \models, \vdash)$ *to* $\mathcal{L}^{\diamond} = (\mathbf{Sign}, Sen, Mod, \vdash, \models, thm, Id_{\mathbf{Set}}, j)$, *and* (Φ, α, β) *to* $(\Phi, \alpha, \beta, \alpha_{|thm})$, *where* $\alpha_{|thm}$ *is the restriction of* α *to the theorems of the domain, is the right adjoint of the forgetful functor* $log: \underline{LogSys} \to \underline{Log}$. *The unit of the adjunction for a logical system* $\mathcal{S} = (\mathbf{Sign}, Sen, Mod, \vdash, \models, P, \mathrm{Pr}, \pi)$ *is the map* $(Id_{\mathbf{Sign}}, Id_{Sen}, Id_{Mod}, \pi)$, *where* π *is now viewed as a natural transformation* $\pi: proofs \Rightarrow thm$.
4. *The functor* $(\_)^{\flat}: \underline{Inst} \to \underline{LogSys}$ *mapping* $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ *to* $\mathcal{I}^{\flat} = (\mathbf{Sign}, Sen, Mod, \vdash_{\models}, thm, Id_{\mathbf{Set}}, j)$, *and* (Φ, α, β) *to* $(\Phi, \alpha, \beta, \alpha_{|thm})$, *is the right adjoint of the forgetful functor* $inst: \underline{LogSys} \to \underline{Inst}$. *The unit of the adjunction for a logical system* $\mathcal{S} = (\mathbf{Sign}, Sen, Mod, \vdash, \models, P, \mathrm{Pr}, \pi)$ *is the map* $(Id_{\mathbf{Sign}}, Id_{Sen}, Id_{Mod}, \pi)$. $\square$

3 Transporting Structures Across Categories

The transportation of structure that we study will involve two categories of logical structures, $\mathbf{C}$ and $\mathbf{D}$, and a couple of functors $U: \mathbf{D} \to \mathbf{C}$ and $R: \mathbf{C} \to \mathbf{D}$ with R right adjoint to U. In the applications that we will consider U, in spite of being a left adjoint, will have the flavor of a forgetful functor. The basic construction applies to arbitrary categories $\mathbf{C}$ and $\mathbf{D}$ with functors U and R as above, and consists in the process of transporting to an object $C \in |\mathbf{C}|$ the structure of an object $D \in |\mathbf{D}|$ via a map $c: C \to U(D)$. The transported structure is enjoyed by an object $\widetilde{C} \in |\mathbf{D}|$ which, roughly speaking, corresponds to C plus the features of D translated by c.

Theorem 8. *Let* $R: \mathbf{C} \to \mathbf{D}$ *be right adjoint to* $U: \mathbf{D} \to \mathbf{C}$, *with* $\eta: 1_{\mathbf{D}} \to R \cdot U$ *the unit of the adjunction, and let us denote by* $\mathbf{D}'$ *the comma category* $\mathbf{D} \downarrow 1_{\mathbf{D}}$ *(see e.g. [14]) and by* $\mathbf{C}'$ *the comma category* $\mathbf{C} \downarrow U$. *If for each object* $C \in |\mathbf{C}|$, *each object* $D \in |\mathbf{D}|$, *and each arrow* $c \in \mathbf{C}(C, U(D))$ *the pullback of* $R(c)$ *and* η_D *exists, then there is a functor* $R': \mathbf{C}' \to \mathbf{D}'$, *defined as follows:*

- *for each object* $(c: C \to U(D), D) \in |\mathbf{C}'|$, *the object* $R'(c, D) = (\widetilde{c}: \widetilde{C} \to D)$ *is defined by the following pullback:*

$$\begin{tikzcd} R(C) \arrow[r, "R(c)"] & R \cdot U(D) \\ \widetilde{C} \arrow[u, "j_{c,D}"] \arrow[r, "\widetilde{c}"'] & D \arrow[u, "\eta_D"'] \end{tikzcd}$$

– *for each arrow* $(g, g') \in \mathbf{C}'((c_1 : C_1 \to U(D_1), D_1), (c_2 : C_2 \to U(D_2), D_2))$ *the arrow* $R'(g, g') = (g^\sharp, g')$*, where* $g^\sharp$ *is the unique arrow* $g^\sharp : \widetilde{C_1} \to \widetilde{C_2}$ *such that:*

  1. $j_{c_2,D_2} \cdot g^\sharp = R(g) \cdot j_{c_1,D_1}$*;*
  2. $\widetilde{c_2} \cdot g^\sharp = g' \cdot \widetilde{c_1}$*.*

If such a functor R' *exists, then* $\mathbf{C}$ *is said to* admit generalization under R and U. $\square$

Theorem 9. *Under the same hypotheses as in Thm. 8,* R' *is the right adjoint of the functor* $U' : \mathbf{D}' \to \mathbf{C}'$*, defined as follows:*

– $U'(d : D \to D') = (U(d) : U(D) \to U(D'), D')$ *for each object* $(d : D \to D') \in |\mathbf{D}'|$*;*
– $U'(f, f') = (U(f), f')$ *for each arrow* $(f, f') \in \mathbf{D}'(d_1, d_2)$*.* $\square$

Proposition 10. *Let* $R_1 : \mathbf{C} \to \mathbf{D}$ *be right adjoint to* $U_1 : \mathbf{D} \to \mathbf{C}$*, and let* $R_2 : \mathbf{D} \to \mathbf{E}$ *be right adjoint to* $U_2 : \mathbf{E} \to \mathbf{D}$*. If* $\mathbf{C}$ *admits generalization under* R_1 *and* U_1*, and* $\mathbf{D}$ *admits generalization under* R_2 *and* U_2*, then* $\mathbf{C}$ *admits generalization under* $R_2 \cdot R_1$ *and* $U_1 \cdot U_2$*.* $\square$

4 Endowing an Institution with an Entailment System

We apply here the general result to the adjunction between institutions and logics to borrow an entailment system from another logic using a map of institutions. This borrowing in fact preserves completeness. Although in general $\underline{Log}$ does not have pullbacks, they do exist in the particular case of pulling back the unit of the adjunction along the image under $(\_)^+$ of a map of institutions.

Lemma 11. *Let* $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ *be an institution,* $\mathcal{L}'$ *a logic, and* $\rho : \mathcal{I} \longrightarrow inst(\mathcal{L}')$ *a map of institutions. Denoting by* $\eta_{\mathcal{L}'}$ *the embedding of* $\mathcal{L}'$ *into* $(inst(\mathcal{L}'))^+$*, the following diagram, where* $\mathcal{L} = (\mathbf{Sign}, Sen, Mod, \models, \vdash)$ *and* $\vdash$ *is defined by:* $\Gamma \vdash_\Sigma \phi$ *iff both* $\alpha_\Sigma(\Gamma) \vdash'_{\Phi(\Sigma,\emptyset)} \alpha_\Sigma(\phi)$ *and* $\Gamma \models_\Sigma \phi$*, is a pullback.*

$$\begin{tikzcd}
\mathcal{I}^+ \arrow[r, "\rho^+"] & (inst(\mathcal{L}'))^+ \\
\mathcal{L} \arrow[u, "(Id_{\mathbf{Sign}}, Id_{Sen}, Id_{Mod})"] \arrow[r, "\rho^+"'] & \mathcal{L}' \arrow[u, "\eta_{\mathcal{L}'}"']
\end{tikzcd} \qquad \square$$

Proposition 12. *The category* $\underline{Inst}$ *admits generalization under* $(\_)^+$ *and the forgetful functor* $inst : \underline{Log} \to \underline{Inst}$*.* $\square$

As a direct application of this result one can obtain the construction of a many-sorted entailment system in [18], translating the Birkhoff system along a map of institutions implicitly defined; a more recent application, formulated in a style closer to ours, is the definition of an inference system for partial conditional higher-order types importing an entailment system for first-order conditional types in [1]. A key general feature of the construction is the preservation of completeness.

Proposition 13. *Using the notation of Lemma 11, if $\mathcal{L}'$ is complete, then $\mathcal{L}$ is complete, too.* □

5 Endowing an Entailment System with a Proof Calculus

Let us now apply the general result to the adjunction between entailment systems and proof calculi to show how an informative proof system can be added to an entailment system $\mathcal{E}$ provided we are given a proof calculus $\mathcal{P}$ and a translation of $\mathcal{E}$ into the entailment system underlying $\mathcal{P}$.

Lemma 14. *Given an entailment system $\mathcal{E} = (\mathbf{Sign}, Sen, \vdash)$, a proof calculus $\mathcal{P}' = (\mathbf{Sign}', Sen', \vdash', P', Pr', \pi')$, and a map of entailment systems $(\Phi, \alpha)\colon \mathcal{E} \to ent(\mathcal{P}')$, then the following diagram is a pullback*

$$\begin{tikzcd}
\mathcal{E}^\sharp \arrow[rr, "{(\Phi, \alpha, \alpha_{|thm})}"] & & (ent(\mathcal{P}'))^\sharp \\
\mathcal{P} \arrow[u, "{(Id_{\mathbf{Sign}}, Id_{Sen}, \pi)}"] \arrow[rr, "{(\Phi, \alpha, \gamma)}"'] & & \mathcal{P}' \arrow[u, "{(Id_{\mathbf{Sign}'}, Id_{Sen'}, \pi')}"']
\end{tikzcd}$$

where $\mathcal{P} = (\mathbf{Sign}, Sen, \vdash, P, \mathrm{Pr}, \pi)$, for $P = proofs$ and $Pr = Id_{\mathbf{Set}}$, and π, γ, $proofs$ are defined by the following pullback square:

$$\begin{tikzcd}
thm \arrow[r, "\alpha_{|thm}"] & thm' \cdot \Phi \\
proofs \arrow[u, "\pi"] \arrow[r, "\gamma"'] & proofs' \cdot \Phi \arrow[u, "{\pi'_\Phi}"']
\end{tikzcd} \quad \square$$

Proposition 15. *The category $\underline{Ent}$ admits generalization under the functors $(\_)^\sharp$ and ent.* □

6 Endowing a Logic or an Institution with a Logical System

Extending the constructions for proof calculi to logics, we can build a logical system on top of a logic by borrowing the missing proof calculus from another logical system. Using Prop. 10, we can compose this construction with the one enriching an institution with a logic, in order to get also a general construction enriching an institution with the models and the proof calculus of another logical system.

Lemma 16. *Given a logic* $\mathcal{L} = (\mathbf{Sign}, Sen, Mod, \models, \vdash)$, *a logical system* $\mathcal{S}' = (\mathbf{Sign}', Sen', Mod', \vdash', \models', P', Pr', \pi')$, *and a map of logics* $(\Phi, \alpha, \beta)\colon \mathcal{L} \to log(\mathcal{S}')$, *then the following diagram is a pullback*

$$\begin{tikzcd} \mathcal{L}^{\diamond} \arrow[r, "{(\Phi, \alpha, \beta, \alpha_{|thm})}"] & (log(\mathcal{S}'))^{\diamond} \\ \mathcal{S} \arrow[u, "{(Id, Id, Id, \pi)}"] \arrow[r, "{(\Phi, \alpha, \beta, \gamma)}"'] & \mathcal{S}' \arrow[u, "{(Id, Id, Id, \pi')}"'] \end{tikzcd}$$

where $\mathcal{S} = (\mathbf{Sign}, Sen, Mod, \vdash, \models, P, \mathrm{Pr}, \pi)$, *for* $P = proofs$ *and* $Pr = Id_{\mathbf{Set}}$, *and* π, γ, *proofs are defined as in Lemma 14.* □

Proposition 17. *The category* $\underline{Log}$ *admits generalization under* $(\_)^{\diamond}$ *and log.* □

Proposition 18. *The category* $\underline{Inst}$ *admits generalization under the functors* $(\_)^{\flat}$ *and inst*: $\underline{LogSys} \to \underline{Inst}$. □

7 Concluding Remarks

We have presented a general result about transportation of structure across comma categories and have shown how it yields a general technique for borrowing logical components from one logic for use in another logic when a map between some basic component of both exists. In addition, we have shown that the constructions are particularly well behaved in that they preserve completeness when it exists.

Given the generality of the categorical techniques used, they could in principle be used not only for the notions of mapping between logics and logical components that we have considered, but also for other axiomatic notions of mapping and logical component proposed in the literature by other authors. Exploring the applicability or our techniques to those other notions seems a worthwhile research direction.

Another topic that deserves more research is studying what limits and colimits exist in categories of logics or in categories of logical components. We have made use of some positive results in this paper and are aware of some negative results as well. A systematic study would be very useful.

Acknowledgements

We are indebted to Narciso Martí-Oliet for his careful reading of the manuscript and for his many very helpful suggestions to improve the exposition. We cordially thank José Luiz Fiadeiro for his valuable suggestions that have also helped us improve the paper. We finally thank the anonymous referees for their positive suggestions and criticism. The diagrams have been realized using P. Taylor's macros.

References

1. E. Astesiano and M. Cerioli. Partial higher-order specifications. *Fundamenta Informaticae*, 16(2):101–126, 1992.
2. E. Astesiano and M. Cerioli. Relationships between logical frames. In M. Bidoit and C. Choppy, editors, *Recent Trends in Data Type Specification*, number 655 in LNCS, pages 126–143, Berlin, 1993. Springer Verlag.
3. R.M. Burstall and J.A. Goguen. The semantics of Clear, a specification language. In D. Bjørner, editor, *Proceedings of the 1979 Copenhagen Winter School on Abstract Software Specification*, number 86 in LNCS, pages 292–332, Berlin, 1980. Springer Verlag.
4. M. Cerioli and J. Meseguer. May I borrow your logic? Technical report, SRI International, Computer Science Laboratory, 1993. To appear.
5. H. Ehrig, M. Baldamus, and F. Cornelius. Theory of algebraic module specification including behavioural semantics, constraints and aspects of generalized morphisms. In *Proceedings of 2nd International Conference on Algebraic Methodology and Software Technology*, pages 101–125, Iowa City, Iowa, USA, 1991.
6. H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specifications 1: Equations and Initial semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, Berlin, 1985.
7. J. Fiadeiro and A. Sernadas. Structuring theories on consequence. In D. Sannella and A. Tarlecki, editors, *Recent Trends in Data Type Specification*, number 332 in LNCS, pages 44–72, Berlin, 1987. Springer Verlag.
8. J.A. Goguen and R.M. Burstall. Introducing institutions. In E. Clarke and D. Kozen, editors, *Logics of Programming Workshop*, number 164 in LNCS, pages 221–255, Berlin, 1984. Springer Verlag.
9. J.A. Goguen and R.M. Burstall. A study in the foundations of programming methodology: Specifications, institutions, charters and parchments. In D. Pitt, S. Abramsky, A. Poigné, and D. Rydehard, editors, *Proceedings of Summer Workshop on Category Theory and Computer Programming*, number 240 in LNCS, pages 313–333, Berlin, 1986. Springer Verlag.
10. J.A. Goguen and R.M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146, 1992.

11. J.A. Goguen and J. Meseguer. Eqlog: Equality, types, and generic modules for logic programming. In D. DeGroot and G. Lindstrom, editors, *Logic Programming: Functions, Relations and Equations*, pages 295–363. Prentice-Hall, 1986.
12. J.A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud. Introducing OBJ. Technical Report SRI-CSL-92-03, SRI International, Computer Science Laboratory, 1992. To appear in J.A. Goguen, editor, *Applications of Algebraic Specification Using OBJ*, Cambridge University Press.
13. R. Harper, D. Sannella, and A. Tarlecki. Logic representation in LF. In D.H. Pitt et al., editor, *Category Theory and Computer Science*, number 389 in LNCS, pages 250–272. Springer Verlag, 1989.
14. S. MacLane. *Categories for the Working Mathematician*. Springer Verlag, 1971.
15. B. Mayoh. Galleries and institutions. Technical Report DAIMI PB - 191, Aarhus University, 1985.
16. J. Meseguer. General logics. In *Logic Colloquium '87*, pages 275–329, Amsterdam, 1989. North Holland.
17. J. Meseguer. A logical theory of concurrent objects and its realization in the Maude language. SRI Technical Report SRI-CSL-92-08, July 1992. To appear in G. Agha, P. Wegner, and A. Yonezawa, editors, *Research Directions in Object-Based Concurrency*, MIT Press, 1993.
18. P. Padawitz and M. Wirsing. Completeness of many-sorted equational logic revisited. *Bulletin EATCS*, (24), 1984.
19. A. Poigné. Foundations are rich institutions, but institutions are poor foundations. In H. Ehrig, H. Herrlich, Kreowski H.-J., and G. Preuß, editors, *Categorical Methods in Computer Science*, number 393 in LNCS, pages 82–101, Berlin, 1989. Springer Verlag.
20. A. Salibra and G. Scollo. A soft stairway to institutions. In *Recent Trends in Data Type Specification*, number 655 in LNCS, pages 310–329, Berlin, 1992. Springer Verlag.
21. D. Sannella and A. Tarlecki. On observational equivalence and specifications. *Journal of Comp. and Sys. Sciences*, 34:150–178, 1987.
22. A. Tarlecki. Free constructions in algebraic institutions. In M.P. Chytil and V. Koubek, editors, *Proceedings of Mathematical Foundation of Computer Science '84*, number 176 in LNCS, pages 526–534, Berlin, 1984. Springer Verlag.
23. A. Tarlecki. On the existence of free models in abstract algebraic institutions. *Theoretical Computer Science*, 37(3):269–304, 1985.

This article was processed using the LaTeX macro package with LLNCS style