Under consideration for publication in Math. Struct. in Comp. Science

Category-based Constraint Logic

Răzvan Diaconescu[†]

Japan Advanced Institute for Science and Technology

Received 6 September 1999

This research exploits the view of constraint programming as computation in a logical system, namely *constraint logic*. The basic ingredients of constraint logic are: *constraint models* for the semantics (they form a comma-category over a fixed model of "built-ins"), *generalized polynomials* in the rôle of basic syntactic ingredient, and a *constraint satisfaction* relation between semantics and syntax. *Category-based* constraint logic means the development of the logic is abstract categorical rather than concrete set theoretical.
We show that (category-based) constraint logic is an institution, and we internalize the study of constraint logic to the abstract framework of category-based equational logic, thus opening the door for considering constraint logic programming over non-standard structures (such as CPO's, topologies, graphs, categories, etc.). By embedding category-based constraint logic into category-based equational logic, we integrate the constraint logic programming paradigm into (category-based) equational logic programming. Results include completeness of constraint logic deduction, a novel Herbrand theorem for constraint logic programming characterizing Herbrand models as initial models in constraint logic, and logical foundations for modular combination of constraint solvers based on amalgamated sums of Herbrand models in the constraint logic institution.

1. Introduction

1.1. *Extensible Constraint Logic Programming*

Constraint logic programming has been recently emerging as a powerful programming paradigm and it has attracted much research interest over the past decade. Constraint logic programming merges two declarative programming paradigms: constraint solving and logic programming. Mathematical Programming, Symbolic Computation, Artificial Intelligence, Program Verification and Computational Geometry are examples of application areas for constraint solving. Constraint solving techniques have been incorporated in many programming systems; CLP (Jaffar and Lassez, 1987), PrologIII (Colmerauer,), and Mathematica are the best known examples. The computational domains include linear arithmetic, boolean algebra, lists, finite sets. Conventional logic programming (i.e., Prolog) can be regarded as constraint solving over term models (i.e., Herbrand universes). In this way, constraint logic programming can be regarded as a generalization of logic programming that replaces unification with constraint solving over computational

[†] On leave from the Institute of Mathematics of the Romanian Academy, PO Box 1-764, Bucharest 70700, ROMANIA.

domains. In general, the actual constraint logic programming systems allow constraint solving for a fixed collection of data types or computational domains.[1] Constraint logic programming allowing constraints over *any* data type (possibly given with loose semantics) will be called **extensible** (abbreviated **ECLP**).

This paper presents an (abstract) model theoretic semantics for ECLP, without directly addressing the computational aspect. This is a rather novel approach on the area of constraints where almost all efforts have been devoted to computational and operational issues; it is important the reader understands the model-theoretic and foundational orientation of this paper. However, we plan to gradually develop the computational side based on these foundations as further research (Section 7.2 sketches some of the directions of such further research). Some computational aspects of this theory can already be found in (Diaconescu, 1996c).

This semantics is

— logical,
— abstract, and
— institution-independent.

The first aspect means that there is an underlying logic in which all main features of ECLP can be rigorously explained. The second means that we develop the main concepts and results at the "highest appropriate level of abstraction", leaving out unnecessary details whilst still addressing the substance of ECLP. Finally, "institution-independent" addresses both former aspects within the theory of institutions (Goguen and Burstall, 1992), which represents now the modern level of algebraic specification. This means that our semantics to ECLP can be internalized to various institutions (i.e., logics), thus providing an uniform way for integrating ECLP into various systems with rigorous logical semantics, but also developing ECLP over novel structures.

The main results reported in this paper are:

— define a generic logic underlying ECLP (called **constraint logic**),
— embedding constraint logic into the category-based equational logic of (Diaconescu, 1994; Diaconescu, 1995; Goguen and Diaconescu, 1995; Diaconescu, 1996b),
— a generic Herbrand Theorem for constraint logics providing foundations for the concept of constraint solving in ECLP,
— a generic institution for ECLP providing foundations for modular ECLP and for connecting constraint logic to other computing logics via institution mappings (morphisms), and
— logical foundations for modular combination of constraint solvers via amalgamation of Herbrand models in constraint logic.

The embedding of constraint logics into category-based equational logic constitute the engine for most of the main results in this paper, but also a potential source for further developments. Due to this embedding, the constraint logic institution has properties close to algebraic specification institutions; also we are able to prove a Herbrand Theorem for ECLP by using the

[1] From a model-theoretic perspective, a computational domain may be abstracted to a model (not necessarily the standard one) of a certain data type specification. There can be several specifications which have a certain domain as their model, a typical example being the case of the real numbers which can be regarded as monoid in two different ways, as ring, as commutative ring, etc. Since in many cases it is not possible to find a finite specification which has the respective domain as its initial model, one has to consider the most appropriate specification of the domain with respect to the intended application.

corresponding result for category-based equational logic (see (Diaconescu, 1995; Diaconescu, 1994)). On the more practical side, this means the possibility to directly transfer software engineering and implementation techniques and methodologies developed for algebraic specification to ECLP. Examples include advanced modularization techniques (for modular combination of constraint solvers, for example) and the operational semantics based on *constraint paramodulation* of (Diaconescu, 1996c). Finally, since the development of ECLP can be internalized to any category-based equational logic, this means the possibility of systematically developing ECLP over non-set-theoretic structures (such as graphs or categories, for example). This is an area of great potential not yet explored, and is the subject of future research.

Our approach to ECLP (informal) As with the CLP approach of Jaffar and Lassez (Jaffar and Lassez, 1987), both constraint relations and programs are (sets of) sentences in the same logical system. But our constraint logics are much more general than Horn clause logic. Also, the computational domain plays a primary rôle in our definition of constraint logic, rather than being axiomatized in Horn clause logic, as in (Jaffar and Lassez, 1987).

When regarded as a model in constraint logic, the computational domain is an *initial* model. This is mathematically linked to the semantics of OBJ-like module systems, the fundamental idea being to regard the models of ECLP as expansions of an appropriate *built-in model A* along a signature inclusion $\iota\colon \Sigma \hookrightarrow \Sigma'$, where Σ is the signature of built-in sorts, operations and relations, and Σ' adds new "logical" symbols. In practice, the constraint relations (i.e., the logical relations one wishes to impose on potential solutions) are limited to atomic sentences involving both Σ-symbols and elements of the built-in model *A*. However, at the theory level there is no reason to restrict constraint relations to be atomic formulae. The models for ECLP are expansions of the built-in model to the larger signature Σ', and morphisms of constraint models must preserve the built-ins. Thus the constraint models form a comma category, $A/\mathrm{MOD}(\iota)$.

Example 1.1. (The Euclidean plane) Consider the example of a specification of the Euclidean plane as a vector space over the real numbers.

```
obj R2 is
  pr FLOAT * (sort Float to Real) .
  sort Vect .

  op 0 : -> Vect .
  op <_,_> : Real Real -> Vect .
  op _+_ : Vect Vect -> Vect .
  op -_ : Vect -> Vect .
  op _*_ : Real Vect -> Vect .

  vars a b a' b' k : Real .
  eq 0 = < 0 , 0 > .
  eq < a , b > + < a' , b' > = < a + a' , b + b' > .
  eq k * < a , b > = < k * a , k * b > .
  eq - < a , b > = < - a , - b > .
endo
```

The signature Σ of built-in sorts, operation and relation symbols contains one sort `Real`[2] for the real numbers together with the usual ring operation symbols and a relation symbol `_<_`. The built-in model is just the usual ring of real numbers (denoted as $\mathbb{R}$) with `_<_` interpreted as the usual 'strictly less than' predicate. The signature Σ' of the module `R2` introduces a new operation symbol `<_,_>` for representing the points of the Euclidean plane as tuples of real numbers, and overloads the ring operations by organizing the Euclidean plane as a vector space over the real numbers. The axioms express the basic fact that the evaluation of the ring operations on vectors is done component-wise.
A standard model for this specification, denoted $\mathbb{R}^2$, is given by the Cartesian representation of the points of the Euclidean plane, i.e., any point is represented as the tuple of its coordinates. Another model for (the corresponding 'theory" version, allowing loose models, of) `R2` interprets the sort `Vect` as the set of real numbers, the $\mathbb{R}$-module[3] operations on `Vect` as ordinary operations on numbers, but `<_,_>` as addition of numbers. We denote this model by $\mathbb{R}+$.

Example 1.2. (Equational logic with a built-in Boolean type) Very oftenly modern algebraic specification systems provide some pre-defined data types, such as the Booleans. For example, in both OBJ (Goguen et al.,) and CafeOBJ (Diaconescu and Futatsugi, 1998) each module imports the data type of the Booleans by default. This has multiple consequences, for example, it supports a more general form of conditional equations, where conditions are Boolean-sorted terms rather than just finite conjunctions of identities (see (Goguen et al., ; Diaconescu and Futatsugi, 1998)). This can be regarded as a special case of constraint logic by letting the signature Σ of built-ins to consist of just one sort `Bool` and the model A of built-institution consisting of just two elements `true` and `false`. The other operations on `Bool` (such as `and`, `or`) can be considered as part of Σ'; however more sophisticated versions of this example treating the other `Bool`-operations as part of the built-in signature would also work well.
This example can also be used for embedding **logic programming with negation** into ECLP by interpreting the predicates (i.e, relational symbols) of a logic program as `Bool`-valued functions. A positive literal will be written as `p(x) = true` rather than `p(x)`, and a negative literal as `p(x) = false` rather than $\neg$`p(x)`. The models are restricted to the *full subcategory* $\mathbb{B}$ of $A/\text{MOD}(\iota)$ with objects isomorphisms $A \sim B\restriction_\iota$; this means exactly the protection of the `Bool`-values `true` and `false`. See also the discussion on *conservative* models in Section 5.2.
Consider the example of a *modus tollens* program:

```
obj MODUS-TOLLENS is
  sort s .
  ops p q : s -> Bool .   *** unary predicates on s
  op a : -> s .           *** constant of s
  var X : s .
  cq q(X) = true if p(X) .
  eq q(a) = false .
jbo
```

[2] Obtained here by renaming the sort `Float` of the imported built-in OBJ module `FLOAT` implementing the real numbers as floating point reals.
[3] In the sense of linear algebra.

The initial (in $\mathbb{B}$) model H for this program interprets $p_H(a)$ as `false`. (Notice that this happens exactly because all models in $\mathbb{B}$ must protect the Booleans. The classical proof-theoretic Herbrand model for this program would introduce `p(a)` as a new value of sort `Bool`.)
As will be explained later in the paper, the initial model plays the role of the (constraint) Herbrand model, so a query such as `p(Y) = false` must get the answer `Y = a`.[4]

1.2. *Constraint Logic, Constraint Institutions, and Category-based Equational Logic*

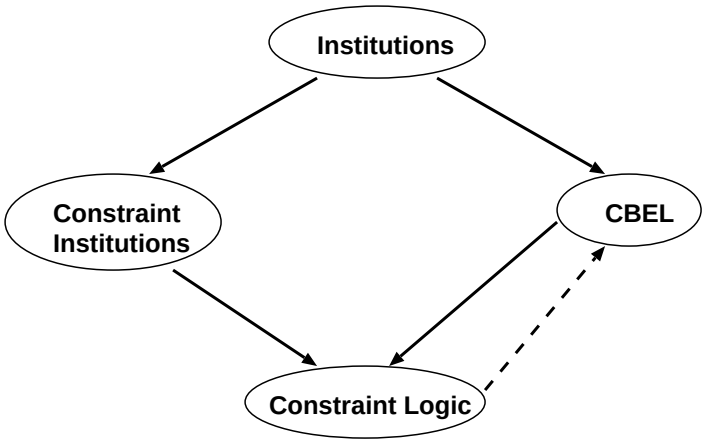
The main concept proposed by this research is that of *constraint logic*. The development of constraint logic and of the main results involves intimately two other frameworks: *category-based equational logic* and *constraint institutions*.

Category-based equational logic (Diaconescu, 1994; Diaconescu, 1995; Diaconescu, 1996b; Goguen and Diaconescu, 1995; Diaconescu, 1996c) (abbreviated **CBEL**) abstracts out the essential ingredients of equational logic. Equations, deduction, models (algebras), congruences, satisfaction, etc. are treated in an arbitrary category of models satisfying certain mild conditions, including a forgetful functor to a category of *domains*. This encodes the principle that a model interprets a signature (often called a "vocabulary" in classical logic) into a domain, usually (but not necessarily) a set, or for typed systems, a collection of sets. Category-based equational logic programming (abbreviated **CB** and **ELP**, respectively) extends this generalization to computation, including rewriting, paramodulation (Diaconescu, 1994; Diaconescu, 1996c), modules (Diaconescu, 1994; Diaconescu, 1996b), and constraint solving ((Diaconescu, 1994; Diaconescu, 1996a) and this paper). CBEL also involves another level of generality by considering equalities between elements of arbitrary models as sentences. This generalizes classical equations by viewing the terms as elements of some free model (i.e., term model). Results include completeness of deduction, a Herbrand theorem, completeness of paramodulation, and generic modularization techniques.

In Section 7 we introduce **constraint institutions** as a special class of institutions internalizing the model part of constraint logics to any institution and leaving the sentence part and the satisfaction between models and sentences abstract. This not only provides a conceptual separation between the model theory and the syntax of constraint logic (which is very beneficial for the economical development of the semantics of modular combination of constraint solvers), but also provides a formalization for the model theory of logics over pre-defined structures with possibly very different concept of satisfaction between models and sentences. A meaningful example is given by the hidden sorted logics (Goguen and Diaconescu, 1994b; Goguen and Malcolm, 1997) used in behavioural specification.

The following is a diagram illustrating a hierarchical relationship between the various concepts and frameworks used in this paper.

[4] Notice that this is a semantics argument, the operational issues related to this example are also interesting and they constitute an important research topic.



The most abstract concept is that of institution. Constraint institutions and CBEL are both institutions. Constraint logic is a constraint institution but can also be embedded into CBEL (in this way ECLP can be regarded as a special case of [an abstract form of] plain equational logic programming). Finally, the dotted arrow shows that constraint logic can be developed abstractly on top of CBEL.

1.3. *Structure of this paper*

This paper is structured as follows. After the Introduction and a section on Preliminaries, we devote a section to Institutions, where we briefly review the basic institution concepts and introduce several new concepts and prove some basic results which are necessary for the development of category-based constraint logic. The next section briefly surveys the main concepts and results of CBEL; the material of this section can be found also in (Diaconescu, 1995; Diaconescu, 1996b; Goguen and Diaconescu, 1995; Diaconescu, 1994; Diaconescu, 1996c). This section also introduces the so-called **Simplifying Assumption** on CBEL which simplifies the presentation of this research but without restricting the real generality of our approach (i.e., everything here can be developed in the absence of the **Simplifying Assumption**). Section 5 develops category-based constraint logic and introduces the core novel concepts of this research. The main result here is the embedding of constraint logic into CBEL. Section 6 is devoted to the main result of this paper, the Herbrand Theorem for category-based constraint logic which is obtained via the embedding of constraint logic into CBEL and by instantiating the Herbrand Theorem for CBEL to constraint logic. This instantiation requires an interesting categorical proof. The final section introduces the more general concept of constraint institution, shows how constraint logic fits this generalization, and at the end sketches a categorical semantics for modular combination of constraint solvers by using both results for constraint institutions and the Herbrand Theorem for category-based constraint logic.

In this paper we formulate previously published results without giving their proof and prove only the new results or results which have not been proved before in published form.

2. Preliminaries

Categories. This work assumes familiarity with the basics of universal algebra and category theory, and generally uses the same notation as Mac Lane (Lane, 1971), except that composition is denoted by ";" and written in the diagrammatic order. The application of functions (functors) to arguments may be written either normally using parentheses, or else in diagrammatic order without parentheses. Categories usually have a name with first letter in capital bbold font; for example the category of sets is $\mathbb{S}et$, and the category of categories is $\mathbb{C}at$. The opposite of a category $\mathbb{C}$ is denoted by $\mathbb{C}^{op}$. Functors are usually (but not always!) denoted by caligraphic capital letters, particularly for 'functor variables' as opposed to functors whose action is known. The class of objects of a category $\mathbb{C}$ is denoted by $|\mathbb{C}|$; also the set of arrows in $\mathbb{C}$ having the object a as source and the object b as target is denoted by $\mathbb{C}(a,b)$.

Given two functors $\mathbb{C} \xrightarrow{\mathcal{C}} \mathbb{E} \xleftarrow{\mathcal{D}} \mathbb{D}$, the **comma category** $(\mathcal{C}/\mathcal{D})$ has arrows $c\mathcal{C} \xrightarrow{t} d\mathcal{D}$ as objects and pairs of arrows $\langle f, g\rangle$ as morphisms, such that $t;g\mathcal{D} = f\mathcal{C};t'$. For functors collapsing everything to a constant object (i.e., to an identity arrow) we use the object itself as notation.

We denote **coproducts** by $+$, **coequalisers** by *coeq*, and **kernels** by *ker* (i.e., a pullback of an arrow with itself). An object A in a category $\mathbb{C}$ is **coequaliser projective** (or just *projective* for short) iff for any coequaliser $B \xrightarrow{c} C$ and any arrow $A \xrightarrow{h} C$ there exists an arrow h' such that $h';c = h$. Projectivity is very oftenly a more abstract alternative of freeness that does not require an adjunction.[5]

A functor $\mathcal{U}\colon \mathbb{A} \to \mathbb{X}$ has a **left-adjoint** $\mathcal{F}\colon \mathbb{X} \to \mathbb{A}$ iff for each $X \in |\mathbb{X}|$ there is an arrow $X\eta\colon X \to X\mathcal{F}\mathcal{U}$ such that for each $f \in \mathbb{X}(X, A\mathcal{U})$ there exists an unique arrow $f^\sharp \in \mathbb{A}(X\mathcal{F}, A)$ such that $X\eta; f = f^\sharp\mathcal{U}$. Then $\mathcal{U}$ is called a **right-adjoint**.

An **indexed category** (Tarlecki et al., 1991) is a functor $\mathbb{C}\colon I^{op} \to \mathbb{C}at$; sometimes $\mathbb{C}$ is also denoted as $\{\mathbb{C}_i\}_{i\in I}$. The following flattening construction[6] plays an important rôle in this paper. Given an indexed category $\mathbb{C}\colon I^{op} \to \mathbb{C}at$, let $Flat(\mathbb{C})$ be the category having (i,a), with $i \in |I|$ and $a \in |\mathbb{C}_i|$, as objects and $(u,f)\colon (i,a) \to (j,b)$, with $u \in I(i,j)$ and $f\colon a \to b\mathbb{C}_u$, as arrows, where the composition of arrows is defined by $(u,f);;(u',f') = (u;u', f;f'\mathbb{C}_u)$.

This work needs the following categorical treatment of binary relations (Diaconescu, 1994; Diaconescu, 1995):

Definition 2.1. Let A be an object of a category $\mathbb{X}$. Then a **binary relation representation on** A is a parallel pair of arrows $s,t \in \mathbb{X}(I,A)$, denoted $I \xrightarrow{\langle s,t\rangle} A$ or just $\langle s,t\rangle$. Let $I \xrightarrow{\langle s,t\rangle} A$ and $I' \xrightarrow{\langle s',t'\rangle} A$ be binary relation representations on the same object A. Then $\langle s,t\rangle$ is **included in** $\langle s',t'\rangle$ (denoted $\langle s,t\rangle \subseteq_A \langle s',t'\rangle$, or just $\langle s,t\rangle \subseteq \langle s',t'\rangle$) iff there is a map $h\colon I \to I'$ between the objects of indices such that $s = h;s'$ and $t = h;t'$. Two relation representations Q and Q' on the same object A are **equivalent** (denoted $Q \equiv_A Q'$, or just $Q \equiv Q'$) iff $Q \subseteq Q'$ and $Q' \subseteq Q$. Then a **binary relation** on A is an equivalence class of $\equiv_A$.

Although binary relations are classes of equivalent representations, for simplicity we often use representations instead of classes. The concept of inclusion between binary relation representa-

[5] However there are situations when projective objects are not necessarily free.
[6] Also known under the name of Grothendick construction.

tions extends to binary relations proper without difficulty. As a matter of notation, by sQt we mean $\langle s,t\rangle \subseteq Q$.

3. Institutions

The theory of institutions (Goguen and Burstall, 1992) has recently emerged as one of the important areas in theoretical computer science, with many applications to modern algebraic specification, declarative and logical programming, programming in the large, etc. Basic concepts and results on institutions can be found in (Goguen and Burstall, 1992). In this section we very briefly review the main concepts and then develop some results concerning model amalgamation in institutions; these results are needed later for the semantics of combining constraint solvers and domains.

We denote an institution $\Im$ by $(\mathbb{S}ign, \text{MOD}, Sen, \models)$, where $\mathbb{S}ign$ is the category of signatures, $\text{MOD}\colon \mathbb{S}ign \to \mathbb{C}at^{op}$ the model functor, $Sen\colon \mathbb{S}ign \to \mathbb{C}at$ the sentence functor, and $\models$ is the satisfaction relation. A **theory** (Σ, E) consists of a signature Σ and a closed (under $\models_\Sigma$) set of Σ-sentences E. A **theory morphism** $(\Sigma, E) \to (\Sigma', E')$ is just a signature morphism $\Sigma \to \Sigma'$ mapping E to a subset of E'. Let $\mathbb{T}h(\Im)$ denote the **category of all theories** in $\Im$. A theory morphism $\varphi\colon (\Sigma, E) \to (\Sigma', E')$ is **liberal** iff the reduct functor $\text{MOD}(\varphi)\colon \text{MOD}(\Sigma, E) \to \text{MOD}(\Sigma', E')$ (often denoted as $\_\restriction_\varphi$) has a left adjoint, and it is **persistent** iff it is liberal and $\text{MOD}(\varphi)$ has a left inverse. An institution $\Im$ is **liberal** iff every theory morphism in $\mathbb{T}h(\Im)$ is liberal. An institution is **exact** iff the model functor MOD preserves all finite co-limits, and **semi-exact** iff it preserves only pushouts. Liberality and exactness are very important desirable properties for institutions, especially in connection to modularization (Diaconescu et al., 1993).

3.1. *Model Amalgamation*

In this section we are concerned with the technicalities of "putting together" models of different signatures. We develop the related concepts and results within the general framework of institutions.

Let $\Im = (\mathbb{S}ign, \text{MOD}, Sen, \models)$ be an institution. Given a model $A \in |\text{MOD}(\Sigma)|$, we denote its signature Σ by $\underline{A}$.

Definition 3.1. Let A and A' be models, not necessarily in the same signature. A **generalized model morphism** from $h\colon A \to A'$ consists of a signature morphism $\underline{h}\colon \underline{A} \to \underline{A'}$ and an ordinary model morphism $h\colon A \to A'\restriction_{\underline{A}}$. Notice that for simplicity we may use the same name for the ordinary morphism component of a generalized morphism; but we always make a notational distinction at the level of composition of such morphisms (denoted by ;; in the case of generalized morphisms and by ; in the case of ordinary morphisms).

Remark 3.1. The category of models and generalized model morphisms is exactly $Flat(\text{MOD})$.

The following definition generalizes Baader and Schultz universal algebra concept of "free amalgamated product" (Baader and Schultz, 1994; Baader and Schulz, 1994).

Definition 3.2. Let A_1 and A_2 be models, not necessarily in the same signature. A **base** for A_1 and A_2 consists of a model A_0 together with generalized model morphisms $h_i\colon A_0 \to A_i$, $i = 1,2$. Given a base for A_1 and A_2, the **amalgamated sum** $A_1 \oplus_{A_0} A_2$ is the pushout of this base.

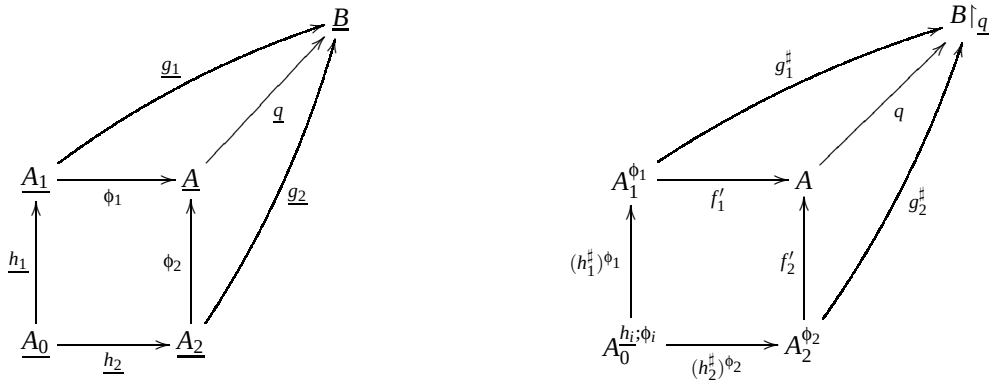
Notice that the pushout mentioned in this definition is considered in $\mathit{Flat}(\text{MOD})$.

The following result establishes the existence of amalgamated sums of models for institutions.

Theorem 1. If $\mathbb{S}\mathit{ign}$ has pushouts, the institution $\Im$ is semi-exact, all signature morphisms in $\Im$ are liberal,[7] and the categories of models for each signature have pushouts, then the amalgamated sum of any two models exists.

Proof. One "high-level" way to prove this result is to make use of the basic theorem (Theorem 2 of (Tarlecki et al., 1991)) on colimits in the flattened of an indexed category. Then conclusion follows directly from the hypotheses.

In the following we give a direct proof of this result. Let $A_1 \xleftarrow{h_1} A_0 \xrightarrow{h_2} A_2$ be a base. The signature $\underline{A}$ of the model A standing for the vertex of the pushout of (h_1, h_2) is defined by the pushout of $(\underline{h_1}, \underline{h_2})$ in $\mathbb{S}\mathit{ign}$, as shown in the left part of the following diagram:



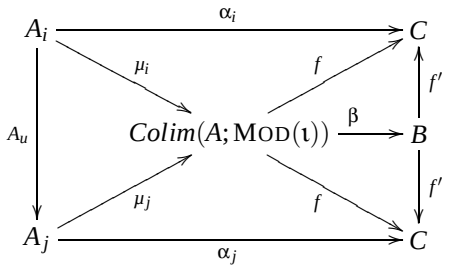
Let $h_i^\sharp \colon A_0^{\underline{h_i}} \to A_i$ be the unique $\underline{A_i}$-morphism "extending" $h_i \in \text{MOD}(\underline{A_0})(A_0, A_i\restriction_{\underline{h_i}})$. Define $f_i \colon A_i \to A$ by $\underline{f_i} = \phi_i$ and $f_i = \eta_i; f_i'\restriction_{\phi_i}$, where f_i' is defined by the pushout of $((h_1^\sharp)^{\phi_1}, (h_2^\sharp)^{\phi_2})$ in the category of $\underline{A}$-models (see the right hand side part of the above diagram), and $\eta_i \colon A_i \to A_i^{\phi_i}\restriction_{\phi_i}$. Straightforward calculations show that $h_1 ;; f_1 = h_2 ;; f_2$. Let $g_i \colon A_i \to B$ such that $h_1 ;; g_1 = h_2 ;; g_2$. We have to show that there exists a unique $q \colon A \to B$ such that $f_i ;; q = g_i$. Notice that $\underline{q}$ should be the unique signature morphism such that $\underline{f_i}; \underline{q} = \underline{g_i}$. $q \colon A \to B\restriction_{\underline{q}}$ is the unique $\underline{A}$-morphism such that $f_i'; q = g_i^\sharp$, where $g_i^\sharp \colon A_i^{\phi_i} \to B\restriction_{\underline{q}}$ is the unique "extension" of $g_i \colon A_i \to B\restriction_{\underline{g_i}} = (B\restriction_{\underline{q}})\restriction_{\phi_i}$ to an $\underline{A}$-morphism. □

We turn now to extending the existence of amalgamated sum of models to the case of theory models. Given an institution $\Im$, we can regard the theories as primitive entities by building a semantic institution with empty sentence functor, $\Im^{Th}$. So, for every institution $\Im$, let $\Im^{Th}$ be the institution $(\mathbb{T}h(\Im), \text{MOD}, \emptyset, \emptyset)$, where, for each theory T, $\text{MOD}(T)$ is the full subcategory of $\text{MOD}(\Sigma(T))$ satisfying the theory T.

Lemma 3.1. If the institution $\Im$ is liberal and the category of models of each signature has J-colimits, then $\text{MOD}(T)$ has J-colimits for each theory T.

[7] When regarded as theory morphisms between the corresponding empty theories.

Proof. Let $A\colon J \to \mathrm{MOD}(T)$ be a J-diagram of T-models. Let μ be the colimit of this diagram in $\mathrm{MOD}(\Sigma)$, where Σ is the signature of T (also let ι denote the inclusion $\Sigma \to T$).



Let β be the "quotienting" Σ-morphism constructing the free T-model over the vertex of the colimit of μ. Then we claim that $\mu;\beta$ is the colimit of A.
Consider a co-cone $\alpha\colon A \to C$ in $\mathrm{MOD}(T)$. This a co-cone in $\mathrm{MOD}(\Sigma)$ too, so there exists a unique
$f\colon Colim(A;\mathrm{MOD}(\iota)) \to C$ such that $\mu; f = \alpha$. Therefore there exists a unique $f'\colon B \to C$ such that $\beta; f' = f$. Thus, $(\mu;\beta); f' = \alpha$. The uniqueness of f' results from the uniqueness of f. □

Corollary 3.1. If the institution $\mathfrak{I}$ is semi-exact and liberal, $\mathbb{S}ign$ has pushouts, and the category of models for each signature has pushouts, then the amalgamated sum of any two theory models (i.e., the amalgamated sum of any models of $\mathfrak{I}^{Th}$) exists.

Proof. Liberality of $\mathfrak{I}$ means the liberality of $\mathfrak{I}^{Th}$ on signature morphisms. Fundamental results of institution theory (see (Goguen and Burstall, 1992; Diaconescu et al., 1993)) show that in any institution colimits and exactness carry from signatures to theories. The existence of pushouts of models in this case follows by previous lemma. □

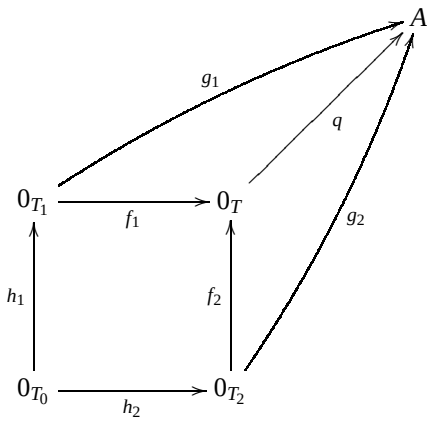
This result together with the following one can be regarded as generalizations of results on existence of "free amalgamated products" of free models in universal algebra varieties (Baader and Schulz, 1994; Baader and Schultz, 1994). Also notice that Proposition 3.1 uses a weaker condition of liberality than Corollary 3.1.

Proposition 3.1. Let $\mathfrak{I}$ be an institution with pushouts for signatures and whose theories admit initial models. Then the amalgamated sum of initial models in $\mathfrak{I}^{Th}$ exists and is initial too.

Proof. Firstly, notice that given any theory morphism $\phi\colon T \to T'$, there exists exactly one generalized model morphism $h\colon 0_T \to 0_{T'}$ with $\underline{h} = \phi$ which is defined as the unique morphism $0_T \to 0_{T'}|_\phi$ in $\mathrm{MOD}(T)$.
Now, consider the theory morphisms $T_1 \xleftarrow{\phi_1} T_0 \xrightarrow{\phi_2} T_2$ and let $T_1 \xrightarrow{\theta_1} T \xleftarrow{\theta_2} T_2$ be their pushout. In the virtue of the remark above, this pushout square generates a commutative square of initial models for the corresponding theories. We have to prove that

$$0_T = 0_{T_1} \oplus_{T_0} 0_{T_2}$$

This is the same with proving that the commutative square of initial models is a pushout square in the category of generalized morphisms of theory models.



Assume two generalized model morphisms $0_{T_1} \xrightarrow{g_1} A \xleftarrow{g_2} 0_{T_2}$ such that $h_1 ; g_1 = h_2 ; g_2$. In the virtue of the remark at the beginning of this proof, there exists a unique generalized model morphism $q\colon 0_T \to A$, and moreover, $f_i ; q = g_i$ for $i = 1, 2$. □

4. Category-based Equational Logic

This section surveys the basic concepts and results in CBEL that are necessary for this paper. A survey of CBEL is (Goguen and Diaconescu, 1995), and the full development of this theory can be found in (Diaconescu, 1994). Other relevant papers are (Diaconescu, 1995; Diaconescu, 1996c; Diaconescu, 1996b; Diaconescu, 1996a).

4.1. *Models and Domains*

The semantics of a logical system is given by its *models*. In general, soundness of the inference rules of a logical system is checked against its models using a satisfaction relation, in the style of Tarski (Tarski, 1944). We assume that models and their morphisms form a category. As in institutions (Goguen and Burstall, 1992), CBELs are "localized" to signatures. A model is an interpretation of a particular signature into a *domain*. Thus any model has an underlying domain, and this correspondence is functorial. Moreover, any two parallel model morphisms identical as maps between their domains should be the same. These assumptions are summed up in the following:

[Basic Framework]: There is an (abstract) category of *models* $\mathbb{A}$ and a *forgetful functor* $\mathcal{U}\colon \mathbb{A} \to \mathbb{X}$ to a category of *domains* $\mathbb{X}$ that is faithful and preserves pullbacks.

The simplicity of these assumptions reflects the simplicity of equational logic. The condition that $\mathcal{U}$ preserves pullbacks relates to congruences being equivalences (see (Diaconescu, 1994; Diaconescu, 1995)).

In practice, the forgetful functor $\mathcal{U}$ always has a left adjoint $\mathcal{F}$, which means that for every $X \in |\mathbb{X}|$, thought as a domain of variables, there is a *free* model $X\mathcal{F}$. Note that $\mathcal{U}$ preserves pullbacks when it has a left adjoint (e.g., see (Lane, 1971)).

The signatures of computing science logics usually involve a set of sorts. Then the categories of domains are categories of many sorted sets, i.e., $\mathbb{X} = \mathbb{S}et^S$ for some set S of sorts. Although CBEL was originally developed more abstractly (Diaconescu, 1995; Diaconescu, 1994), here we sometimes simplify by assuming that domains are many sorted sets, i.e., that the following

[Simplifying Assumption]: $\mathbb{X} = \mathbb{S}et^S$

holds. This avoids the difficult technical details of finiteness properties of categorical relations found in (Diaconescu, 1994) and (Diaconescu, 1995).

4.2. *Examples*

This subsection briefly sketches several examples, assuming familiarity with their basic concepts, and showing how they fall under the **Basic Framework**. A more detailed presentation of some of these examples can be found in (Diaconescu, 1994; Goguen and Diaconescu, 1995; Diaconescu, 1995; Diaconescu, 1996c; Diaconescu, 1996b).

Many Sorted Algebra. Given a many sorted signature (S, Σ), let $\mathbb{A}lg_\Sigma$ denote the category of Σ-algebras with Σ-homomorphisms. There is a forgetful functor $\mathcal{U}_\Sigma \colon \mathbb{A}lg_\Sigma \to \mathbb{S}et^S$ from Σ-algebras to S-sorted sets, forgetting the interpretations of the operation symbols in Σ. This functor has a left adjoint. Given a set X of variable symbols, let $T_\Sigma(X)$ denote the (S-sorted) **term algebra** with operation symbols from Σ and variable symbols from X.

Order Sorted Algebra. Order sorted algebra (abbreviated **OSA**) adds to MSA a partial ordering on sorts, which is interpreted as inclusion among the corresponding carriers; all approaches to OSA share this essential idea. See (Goguen and Diaconescu, 1994a) for a recent survey, including all basic OSA definitions (signature, algebra, homomorphism, regularity, etc.).

Given an order sorted signature $(S, \leq, \Sigma)$, the Σ-algebras and their homomorphisms form a category $\mathbb{A}lg_\Sigma$ of models for OSA. The forgetful functor $\mathcal{U}_\Sigma \colon \mathbb{A}lg_\Sigma \to \mathbb{S}et^S$ forgets both the algebraic *and* the subsorting structure. We emphasize that the domains for OSA should *not* have a subsorting structure, as is supported by the way OSA is implemented. Other approaches to OSA mentioned in (Goguen and Diaconescu, 1994a) can be treated similarly.

Rewriting Logic. Algebraic signatures can be interpreted into non-conventional structures that are more complex than the ordinary plain sets. Meseguer's *rewriting logic* (Meseguer, 1992) (abbreviated **RWL**) provides an interesting and important example, since RWL can be very effectively used as a unifying semantic framework for concurrency (see (Meseguer, 1992)).

Consider an algebraic signature (S, Σ). A **Σ-system** interprets each sort $s \in S$ as a category G_s and each operation $\sigma \in \Sigma_{s1 \ldots sn, s}$ as a functor $\sigma_G \colon G_{s1} \times \ldots \times G_{sn} \to G_s$. Σ-systems gives a nice formalization for *distributed concurrent systems*, the arrows between the elements of the carriers of a Σ-system encoding the transitions in the local states of the system. Σ-systems and their *morphisms* (defined as S-indexed functors commuting with the operations of Σ) form a category $\mathbb{S}ys_\Sigma$. The category of domains is taken to be $\mathbb{S}et^S$, and the forgetful functor $\mathcal{U}_\Sigma \colon \mathbb{S}ys_\Sigma \to \mathbb{S}et^S$ forgets the interpretations of the algebraic operations and the arrow composition, i.e., mapping each category to its set of arrows.

Horn Clause Logic. In (Diaconescu, 1990) we introduce an embedding of the category of models $\mathbb{M}od_{\Sigma,\Pi}$ of a first order signature (S,Σ,Π) as a retract of the category of algebras of an MSA signature $(S^b,\Sigma^b\cup\Pi^b)$ obtained from the original first order signature by turning predicates into operations. Interpreting predicates as boolean valued operations is hardly new; it has even been used to lift narrowing to an operational semantics for logic programming (Dershowitz, 1983). However, this approach (further exploited in (Diaconescu, 1995; Diaconescu, 1996c; Diaconescu, 1996b; Diaconescu, 1994)) is somewhat different, because it does not assume a full boolean structure on the new sort of truth values. Moreover, the model theoretic aspect is emphasized.

A consequence of this result is that given a first order signature (S,Σ,Π), the category of models for Horn clause logic (abbreviated **HCL**) can be taken as $\mathbb{A}lg_{\Sigma^b\cup\Pi^b}$ instead of $\mathbb{M}od_{\Sigma,\Pi}$, and its sentences as conditional equations instead of Horn clauses. Notice that in HCL, unlike MSA, the forgetful functor from models to domains, $\mathbb{A}lg_{\Sigma^b\cup\Pi^b}\to\mathbb{S}et^S$, is *not* monadic.

Equational Logic Modulo Axioms. Equational deduction modulo a set of axioms (abbreviated **ELM**) is needed for rewriting when there are non-orientable equations; detailed expositions are given in many surveys or textbooks; we mention (Goguen, 2000). Although in practice non-orientable rules are mostly unconditional, there is no theoretical reason to exclude equational deduction modulo a set of conditional equations. Idempotence is a non-orientable conditional axiom, when given in the form `x + y = x if x = y`. Equational deduction modulo E generalizes the usual concepts of MSA to "concepts modulo E", including the inference rules (Goguen, 2000). A model theory for equational logic modulo E requires an adequate notion of model, and it is natural to use $\mathbb{A}lg_{\Sigma,E}$, which gives "algebras modulo axioms" (i.e., all Σ-algebras satisfying each axiom in E). The category of domains is the category $\mathbb{S}et^S$ of S-sorted sets and functions, and the forgetful functor $\mathcal{U}_{\Sigma,E}:\mathbb{A}lg_{\Sigma,E}\to\mathbb{S}et^S$ forgets both the axioms and the MSA structure.

ELM includes the example of Mosses's unified algebras (Mosses, 1989) whose logic can be regarded as equational logic modulo a conditional theory.

Summary of Examples. The following summarizes the examples discussed above:

	$\mathbb{A}$ (category of models)	$\mathcal{U}$ forgets:
MSA	$\mathbb{A}lg_{\Sigma}$	algebraic structure
OSA	$\mathbb{A}lg_{\Sigma}$	algebraic structure + subsorting
RWL	$\mathbb{S}ys_{\Sigma}$	algebraic structure + arrow composition
HCL	$\mathbb{A}lg_{\Sigma^b\cup\Pi^b}$	algebraic structure + sort b
ELM	$\mathbb{A}lg_{\Sigma,E}$	algebraic structure + axioms

Any combination of these logical systems is possible, e.g., order sorted Horn clause logic with equality which is the logic underlying Eqlog.

4.3. *Category-based Equational Deduction*

Equations are traditionally pairs of terms constructed from the symbols of a signature plus some variables. Goguen and Meseguer (Goguen and Meseguer, 1985) first made quantifiers part of the

concept of equation, for MSA. Although terms are syntactic constructs, from a model theoretic perspective they are just elements of the free term model over the set of quantified variables. Any valuation of the variables into a model extends uniquely to a model morphism evaluating both sides of the equation. Thus, a more semantic treatment of quantification regards quantifiers as models rather than sets of variables, and regards valuations as model morphisms rather than functions; this was already done in (Căzănescu, 1993) for MSA. This non-trivial generalization of equation and satisfaction extends naturally to equational deduction, and in our opinion gives a pleasing unity and generality to the whole area.

Definition 4.1. Let A be any model. Then a $\mathcal{U}$-**identity** on A is a binary relation $k \xrightarrow{\langle s,t\rangle} A\mathcal{U}$ on the underlying domain of A. An identity $\langle s,t\rangle$ in A is **satisfied** in a model B with respect to a model morphism $h\colon A \to B$ iff $s;h\mathcal{U} = t;h\mathcal{U}$. This is denoted $B \models \langle s,t\rangle[h]$.
A $\mathcal{U}$-**equation** is a universally quantified expression $(\forall A)\langle s,t\rangle$ where A is a model representing the quantifier and $\langle s,t\rangle$ is an identity in A. A model B **satisfies** $(\forall A)\langle s,t\rangle$ iff B satisfies the identity $\langle s,t\rangle$ for all model morphisms $h\colon A \to B$. This is written $B \models (\forall A)\langle s,t\rangle$. A **conditional** $\mathcal{U}$-**equation** is an expression having the form $(\forall A)\ \langle s',t'\rangle$ **if** $\langle s,t\rangle$ where A is a model representing the quantifier, $\langle s',t'\rangle$ is a $\mathcal{U}$-identity on A, and $\langle s,t\rangle$ is a finite (i.e., the "index" object [i.e., the source of s and t] of the relation is finite[8] (Diaconescu, 1994) or (Diaconescu, 1995)) binary relation on the domain of A representing the hypotheses (i.e, the condition) of the equation. A model B **satisfies** $(\forall A)\langle s',t'\rangle$ **if** $\langle s,t\rangle$ iff for any morphism $h\colon A \to B$, $s;h\mathcal{U} = t;h\mathcal{U}$ implies $s';h\mathcal{U} = t';h\mathcal{U}$.
A $\mathcal{U}$-**query** is an existentially quantified expression $(\exists A)\langle s,t\rangle$ where A is a model representing the quantifier and $\langle s,t\rangle$ is an identity in A. A **solution** of $(\exists A)\langle s,t\rangle$ in a model B is any model morphism $h\colon A \to B$ for which $\langle s,t\rangle$ is satisfied in B with respect to h.

Notice that the notion of $\mathcal{U}$-equation (query) deals with *families of equations (queries)*, rather than single equations (queries), as sentences.

Completeness of CB equational deduction is traditional in that the central concept is the congruence determined by a set Γ of (possibly conditional) equations on a model A (e.g., see (Birkhoff, 1935)). The most abstract completeness result states the equivalence of two versions of this congruence: all unconditional equations quantified by A that can be *syntactically inferred* from Γ; and all unconditional equations quantified by A that are *semantic consequences* of Γ. This semantic treatment of equation and satisfaction (see (Diaconescu, 1994; Diaconescu, 1995)) allows the congruences determined by Γ on free models and on other models to be treated the same way. Despite the generality and abstraction, the rules of inference for CB equational deduction can be made explicit for concrete examples, and can be recognized even in the most abstract formulation.

The following technical assumption underlies the proof theory for CBEL:

[Deduction Framework]: Basic Framework + the category $\mathbb{A}$ of models has pullbacks and coequalisers.

[8] It has been observed that in many important examples, the intuitive notion of an object A in a category $\mathbb{C}$ being "finite" coincides with the technical condition of the set-valued "hom" functor $\mathbb{C}(A,-)$ preserving filtered colimits. Therefore we adopt here this technical notion of finiteness. The book (Adamek and J.Rossicki, 1994) contains a good study of this categorical notion of finiteness.

Let A be an arbitrary model. Then a binary relation Q on the domain of A is a **congruence** iff it is a kernel of a model morphism, i.e., iff there is a morphism φ in $\mathbb{A}$ such that $Q = \mathcal{U}(ker\varphi)$. The **quotient** of A by Q is the coequaliser of $ker\varphi$. Its target model is denoted A/Q and is also called the **quotient** of A. The **congruence closure of** a binary relation Q on the domain of A is the least congruence on A containing Q. Given Γ be a set of conditional equations, a congruence $\equiv$ on A is **closed under Γ-substitutivity** iff for any $(\forall B)\langle s', t'\rangle$ **if** $\langle s, t\rangle$ in Γ and any morphism $h\colon B \to A$, $s;h\mathcal{U} \equiv t;h\mathcal{U}$ implies $s';h\mathcal{U} \equiv t';h\mathcal{U}$. The least congruence on A closed under Γ-substitutivity is denoted $\equiv^A_\Gamma$. In the usual concrete examples, a relation is closed under Γ-substitutivity iff it contains all the pairs generated as substitution instances of the equations in Γ.

The completeness of the CBEL proof theory depends on a finiteness condition for $\mathcal{U}$ which in the usual concrete cases corresponds to the fact that all operation (and relational) symbols take only a finite number of arguments. In (Diaconescu, 1994; Diaconescu, 1995; Goguen and Diaconescu, 1995) this is called **finitarity of** $\mathcal{U}$ and comes under different formulations corresponding to various abstraction levels.

Theorem 2. Completeness Theorem If the forgetful functor $\mathcal{U}$ is finitary and all equations in Γ have projective quantifiers, then:

1 the least congruence closed under Γ-substitutivity, denoted $\equiv^A_\Gamma$, exists;
2 $A/\equiv^A_\Gamma$ is the free Γ-model over A; and
3 $\Gamma \models (\forall A)\langle s, t\rangle$ iff $s \equiv^A_\Gamma t$.

The proof (which may be found in (Diaconescu, 1994; Diaconescu, 1995)) brings out the syntactic character of $\equiv^A_\Gamma$, showing it is the closure under the syntactic consequences of Γ using congruence and substitutivity as inference rules. Under

[Adjointness Framework]: Deduction Framework + the forgetful functor $\mathcal{U}$ has a left adjoint $\mathcal{F}$.

the congruence rule can be explicitated as reflexivity + symmetry + transitivity + operations, and by adding the **Simplifying Assumption**, $\mathcal{U}$ is finitary if it preserves filtered colimits.[9] Recall that a filtered colimit is the colimit of a filtered diagram and that a filtered diagram is a diagram for which any two nodes have and "upper bound" in the diagram. Preservation of filtered colimits by forgetful functors is a categorical concept of finiteness well established in categorical algebra. For example, it is well-known (Gabriel and Ulmer, 1971) that in a variety $\mathbb{A}lg_{\Sigma,E}$, an algebra A is finitely presented if and only if its representable hom-functor $\mathbb{A}lg_{\Sigma,E}(A,-)$ preserves filtered colimits.

4.4. *A Herbrand Theorem*

An important consequence of the most abstract version of the completeness result (Theorem 2) is a Herbrand theorem for CBEL. Our approach uses the categorical characterisation of Herbrand universes as initial models suggested in (Goguen and Meseguer, 1987). Herbrand theorem for CBEL admits various formulations corresponding to different abstraction levels. The one that fits best the level of presentation of this paper is the following:

[9] See (Diaconescu, 1995; Diaconescu, 1994) for a more general version of this result using a finiteness condition on the category of domains instead of the **Simplifying Assumption**.

Corollary 4.1. Herbrand Theorem Assume the **Adjointness Framework** and the **Simplifying Assumption**. If $\mathbb{A}$ has an initial model $0_{\mathbb{A}}$, $\mathcal{U}$ preserves filtered colimits, and all quantifiers of equations in Γ are projective, then:

1 the initial model of Γ exists; let us denote it 0_Γ;
2 $\Gamma \models (\exists A)q$ iff $0_\Gamma \models (\exists A)q$, for any $\mathcal{U}$-query $(\exists A)q$ and any model A; and
3 if in addition $\mathcal{U}$ has *non-empty sorts* (i.e., for each domain y there exists a map $y \to 0_{\mathbb{A}}\mathcal{U}$) and A is projective, $\Gamma \models (\exists A)q$ iff $\Gamma \models (\forall y\mathcal{F})q; h\mathcal{U}$ for some domain y and some model morphism $h\colon A \to y\mathcal{F}$.

In the usual concrete examples the non-empty sorts condition corresponds to the fact that the domain of the initial model has all carriers non-empty. The non-empty sorts version of Herbrand theorem provides foundations for solving queries using techniques like resolution and paramodulation. The proof of this result in a more general setting, without the **Simplifying Assumption**, can be found in (Diaconescu, 1994; Diaconescu, 1995).

4.5. *The Category-based Equational Logic Institution*

In (Diaconescu, 1996b) the CBEL institution plays the central role for the study of equational logic programming modularisation in the general category-based equational logic programming setting; in this paper we use it for defining the constraint logic internally to any CBEL.

To get an institution for CBEL, we need to define signature morphisms for CBEL, and define how models and sentences translate along signature morphisms; in particular, we need to know how quantifiers translate along signature morphisms. Then we must check that the satisfaction relation between CBEL models and the sentences in Definition 4.1 satisfies the so-called Satisfaction Condition for institutions (Goguen and Burstall, 1992).

Definition 4.2. A **CB equational signature** is a functor $\mathcal{U}\colon \mathbb{A} \to \mathbb{X}$, and a **morphism of CB equational signatures** is a pair of functors $\langle(\mathbb{A}' \xrightarrow{\mathcal{M}} \mathbb{A}), (\mathbb{X}' \xrightarrow{\mathcal{D}} \mathbb{X})\rangle\colon (\mathbb{A} \xrightarrow{\mathcal{U}} \mathbb{X}) \to (\mathbb{A}' \xrightarrow{\mathcal{U}'} \mathbb{X}')$ such that $\mathcal{M}; \mathcal{U} = \mathcal{U}'; \mathcal{D}$ and $\mathcal{D}$ has a left adjoint.

Notice that a morphism of CB equational signatures is liberal iff $\mathcal{M}$ has a left adjoint.

The following shows the analogy of concepts in MSA and CBEL:

MSA	CBEL
signature (S, Σ)	functor $\mathcal{U}\colon \mathbb{A} \to \mathbb{X}$
S	$\mathbb{X}$
Σ	$\mathbb{A}$
$\varphi = \langle f, g\rangle\colon (S,\Sigma) \to (S',\Sigma')$	$\langle \mathcal{M}, \mathcal{D}\rangle\colon \mathcal{U} \to \mathcal{U}'$
$f\colon S \to S'$	$\mathcal{D}\colon \mathbb{X}' \to \mathbb{X}$
$g\colon \Sigma \to \Sigma'$	$\mathcal{M}\colon \mathbb{A}' \to \mathbb{A}$
$\mathbb{S}et^f\colon \mathbb{S}et^{S'} \to \mathbb{S}et^S$	$\mathcal{D}$
$\mathbb{A}lg(\varphi)\colon \mathbb{A}lg_{\Sigma'} \to \mathbb{A}lg_\Sigma$	$\mathcal{M}\colon \mathbb{A}' \to \mathbb{A}$
Σ-equation	$\mathcal{U}$-equation

Before defining translations of equations along CB equational signature morphisms, we look

again at the many sorted case. A function $f\colon S \to S'$ translates an S-sorted set X into the S'-sorted set $X^\sim$ by taking the (pointwise) left Kan extension of f along X. Given a MSA signature morphism $\varphi = \langle f, g\rangle\colon (S,\Sigma) \to (S',\Sigma')$, the term algebra $T_{\Sigma'}(X^\sim)$ is exactly the free expansion of $T_\Sigma(X)$ along φ. From this, we conclude that:

Translations of quantifiers are free extensions along signature morphisms.

This also covers quantifiers that are not free models. The translation of equations along signature morphisms in MSA is a particular case of the following:

Definition 4.3. Let $\langle \mathcal{M}, \mathcal{D}\rangle$ be a liberal morphism of CB equational signatures $(\mathbb{A} \xrightarrow{\mathcal{U}} \mathbb{X}) \to (\mathbb{A}' \xrightarrow{\mathcal{U}'} \mathbb{X}')$. Then the $\mathcal{U}$-equation $(\forall A)\langle s, t\rangle$ translates to the $\mathcal{U}'$-equation $(\forall A^{\mathcal{M}})\langle s^*, t^*\rangle$,

$$\begin{array}{ccc} I & \xrightarrow{I\theta} & I^{\mathcal{D}}\mathcal{D} \\ {\scriptstyle t}\downarrow{\scriptstyle s} & & {\scriptstyle t^*\mathcal{D}}\downarrow{\scriptstyle s^*\mathcal{D}} \\ A\mathcal{U} & \xrightarrow[A\alpha\mathcal{U}]{} & A^{\mathcal{M}}\mathcal{M}\mathcal{U} = A^{\mathcal{M}}\mathcal{U}'\mathcal{D} \end{array}$$

where $\_^{\mathcal{D}}$ denotes the left adjoint of $\mathcal{D}$, $\_^{\mathcal{M}}$ denotes the left adjoint of $\mathcal{M}$, α and θ denote the units of the adjunctions determined by $\mathcal{M}$ and $\mathcal{D}$, and s^* and t^* denote the unique "extensions" of $s; A\alpha\mathcal{U}$ and $t; A\alpha\mathcal{U}$ to maps in $\mathbb{X}'$. Similarly, the $\mathcal{U}$-query $(\exists A)\langle s, t\rangle$ translates to the $\mathcal{U}'$-query $(\exists A^{\mathcal{M}})\langle s^*, t^*\rangle$.

The following result (from (Diaconescu, 1994; Diaconescu, 1996b)) is the Satisfaction Condition for equational logic systems; it extends to conditional equations without difficulty. A proof of institutionality for each example in Section 4.2 can be obtained by specialising the proof of the following.

Theorem 3. Let $\langle \mathcal{M}, \mathcal{D}\rangle$ be a liberal morphism of CB equational signatures $(\mathbb{A} \xrightarrow{\mathcal{U}} \mathbb{X}) \to (\mathbb{A}' \xrightarrow{\mathcal{U}'} \mathbb{X}')$. Then for any model $B \in |\mathbb{A}'|$ and any sentence $(\lambda A)\langle s, t\rangle$ with $\lambda \in \{\forall, \exists\}$,

$$B \models_{\mathcal{U}'} (\lambda A^{\mathcal{M}})\langle s^*, t^*\rangle \text{ iff } B\mathcal{M} \models_{\mathcal{U}} (\lambda A)\langle s, t\rangle\ .$$

5. (Category-based Equational) Constraint Logic

Constraint logic is central to our approach to constraint logic programming in that it is the logic underlying this programming paradigm. This matches the principle of *logical programming* introduced by Goguen and Meseguer in (Goguen and Meseguer, 1987). This section does actually more than setting up the logic underlying ECLP; it also shows how constraint logic is a CBEL, which means ECLP is semantically integrated to the equational logic programming paradigm.

On the other hand, we develop constraint logic internally to CBEL (we might thus call this "category-based equational constraint logic", but for simplicity of terminology we will stick with "constraint logic"); in this way ECLP is accommodated by any logical system that is a CBEL.

5.1. *Generalized Polynomials*

Generalized polynomials constitute the basic syntactic ingredient in constraint logic; the rôle played by the terms in ordinary logic is played by *generalized polynomials*[10] in constraint logic. In particular cases, generalized polynomials are term-like structures involving operator symbols, variables, and elements of a fixed model referred as the "built-in model".

Generalized polynomials can be regarded as elements of models in the same way as ordinary terms are regarded as elements of [free] models as a basis for a semantic approach to sentences and satisfaction in CBEL. The universal property of the models of generalized polynomials allows a more general definition that extends the concept of generalized polynomial to the semantic case when models play the rôle of the collections of variables and model morphisms play the rôle of the valuation maps.

Definition 5.1. Let $(\mathbb{A} \xrightarrow{\mathcal{U}} \mathbb{X})$ be a CB equational signature of built-ins, $A \in |\mathbb{A}|$ be a model of built-ins, and
$\langle \mathcal{M}, \mathcal{D} \rangle \colon \mathcal{U} \to \mathcal{U}'$ be a morphism of CB equational signatures from $\mathcal{U}$ to any CB equational signature $(\mathbb{A}' \xrightarrow{\mathcal{U}'} \mathbb{X}')$. For any $\mathcal{U}'$-model $B \in |\mathbb{A}'|$, the **model of generalized polynomials over** B is the coproduct $A^{\mathcal{M}} + B$, and it is denoted as $A[B]$.

In practice, B is a free model over a collection X of variables (i.e., a domain $X \in |\mathbb{X}'|$ in the abstract case of CBEL). More precisely, if $\mathcal{F}' \colon \mathbb{X}' \to \mathbb{A}'$ is a left-adjoint to $\mathcal{U}'$, then the model of generalized polynomials is $A[X\mathcal{F}']$, usually denoted as $A[X]$. This generalizes universal algebra polynomials when $\mathcal{U} = \mathcal{U}'$ is a forgetful functor from a category of (unsorted) algebras to $\mathbb{S}et$. However, the best known example still comes from linear algebra:

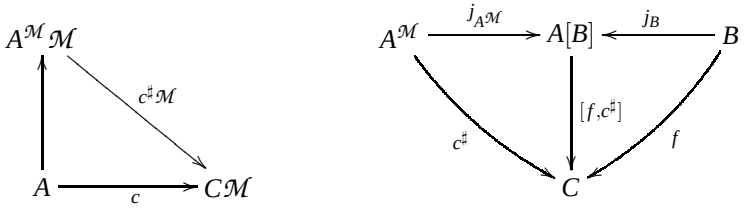
Example 5.1. Let $\vec{X}$ be a set of variables. $\mathbb{R}[\vec{X}]$ is the ring of polynomials over $\vec{X}$ and with real numbers as coefficients. In this example, the signature $\mathcal{U}$ of built-ins is a ring signature,[11] and $\mathcal{U} = \mathcal{U}'$. The model of the built-ins is $\mathbb{R}$, the usual ring of real numbers.

5.2. *Internal Constraint Satisfaction*

The following provides a formal definition for constraint sentences, constraint models, and a satisfaction relation between them, by following the principle of the preservation of the built-ins.

Definition 5.2. Let $\langle \mathcal{M}, \mathcal{D} \rangle \colon (\mathbb{A} \xrightarrow{\mathcal{U}} \mathbb{X}) \to (\mathbb{A}' \xrightarrow{\mathcal{U}'} \mathbb{X}')$ be any liberal morphism of CB equational signatures, and consider a model $A \in |\mathbb{A}|$ for the built-ins.
A **constraint model** is an interpretation of A into the reduct of a model in $\mathbb{A}'$, i.e., an arrow $c \colon A \to C\mathcal{M}$ with $C \in |\mathbb{A}'|$. A model morphism $h \colon c \to c'$ is a map $C \to C'$ in $\mathbb{A}'$ such that $c; h\mathcal{M} = c'$. A constraint model $c \colon A \to C\mathcal{M}$ is **conservative** iff c is an isomorphism.
A **constraint identity** on $B \in |\mathbb{A}'|$ is a binary relation $\langle s, t \rangle$ on $(A[B])\mathcal{U}'$. An identity $\langle s, t \rangle$ in B is satisfied in a model $A \xrightarrow{c} C\mathcal{M}$ with respect to a model morphism $f \colon B \to C$ iff $s; [f, c^\sharp]\mathcal{U}' = t; [f, c^\sharp]\mathcal{U}'$, where $c^\sharp$ is the unique 'extension' of c to a model morphism $A^{\mathcal{M}} \to C$.

[10] The ordinary polynomials from linear algebra are an instantiation of this notion. The word *generalized* plays here the same rôle as the word *general* plays in the so-called "general algebra."
[11] More precisely, the forgetful functor from the category of rings to $\mathbb{S}et$.



A **constraint equation** is a universally quantified expression $(\forall B)\langle s,t\rangle$ where $B \in |\mathbb{A}'|$ is the model representing the quantifier and $\langle s,t\rangle$ is a constraint identity on B. A constraint model c **satisfies** $(\forall B)\langle s,t\rangle$ iff c satisfies the identity $\langle s,t\rangle$ for all model morphisms $f\colon B \to C$. This is written $c \models (\forall B)\langle s,t\rangle$. This definition extends to **constraint conditional equations**, **queries**, and their satisfaction by constraint models in the same manner.

Example 5.2. An example of a constraint equation within the context of Example 1.1 is

```
< 3.14 * X , Y > + - < Y , 3.14 * X > = 0
```

Notice that although this equation is *not* satisfied by the standard model $\mathbb{R}^2$, the constraint model $\mathbb{R}+$ does satisfy it.
An example of a constraint query within the same context is

```
3 * < X , Y > = < Y , Z > ;
2.79 * X + Y < Z  = true .
```

Finding a solution to this query in the standard model $\mathbb{R}^2$ reduces by the application of a rewrite step followed by a simplification step to finding a solution for the system of linear inequalities

```
3 * X = Y ;
3 * Y = Z ;
2.79 * X + Y < Z = true .
```

A stronger version (and of higher practical relevance) of CB constraint logic can be obtained by restricting the semantics only to conservative models.

Definition 5.3. A constraint sentence ρ is a **conservative** consequence of a set of constraint sentences Γ iff $M \models \rho$ whenever $M \models \Gamma$ for all *conservative* models M. This is denoted as $\Gamma \models\!\!\!\cong \rho$.

Example 5.3. Consider the OBJ3 or CafeOBJ conditional equations. They are expression of the form

$$(\forall X)t \;=\; t' \;\textbf{if}\; C$$

where C can be any `Bool`-sorted term. The usual MSA (or OSA) equations appear as a special case by using the built-in semantic equality[12] for the identities in the condition.
However, in the context of Example 1.2 if we view the condition C as an identity between generalized polynomials

$$C \;=\; \texttt{true}$$

[12] Implemented in both languages and denoted by `_==_`. Its implementation realizes the true semantic equality under the confluence and termination of the specification regarded as a rewrite system.

then the sentences of equational logic with a built-in type of Booleans are just constraint sentences. In this case the constraint models correspond to models of real OBJ or CafeOBJ modules, and the constraint satisfaction of OBJ or CafeOBJ conditional equations is the expected one.

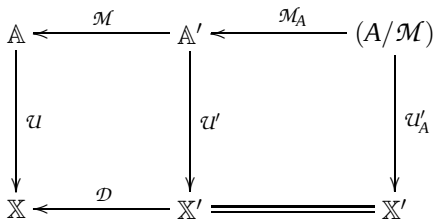
Also notice that the background CBEL can be realized as the trivial constraint logic for the case when the model of built-ins is just the initial model in $\mathbb{A}$.

5.3. *Constraint Logic is a CBEL*

The crucial technical idea underlying our semantics of ECLP is to fit constraint logic to CBEL. Whilst such an integration of constraint logic into equational logic cannot be achieved within the usual concrete algebraic or model theoretic approaches (no notion of algebraic signature being abstract enough for this purpose), it works at our level of abstraction. We consider this a good example of the benefits the use of abstract model theoretic methodology[13] can bring to Computing Science. This idea is summarized in the following slogan, and formalized by the subsequent definition:

Constraint logic = equational logic in a special CB equational signature.

Definition 5.4. Let $\langle \mathcal{M}, \mathcal{D} \rangle \colon (\mathbb{A} \xrightarrow{\mathcal{U}} \mathbb{X}) \to (\mathbb{A}' \xrightarrow{\mathcal{U}'} \mathbb{X}')$ be a liberal morphism of CB equational signatures. Then any model $A \in |\mathbb{A}|$ determines a forgetful functor $\mathcal{U}'_A \colon (A/\mathcal{M}) \to \mathbb{X}'$, such that $\mathcal{U}'_A = \mathcal{M}_A; \mathcal{U}'$, where $\mathcal{M}_A$ is the forgetful functor $(A/\mathcal{M}) \to \mathbb{A}'$.



Proposition 5.1. If $\mathcal{U}'$ is faithful and preserves pullbacks, then $\mathcal{U}'_A$ is faithful and preserves pullbacks.

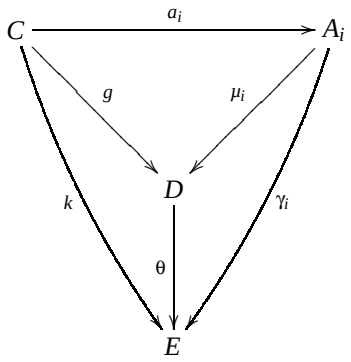
Proof. $\mathcal{M}$ preserves pullbacks because it is a right adjoint. By using this fact, it is straightforward to show that the forgetful functor $\mathcal{M}_A \colon (A/\mathcal{M}) \to \mathbb{A}'$ creates pullbacks, thus it preserves them too. $\mathcal{U}'_A$ preserves pullbacks as a composite of two pullback preserving functors.
$\mathcal{U}'_A$ is faithful as a composite of two faithful functors, since the forgetful functor $\mathcal{M}_A \colon (A/\mathcal{M}) \to \mathbb{A}'$ is faithful. □

Proposition 5.2. Let $\langle \mathcal{M}, \mathcal{D} \rangle \colon (\mathbb{A} \xrightarrow{\mathcal{U}} \mathbb{X}) \to (\mathbb{A}' \xrightarrow{\mathcal{U}'} \mathbb{X}')$ be a liberal morphism of CB equational signatures. Then for any model $A \in |\mathbb{A}|$:

1 there is an isomorphism of categories $(A/\mathcal{M}) \cong (A^{\mathcal{M}}/\mathbb{A}')$;
2 if $\mathbb{A}'$ has binary coproducts, then $\mathcal{M}_A$ has a left adjoint; and
3 the forgetful functor $\mathcal{M}_A$ creates filtered colimits.

[13] In the sense of CBEL and institutions.

Proof.
1. This isomorphism is given by the adjunction isomorphism $\mathbb{A}(A, B\mathcal{M}) \cong \mathbb{A}'(A^{\mathcal{M}}, B)$ mapping any $b\colon A \to B\mathcal{M}$ to $b^\sharp\colon A^{\mathcal{M}} \to B$.
2. The forgetful functor $(C/\mathbb{A}') \to \mathbb{A}'$ has a left adjoint for any $C \in |\mathbb{A}'|$ mapping each object A' to $C \xrightarrow{j_C} C + A'$. Then we apply 1.
3. We first show that for any model $C \in |\mathbb{A}'|$, the forgetful functor $(C/\mathbb{A}') \to \mathbb{A}'$ creates filtered colimits. Then we take $C = A^{\mathcal{M}}$ and apply 1.
Let $\{a_i\}_{i\in I}$ be a filtered diagram in $(C/\mathbb{A}')$. The forgetful functor $(C/\mathbb{A}') \to \mathbb{A}'$ maps this diagram to a filtered diagram $\{A_i\}_{i\in I}$ in $\mathbb{A}'$. Consider $\mu\colon A \to D$ a colimit of this diagram in $\mathbb{A}'$. We define $g\colon C \to D$ as $a_i;\mu_i$ for $i \in |I|$; the correctness of this definition is given by the fact that $a_i;\mu_i = a_j;\mu_j$ for all $i, j \in |I|$ because of the filteredness of I.



Now, we show that μ is a colimiting co-cone $a \to g$ in $(C/\mathbb{A}')$. Consider another co-cone $\gamma\colon a \to k$ in $(C/\mathbb{A}')$, where $k\colon C \to E$. γ is also a co-cone $A \to E$ in $\mathbb{A}'$. By the universal property of μ as a colimiting co-cone in $\mathbb{A}'$, there exists a unique arrow $\theta\colon D \to E$ such that $\mu;\theta = \gamma$ in $\mathbb{A}'$. All it remains to be shown is that θ is a map $g \to k$. But $g;\theta = a_i;\mu_i;\theta$ for some $i \in |I|$. Since $\mu_i;\theta = \gamma_i$ we deduce $g;\theta = k$. □

Corollary 5.1. The constraint logic determined by $\langle \mathcal{M}, \mathcal{D}\rangle\colon (\mathbb{A} \xrightarrow{\mathcal{U}} \mathbb{X}) \to (\mathbb{A}' \xrightarrow{\mathcal{U}'} \mathbb{X}')$ and a built-in model $A \in |\mathbb{A}|$ is the CBEL (satisfying the **Basic Framework**) determined by the forgetful functor $\mathcal{U}'_A\colon (A/\mathcal{M}) \to \mathbb{X}'$.

Proof. By noticing that constraint models, sentences, and satisfaction are respectively $\mathcal{U}'_A$-models, sentences, and satisfaction. □

This result plays a crucial rôle for the development of the results in this paper, such as the Herbrand Theorem (Theorem 4) and the complete proof theory of Section 6.1 for CB constraint logic.

6. A Herbrand Theorem for ECLP

Herbrand theorem for constraint logic provides mathematical foundations for the concept of constraint solving in the same way the original Herbrand theorem provides foundations for traditional logic programming, and Herbrand theorem for order sorted Horn clause logic with equality provides foundations for equational logic programming in Eqlog (Goguen and Meseguer, 1987).

Our approach is to instantiate the CBEL version of Herbrand Theorem 4.1 to the particular case of constraint logic viewed as the CBEL determined by the forgetful functor $\mathcal{U}'_A$ of Definition 5.2.

Theorem 4. Let $\langle \mathcal{M}, \mathcal{D} \rangle \colon (\mathbb{A} \xrightarrow{\mathcal{U}} \mathbb{S}et^S) \to (\mathbb{A}' \xrightarrow{\mathcal{U}'} \mathbb{S}et^{S'})$ be a liberal morphism of CB equational signatures. Fix any model $A \in |\mathbb{A}|$. Assume the **Adjointness Framework** for $\mathcal{U}'$, that $\mathbb{A}'$ has binary coproducts, and that $\mathcal{U}'$ preserves filtered colimits.
Consider a collection Γ of conditional constraint equations with projective quantifiers, and a $\mathcal{U}'$-constraint query $(\exists B)q$ with $B \in |\mathbb{A}'|$ projective. Then,

1 there exists the initial Γ-constraint model 0_Γ;
2 $\Gamma \models (\exists B)q$ iff $0_\Gamma \models (\exists B)q$; and
3 if $\mathcal{U}'$ has non-empty sorts, then $\Gamma \models (\exists B)q$ iff $\Gamma \models (\forall y)q; [h, j_{A^{\mathcal{M}}}]$ for some domain $y \in |\mathbb{X}'|$ and some model morphism $h \colon B \to A[y]$.

Proof. The basis of this proof is to regard the constraint sentences (either equations or queries) as ordinary $\mathcal{U}'_A$-sentences (in the sense of Definition 4.1). Any quantifier B of a constraint sentence appears as $A\eta; j_{A^{\mathcal{M}}}\mathcal{M}$ in the rôle of the quantifier of the corresponding $\mathcal{U}'_A$-sentence.

$$A \xrightarrow{A\eta} A^{\mathcal{M}}\mathcal{M} \xrightarrow{j_{A^{\mathcal{M}}}} (B + A^{\mathcal{M}})\mathcal{M}$$

The category of constraint models is $(A/\mathcal{M})$ and the satisfaction relation between constraint models and constraint sentences reduces to CB equational satisfaction (cf. Corollary 5.1), so we have only to check the hypotheses of Theorem 4.1 for the forgetful functor $\mathcal{U}'_A$ and to explicitate the instantiation of its conclusions to the CBEL determined by $\mathcal{U}'_A$.
Deduction Framework for $\mathcal{U}'_A$: $(A/\mathcal{M})$ has pullbacks because of 1. of Proposition 5.2, because $\mathcal{M}_A$ creates limits, and because $\mathbb{A}'$ has pullbacks. Similarly, $(A/\mathcal{M})$ has coequalisers ($\mathcal{M}_A$ creates coequalisers by 3. of Proposition 5.2).
Adjointness Framework for $\mathcal{U}'_A$: $\mathcal{U}'_A$ has a left-adjoint which is the composite of two left adjoints $\mathbb{S}et^{S'} \xrightarrow{\mathcal{F}'} \mathbb{A}' \longrightarrow (A/\mathcal{M})$ (cf. 2. of Proposition 5.2 for the existence of the left-adjoint $\mathbb{A}' \to (A/\mathcal{M})$).
$(A/\mathcal{M})$ *has the initial model* $A \xrightarrow{A\eta} A^{\mathcal{M}}\mathcal{M}$.
$\mathcal{U}'_A$ *preserves filtered colimits* as a composite of two filtered preserving functors (see 3. of Proposition 5.2 for the preservation of filtered colimits by the forgetful functor $(A/\mathcal{M}) \to \mathbb{A}'$).
Projectivity of quantifiers: We have to show that if B is projective in $\mathbb{A}'$ then $A\eta; j_{A^{\mathcal{M}}}\mathcal{M}$ is projective in $(A/\mathcal{M})$. Notice that $A\eta; j_{A^{\mathcal{M}}}\mathcal{M}$ is free over B with respect to the forgetful functor $\mathcal{M}_A$. Therefore $A\eta; j_{A^{\mathcal{M}}}\mathcal{M}$ is projective because left-adjoint to coequalisers preserving functors preserve projectivity (this rather simple categorical lemma can be easily checked by the reader).
Non-empty sorts for $\mathcal{U}'_A$: Because $\mathcal{U}'$ has non-empty sorts, for any domain $y \in |\mathbb{X}'|$, there exists at least one arrow $y \to 0_{\mathbb{A}'}\mathcal{U}'$, where $0_{\mathbb{A}'}$ is the initial model in $\mathbb{A}'$. Therefore, there exists at least one arrow $y \to A^{\mathcal{M}}\mathcal{U}'$, which is $y \to 0_{\mathbb{A}'}\mathcal{U}' \to A^{\mathcal{M}}\mathcal{U}' = (A\eta)\mathcal{U}'_A$. □

The following is a proof-theoretic corollary of the construction of Theorem 4, giving a complete set of inference rules for CB constraint logic deduction:

Corollary 6.1. Under the hypotheses of Theorem 4, $\Gamma \models (\forall B)\ p = p'$ iff $(\forall B)\ p = p'$ can be deduced by the following inference rules:

$$\text{[reflexivity]}\quad \frac{}{(\forall B)\ p = p}$$

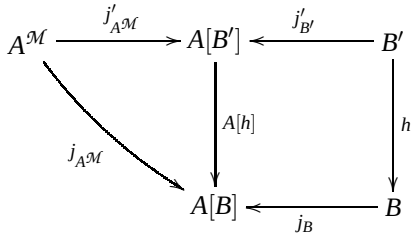
$$\text{[symmetry]}\quad \frac{(\forall B)\ p = p'}{(\forall B)\ p' = p}$$

$$\text{[transitivity]}\quad \frac{(\forall B)\ p = p' \qquad (\forall B)\ p' = p''}{(\forall B)\ p = p''}$$

$$\text{[congruence]}\quad \frac{(\forall B)\ p = p'}{(\forall B)\ p^\sharp \mathcal{U}' = p'^\sharp \mathcal{U}'}$$

$$\text{[substitutivity]}\quad \frac{(\forall B)\ (r;A[h])\mathcal{U}' = (r';A[h])\mathcal{U}'}{(\forall B)\ (p;A[h])\mathcal{U}' = (p';A[h])\mathcal{U}'}$$

where B is any model in $\mathbb{A}'$, p, p', p'' are generalized polynomials in $A[B]$, $p^\sharp, p'^\sharp$ the (unique) extensions of $p, p' : I \to A[B]\mathcal{U}'$ to arrows $I\mathcal{F}' \to A[B]$, $(\forall B')\ p = p'$ **if** $r = r'$ is any conditional constraint equation in Γ, $h : B' \to B$ is any model morphism, and $A[h]$ is the unique arrow between the models of generalized polynomials such that the following diagram commutes:

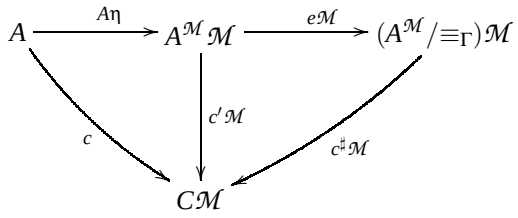


In practice, it rarely happens that the sentences in Γ involve the built-in model A. Usually, the sentences in Γ don't involve any elements of the built-in model (i.e., Γ contains only $\mathcal{U}'$-sentences, if we were to use the notation from the discussion in the Introduction) and only the queries appear as full constraint sentences involving elements from the built-in model. In this case, the initial constraint model 0_Γ has a simpler representation as a quotient of the free expansion of the built-in model.

Notice that $\mathcal{U}'$-sentences can be canonically viewed as constraint sentences (i.e., $\mathcal{U}'_A$-sentences) via the translation along the morphism of CB equational signatures $\langle \mathcal{M}_A, 1_{\mathbb{X}'} \rangle : \mathcal{U}' \to \mathcal{U}'_A$ (see Definition 4.3).

Proposition 6.1. Assuming the hypotheses of Theorem 4, suppose that Γ contains only $\mathcal{U}'$-equations. Then the initial constraint model 0_Γ is isomorphic to the canonical map $!_\Gamma = A \xrightarrow{A\eta} A^{\mathcal{M}}\mathcal{M} \xrightarrow{e\mathcal{M}} (A^{\mathcal{M}}/\equiv_\Gamma)\mathcal{M}$, where $\equiv_\Gamma$ is the least congruence on $A^{\mathcal{M}}$ closed under Γ-substitutivity.

Proof. We show that $!_\Gamma$ satisfies the initiality property in the full subcategory of $(A/\mathcal{M})$ of all models satisfying $\hat{\Gamma}$, where $\hat{\Gamma}$ is the translation of Γ along $\langle \mathcal{M}_A, 1_{\mathbb{X}'} \rangle$.
Let $c : A \to C\mathcal{M}$ be any constraint model satisfying $\hat{\Gamma}$. By the Satisfaction Condition (Theorem 3), this is equivalent to $C \models \Gamma$. All we have to prove is that there exists a unique arrow $c^\sharp : A^{\mathcal{M}}/\equiv_\Gamma \to C$ such that $!_\Gamma; c^\sharp \mathcal{M} = c$.



By the universal property of the free extension $A\eta$ along $\mathcal{M}$, there exists a unique map $c' : A^{\mathcal{M}} \to C$ such that $A\eta; c'\mathcal{M} = c$. By the universal property of e (Theorem 2), there exists a unique map $c^\sharp : A^{\mathcal{M}}/\equiv_\Gamma \to C$ such that $e; c^\sharp = c'$. □

In the case of order sorted Horn clause logic with equality, Goguen and Meseguer proved the existence of initial constraint models for the particular case of Γ containing only Σ'-sentences (Goguen and Meseguer, 1987). This crucial result for the semantics of Eqlog can obtained as an instantiation of our previous results.

6.1. *The conservative case*

For the rest of this section we investigate practically important sufficient conditions for which Herbrand Theorem can be lifted to the conservative case. As pointed out by Example 1.2, the Herbrand model for a constraint specification Γ should be the initial *conservative* model; let us denote is by $\tilde{0}_\Gamma$.

Corollary 6.2. Under the hypotheses of Theorem 4, let Γ be any constraint logic specification, and suppose it has an initial conservative model $\tilde{0}_\Gamma$. Then for any query $(\exists B)q$

1 $\Gamma \mathrel{\dot{\models}} (\exists B)q$ iff $\tilde{0}_\Gamma \models (\exists B)q$ if $0_\Gamma \models (\exists B)q$ iff $\Gamma \models (\exists B)q$, and
2 if the (unique) arrow $0_\Gamma \to \tilde{0}_\Gamma$ is a coequalizer and $\mathcal{U}'$ has non-empty sorts, then $\Gamma \mathrel{\dot{\models}} (\exists B)q$ iff $\Gamma \mathrel{\dot{\models}} (\forall y)q; [h, j_{A^{\mathcal{M}}}]$ for some domain $y \in |\mathbb{X}'|$ and some model morphism $h : B \to A[y]$.

Corollary 6.3. Assume the hypotheses of Theorem 4. If 0_Γ is conservative, then

$$\Gamma \mathrel{\dot{\models}} (\exists B)q \text{ iff } 0_\Gamma \models (\exists B)q$$

Proof. By noticing that in this case 0_Γ is the initial conservative constraint model satisfying Γ. □

The following gives an important sufficient condition for which 0_Γ is conservative:

Proposition 6.2. Assuming the hypotheses of Theorem 4, further suppose that for all projective models B, the corresponding models $A[B]$ of generalized polynomials are conservative. Then 0_Γ is conservative.

Proof. The proof of this result repeats the steps of the proof of Theorem 4 for the CBEL determined by the forgetful functor $(A|\mathcal{M}) \to (A/\mathcal{M}) \xrightarrow{\mathcal{U}'_A} \mathbb{X}'$ rather than just $\mathcal{U}'_A$, where $(A|\mathcal{M})$ is the full subcategory of $(A/\mathcal{M})$ of conservative constraint models. Notice that the full subcategory embedding $(A|\mathcal{M}) \to (A/\mathcal{M})$ creates limits and filtered colimits, a conservative model in $(A|\mathcal{M})$ is projective if and only if it is projective as a model in $(A/\mathcal{M})$, and that because all models of generalized polynomials are conservative, the forgetful functor $(A|\mathcal{M}) \to \mathbb{A}'$ has a left adjoint

which is the same as the left adjoint to the forgetful $(A/\mathcal{M}) \to \mathbb{A}'$ (constructing the models of generalized polynomials). This means that all arguments from the proof of Theorem 4 carry from the CBEL determined by $\mathcal{U}'_A$ to the CBEL determined by $(A|\mathcal{M}) \to (A/\mathcal{M}) \xrightarrow{\mathcal{U}'_A} \mathbb{X}'$. Therefore $\tilde{0}_\Gamma$ exists.
Because the initial model of $(A/\mathcal{M})$ is the same with the initial model of $(A|\mathcal{M})$ and since the construction of the least congruence closed under Γ-substitutivity is obtained by using only free extensions, pullbacks, and coequalizers in $(A|\mathcal{M})$, which are the same with doing them in $(A/\mathcal{M})$ in the virtue of the above arguments, this means that $\tilde{0}_\Gamma = 0_\Gamma$. □

This situation corresponds to Example 1.1 and occurs very often in practice. Related in goal to the above result, we mention the Goguen and Meseguer results (Goguen and Meseguer, 1987) giving, in the context of order sorted Horn clause logic with equality and of Γ containing only sentences without built-ins, a set of conditions that guarantee that 0_Γ is conservative by imposing some restrictions[14] on the sentences in Γ.

There are however interesting practical examples (such as Example 1.2) when 0_Γ is not conservative. In general, in such cases, even the existence of the initial conservative constraint model cannot be guaranteed. Finding out sufficient conditions for the existence of the initial conservative model $\tilde{0}_\Gamma$ when this is different from 0_Γ is an important topic of future research.

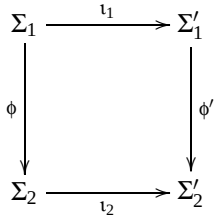
7. Constraint Institutions

In this section we study constraint logic from the point of view of institutions. This brings us to the concept of *constraint institution* which internalizes the concept of constraint models to any institution. In particular, constraint logic can be obtained as a constraint institution on top of the CBEL institution, on the other hand, we show how the constraint logic institution can be regarded as a CBEL sub-institution. One consequence of this result is that ECLP supports modularization in the style of OBJ and Clear; the mathematical results on modularization based on institutions can be directly applied to any ECLP system rigorously based on some version of constraint logic. Finally, we show how constraint institutions serve as as a semantic framework for combining constraint solvers over different data types.

Definition 7.1. Let $\mathfrak{I} = (\mathbb{S}ign, \text{MOD}, Sen, \models)$ be any institution. A **constraint institution over** $\mathfrak{I}$ is an institution $\mathfrak{I}^{\mathcal{C}} = (\mathbb{S}ign^{\mathcal{C}}, \text{MOD}^{\mathcal{C}}, Sen^{\mathcal{C}}, \models^{\mathcal{C}})$ such that

1 a signature in $\mathbb{S}ign^{\mathcal{C}}$ is a pair (A, ι) consisting of a signature morphism $\iota\colon \Sigma \to \Sigma'$ in $\mathbb{S}ign$ and a "built-in" Σ-model A (Σ is called the **signature of built-ins**),
2 a morphism $(h, \phi, \phi')\colon (A_1, \iota_1) \to (A_2, \iota_2)$ in $\mathbb{S}ign^{\mathcal{C}}$ consists of
   — a morphism between the signatures of built ins $\phi\colon \Sigma_1 \to \Sigma_2$,
   — a liberal "extension" $\phi'\colon \Sigma'_1 \to \Sigma'_2$ of ϕ, i.e., the diagram

[14] However, these restrictions are almost always met in practice.



commutes and $\text{MOD}(\phi')$ has a left-adjoint, and

— a model morphism $h\colon A_1 \to A_2{\restriction_\phi}$ interpreting the model of built-ins A_1 into the model of built-ins A_2.

Composition of signature morphisms is the obvious "pointwise" composition,

3 $\text{MOD}^{\mathcal{C}}(A,\iota)$ is a full subcategory of $(A/\text{MOD}(\iota))$ (on signatures), and
$\text{MOD}^{\mathcal{C}}(h,\phi,\phi')(A_2 \xrightarrow{c} C{\restriction_{\iota_2}}) = A_1 \xrightarrow{h} A_2{\restriction_\phi} \xrightarrow{c{\restriction_\phi}} (C{\restriction_{\phi'}}){\restriction_{\iota_1}}$ (on signature morphisms).

Example 7.1. Consider a module `CMPLX` specifying the complex numbers by adding a supersort `C` to `Real`, the constant `i : -> C`, extending the usual ring operators from reals to the complex numbers, etc. We may consider the signature of this specification as the signature of built-ins Σ_2 and ι_2 to be just adding an operation `<_,_> : Real Real -> C` . The built-in model for this specification is the ring of complex numbers. ϕ' maps the sort `Real` to `Real`, and `Vect` to `C` and also maps the ring operations in the obvious way, and `<_,_>` to `<_,_>`. h is just the ordinary inclusion of the reals into the complex numbers. Notice that by adding a new equation

```
< a , b > = a + b * i
```

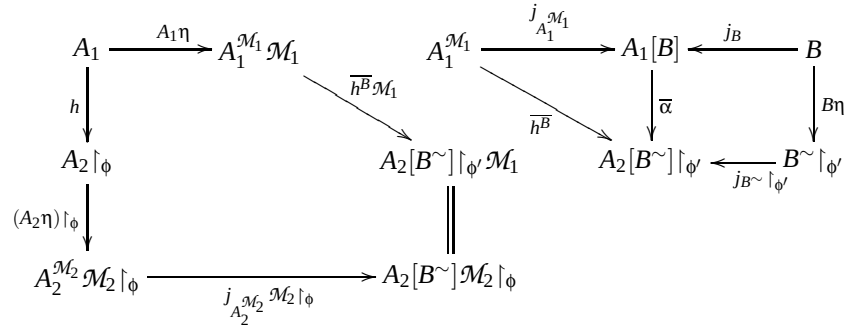
ι_2 gets the translation from the Euclidean plane to the complex numbers.

7.1. *The Constraint Logic Institution*

When instantiating Definition 7.1 to the CBEL institution one gets the signatures and models of constraint logic (see Definition 5.2). In order to fully define the institution of constraint logic we still need to define the translation of constraint sentences of Definition 5.2 along signature morphisms and prove the Satisfaction Condition.

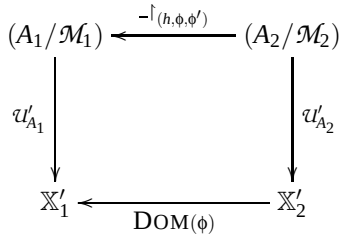
The Sentence Translation. Given a morphism of constraint logic signatures $(h,\phi,\phi')\colon (A_1,\langle\mathcal{M}_1,\mathcal{D}_1\rangle) \to (A_2,\langle\mathcal{M}_2,\mathcal{D}_2\rangle)$, where $\langle\mathcal{M}_i,\mathcal{D}_i\rangle\colon (\mathbb{A}_i \xrightarrow{\mathcal{U}_i} \mathbb{X}_i) \to (\mathbb{A}'_i \xrightarrow{\mathcal{U}'_i} \mathbb{X}'_i)$, $i = 1,2$, are morphisms of CB equational signatures, $A_1 \in |\mathbb{A}_1|$, $A_2 \in |\mathbb{A}_2|$, and given $B \in |\mathbb{A}'_1|$, then we have to define a translation mapping "generalized polynomials" from $A_1[X]$ to "generalized polynomials" in $A_2[X^\sim]$, where $B^\sim \in |\mathbb{A}'_2|$ is the free expansion of B along ϕ'.[15] This translation is given by the unique arrow $\overline{\alpha}\colon A_1[B] \to A_2[B^\sim]{\restriction_{\phi'}}$ making the following diagram commute:

[15] Here we apply the principle that translation of the quantifiers are just free expansions along signature morphisms; see Section 4.5.



In the particular case when B is the free model over some "variables" X, then $B^\sim$ appears also as a free model over the "variables" $X^\sim$, where $X^\sim$ is the free expansion of X along the domain translation component of ϕ. More concretely, if the domains are many sorted sets, this is the same as the Kan-extension translation defined in Section 4.5.

Proposition 7.1. When regarding constraint logic signatures as CBEL signatures (see Definition 5.4), the translation of generalized polynomials previously defined is the same as the translation of sentences (Definition 4.3) along the morphism of CB equational signatures $\langle \text{MOD}(h,\phi,\phi'), \text{DOM}(\phi)\rangle\colon \mathcal{U}'_{A_1} \to \mathcal{U}'_{A_2}$, where $\text{DOM}(\phi)$ is the domain functor component of ϕ.



Proof. From the construction of $\overline{\alpha}\colon A_1[B] \to A_2[B^\sim]$ (in the above paragraph) it is easy to check that

$$\overline{\alpha}\colon (A_1 \xrightarrow{A_1\eta;j_{A^{\mathcal{M}_1}}\mathcal{M}_1} A_1[B]\mathcal{M}_1) \longrightarrow (A_1 \xrightarrow{h;A_2\eta\restriction_\phi;j_{A^{\mathcal{M}_2}}\restriction_\phi} A_2[B^\sim]\mathcal{M}_2\restriction_\phi = A_2[B^\sim]\restriction_{\phi'}\mathcal{M}_1)$$

is a universal arrow. Therefore, $A_2[B^\sim]$ is free over $A_1[B]$ with respect to the forgetful functor $(A_2/\mathcal{M}_2) \to (A_1/\mathcal{M}_1)$.
In this way, the rôle of "$A\alpha$" of Definition 4.3 is played here by $\overline{\alpha}$, and the rôle of "A" by $A_1\eta; j_{A^{\mathcal{M}_1}}\mathcal{M}_1$. Since any polynomial in $A_1[B]$ (or in $A_2[B^\sim]$) can be regarded as an arrow $\bullet \to (A_1\eta; j_{A^{\mathcal{M}_1}}\mathcal{M}_1)\mathcal{U}'_{A_1}$ (or respectively $\bullet \to (A_2\eta\restriction_\phi; j_{A^{\mathcal{M}_2}}\restriction_\phi)\mathcal{U}'_{A_2}$), we can conclude the translation of constraint sentences previously defined is the same as the one introduced by Definition 4.3. □

Corollary 7.1. Constraint logic is a CBEL sub-institution, i.e.,

$$c \models_{(A_2,\langle\mathcal{M}_2,\mathcal{D}_2\rangle)} (h,\phi,\phi')\rho \text{ iff } h;c\restriction_\phi \models_{(A_1,\langle\mathcal{M}_1,\mathcal{D}_1\rangle)} \rho$$

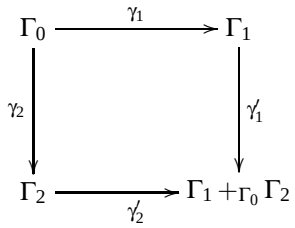
for any morphism of constraint logic signatures $(h,\phi,\phi')\colon (A_1,\langle\mathcal{M}_1,\mathcal{D}_1\rangle) \to (A_2,\langle\mathcal{M}_2,\mathcal{D}_2\rangle)$, any constraint model $c\colon A_2 \to C\mathcal{M}_2$, and any constraint sentence ρ over $(A_1,\langle\mathcal{M}_1,\mathcal{D}_1\rangle)$.

7.2. *Combining Constraint Solvers*

In this section we sketch some logical foundations of modular combination of constraint solvers over different data types by using the constraint logic institution. By a *constraint solver* we mean a decision procedure for solving constraint queries for a given specification or program. The topic of this section constitutes subject for further development, therefore the results presented here should be considered as a starting point for further research.

Firstly, we need to formulate the problem within our framework.

Definition 7.2. Given two constraint logic theories Γ_1 and Γ_2 sharing a common part Γ_0 (represented as a pair of constraint logic theory morphisms $\Gamma_1 \stackrel{\gamma_1}{\leftarrow} \Gamma_0 \stackrel{\gamma_2}{\rightarrow} \Gamma_2$), the **shared combination** of Γ_1 and Γ_2 is denoted by $\Gamma_1 +_{\Gamma_0} \Gamma_2$ and is defined as the following pushout

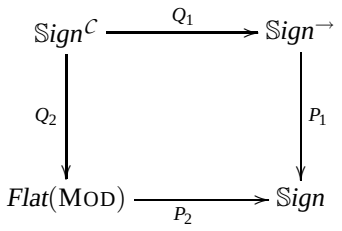


The pushout definition of the shared combination of constraint logic theories follows the Clear-OBJ modularization tradition (Goguen et al., ; Burstall and Goguen, 1980) of "putting together" theories via co-limits. For example, the sum (+) and the parameter instantiations for modules are realized as pushouts of theories (see (Diaconescu et al., 1993; Diaconescu, 1996b)).

The following result shows that under normal technical conditions the shared combination of theories exists in any constraint institution:

Theorem 5. Assume the institution $\mathfrak{I} = (\mathbb{S}ign, \text{MOD}, Sen, \models)$ has pushouts of signatures, is semi-exact, its signature morphisms are liberal, and the category of models for each signature has pushouts. Then any constraint institution $\mathfrak{I}^{\mathcal{C}}$ over $\mathfrak{I}$ has pushouts of theories.

Proof. By the fundamental result on the existence of co-limits of theories in institutions (Goguen and Burstall, 1992), it is enough to prove that $\mathfrak{I}^{\mathcal{C}}$ has pushouts of signatures.
Notice that the following square



is a pullback, where $\mathbb{S}ign^{\rightarrow}$ is the functor category $\mathbb{C}at((\bullet \rightarrow \bullet) \rightarrow \mathbb{S}ign)$, P_1 is the projection on the source, and P_2 is the projection on the signature component.
In virtue of Theorem 1 we have that $Flat(\text{MOD})$ has pushouts. We also have that $\mathbb{S}ign^{\rightarrow}$ has pushouts because colimits in functor categories are calculated pointwise from co-limits in the base category. Moreover, P_1 and P_2 create pushouts.

Consider a base $b\colon (\bullet \leftarrow \bullet \rightarrow \bullet) \longrightarrow \mathbb{S}ign$ in $\mathbb{S}ign^{\mathcal{C}}$ and let $\mu\colon bQ_iP_i \rightarrow \Sigma$ be its pushout in $\mathbb{S}ign$. Because P_i create pushouts, there are pushouts μ_i of bQ_i with $\mu_i P_i = \mu$. From the structure of pullbacks in $\mathbb{C}at$, we get a co-cone μ' over b such that $\mu' Q_i = \mu_i$. We show that μ' is a pushout of b. Let β be another co-cone over b. Then, βQ_i is a co-cone over μ_i, so we have unique q_i such that $\mu_i; q_i = \beta Q_i$. Again by the structure of pullbacks in $\mathbb{C}at$, this gives the unique q such that $\mu'; q = \beta$. □

The problem of combining constraint solvers over different data types can then be formulated as:

Solve a query for the shared combination $\Gamma_1 +_{\Gamma_0} \Gamma_2$ when given constraint solvers for the components Γ_1 and Γ_2.

One very elegant algebraic approach to the problem of combining constraint solvers was given by Baader and Schultz (Baader and Schultz, 1994; Baader and Schulz, 1994). Their idea is essentially based on studying algorithms that decompose the problem of finding solutions for q to problems at the level of the components for which constraint solvers already exist.

Definition 7.3. Any algorithm such that given a query q in some shared combination of signatures and assuming some fixed models A_1 and A_2 for the signatures, outputs queries q_i, $i = 1,2$, for each component such that

$$(A_1 \models q_1 \text{ and } A_2 \models q_2) \text{ if } A_1 \oplus_{A_0} A_2 \models q.$$

is called a **decomposition algorithm**.

Then a constraint solver finding solutions for q would alternate applications of constraint solvers for q_1 and q_2 with steps of the decomposition algorithm. We refer to the work of Baader and Schultz (Baader and Schultz, 1994; Baader and Schulz, 1994) for examples of such decomposition algorithms, for the purpose of this section the general concept suffices. Notice also that the component queries generated by the decomposition algorithms are expected to have solutions only if the original (composed) problem admits a solution. This one-way implication reflects the intuition that the decomposition algorithms are irreversible processes in the sense that they do not necessarily admit inverse composition algorithms.

Although their approach is developed only for the case of unsorted universal algebra[16] and it lacks a conceptual framework such as constraint logics (including full constraint sentences, a Herbrand theorem and a concept of morphism of constraint logic theories), we feel that its essence fits very well our constraint logic approach.

The following result provides logical foundations for combination of constraint solving techniques within the framework of the constraint logic institution.

Corollary 7.2. Assume a CBEL institution satisfying the following:

— the **Adjointness Framework**,
— the categories of models have finite colimits,
— the reducts between the categories of models have left-adjoints,
— the forgetful functors from models to domains preserve filtered colimits,
— has pushouts for signatures, and

[16] Though easily extendible to MSA.

— is semi-exact.

Then for any constraint logic (over the assumed CBEL) theories Γ_1 and Γ_2 sharing Γ_0, and any query q for $\Gamma_1 +_{\Gamma_0} \Gamma_2$, there exists queries q_i for Γ_i, $i = 1,2$, such that

$$\Gamma_1 +_{\Gamma_0} \Gamma_2 \models q \text{ implies } \Gamma_1 \models q_1 \text{ and } \Gamma_2 \models q_2$$

provided there is a decomposition algorithm for queries at the level of the Herbrand models.

Proof. The shared combination of constraint logic theories $\Gamma_1 +_{\Gamma_0} \Gamma_2$ exists by Theorem 5. Then

$$\begin{array}{rll} \Gamma_1 +_{\Gamma_0} \Gamma_2 \models q & & \\ \text{(Herbrand Theorem for constraint logic 4)} & \text{iff} & 0_{\Gamma_1 +_{\Gamma_0} \Gamma_2} \models q \\ \text{(Proposition 3.1)} & \text{iff} & 0_{\Gamma_1} \oplus_{\Gamma_0} 0_{\Gamma_2} \models q \\ \text{(decomposition algorithm)} & \text{implies} & 0_{\Gamma_1} \models q_1 \text{ and } 0_{\Gamma_2} \models q_2 \\ \text{(Herbrand Theorem for constraint logic 4)} & \text{iff} & \Gamma_1 \models q_1 \text{ and } \Gamma_2 \models q_2 \end{array}$$

□

The technical conditions in the hypotheses of the previous result represent the joining of the hypotheses of Theorem 4 and of Theorem 5. At a first glance they might look quite heavy, however they are just very basic and describe a normal technical framework. In fact, all examples of Section 4.2 satisfy these conditions, all of them being quite intensively used logics nowadays in algebraic specification.

The meaning of Corollary 7.2 is the reduction of the problem in solving queries in combination of constraint logic theories which already have constraint solvers to that of decomposition algorithms at the level of Herbrand models.

8. Conclusions and Future Work

We developed a category-based semantics for constraint logic programming rigorously based on logic, the corresponding logical system being called *constraint logic*. We showed that models of constraint logic form a comma category over a model of built-ins and that the so-called "generalised polynomials" play the rôle of terms. We showed that constraint logic is an institution, and proved that constraint logic is a special case of category-based equational logic. One of the consequences of integrating constraint logic into (category-based) equational logic is a novel Herbrand theorem for constraint logic programming; we also proved that in practice the Herbrand model of a program is the quotient determined by the program on the free extension of the built-in model. Finally, we sketched some logical foundations for modular combination of constraint solvers based on the amalgamated sum of the Herbrand models of the corresponding constraint theories.

One of the most important research directions is to further explore the computational consequences of this semantics, especially in the context of modular combination of constraint solving techniques. "Constraint paramodulation" of (Diaconescu, 1996c) can be used[17] as a basis for developing *extensible modular* constraint languages, i.e., languages combining constraint solvers over different data types in truly modular fashion á la Clear-OBJ tradition. This needs further

[17] In a remotely similar way to the 'theory resolution' of (Stickel, 1985). Also, recent advances in making paramodulation-based techniques more effcient (see (Bachmair et al., 1995), for example) have to be incorporated in any system implementing constraint paramodulation.

development of the topic of Section 7.2 in conjunction with constraint paramodulation as operational semantics.

Another important research direction is the study of ECLP over non-conventional structures which are not based on sets, which might result in interesting new applications. This should technically be based on the abstract development of constraint logics over any category-based equational logic.

Finally, more theoretical research directions would be to explore new sufficient conditions for the existence of the initial conservative constraint model, and to investigate the mathematical and logical properties of constraint logic, including axiomatisability results specific to constraint logic.

Acknowledgments

I am grateful to Joseph Goguen for discussing this research over many years, for encouraging it, and for helping with polishing this paper. I also thank the second (anonymous) referee for insisting much on several points which led to better explanations and even some conceptual improvement.

References

Adamek, J. and J.Rossicki (1994). *Locally Presentable and Accessible Categories*. Number 189 in London Mathematical Society Lecture Notes. Cambridge Univ. Press.

Baader, F. and Schultz, K. U. (1994). On the Combination of Symbolic Constraints, Solution Domains, and Constraint Solvers. Technical Report CIS-Rep-94-82, CIS-Universitat Muenchen.

Baader, F. and Schulz, K. U. (1994). Combination of constraint solving techniques: An algebraic point of view. Technical Report CIS-Rep-94-75, CIS-Universitat Muenchen.

Bachmair, L., Ganzinger, H., Lynch, C., and Snyder, W. (1995). Basic paramodulation. *Information and Computation*, 121(2):172–192.

Birkhoff, G. (1935). On the structure of abstract algebras. *Proceedings of the Cambridge Philosophical Society*, 31:433–454.

Burstall, R. and Goguen, J. (1980). The semantics of Clear, a specification language. In Bjorner, D., editor, *Proceedings, 1979 Copenhagen Winter School on Abstract Software Specification*, pages 292–332. Springer. Lecture Notes in Computer Science, Volume 86; based on unpublished notes handed out at the Symposium on Algebra and Applications, Stefan Banach Center, Warsaw, Poland, 1978.

Colmerauer, A. An introduction to PrologIII. Technical report, Groupe Intelligence Artificielle, Faculte de Sciences de Luminy.

Căzănescu, V. (1993). Local equational logic. In Esik, Z., editor, *Proceedings, 9th International Conference on Fundamentals of Computation Theory FCT'93*, pages 162–170. Springer-Verlag. Lecture Notes in Computer Science, Volume 710.

Dershowitz, N. (1983). Computing with rewrite rules. Technical Report ATR-83(8478)-1, The Aerospace Corp.

Diaconescu, R. (1990). The logic of Horn clauses is equational. Technical Report PRG-TR-3-93, Programming Research Group, University of Oxford.

Diaconescu, R. (1994). Category-based semantics for equational and constraint logic programming. DPhil thesis, University of Oxford.

Diaconescu, R. (1995). Completeness of category-based equational deduction. *Mathematical Structures in Computer Science*, 5(1):9–41.

Diaconescu, R. (1996a). A category-based equational logic semantics to constraint programming. In Haveraaen, M., Owe, O., and Dahl, O.-J., editors, *Recent Trends in Data Type Specification*, volume 1130 of *Lecture Notes in Computer Science*, pages 200–221. Springer. Proceedings of 11th Workshop on Specification of Abstract Data Types. Oslo, Norway, September 1995.

Diaconescu, R. (1996b). Category-based modularisation for equational logic programming. *Acta Informatica*, 33(5):477–510.

Diaconescu, R. (1996c). Completeness of semantic paramodulation: a category-based approach. Technical Report IS-RR-96-0006S, Japan Advanced Institute for Science and Technology.

Diaconescu, R. and Futatsugi, K. (1998). *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*, volume 6 of *AMAST Series in Computing*. World Scientific.

Diaconescu, R., Goguen, J., and Stefaneas, P. (1993). Logical support for modularisation. In Huet, G. and Plotkin, G., editors, *Logical Environments*, pages 83–130. Cambridge. Proceedings of a Workshop held in Edinburgh, Scotland, May 1991.

Gabriel, P. and Ulmer, F. (1971). *Lokal Präsentierbare Kategorien*. Springer. Lecture Notes in Mathematics, Volume 221.

Goguen, J. (2000). *Theorem Proving and Algebra*. MIT. To appear.

Goguen, J. and Burstall, R. (1992). Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146.

Goguen, J. and Diaconescu, R. (1994a). An Oxford survey of order sorted algebra. *Mathematical Structures in Computer Science*, 4(4):363–392.

Goguen, J. and Diaconescu, R. (1994b). Towards an algebraic semantics for the object paradigm. In Ehrig, H. and Orejas, F., editors, *Recent Trends in Data Type Specification*, volume 785 of *Lecture Notes in Computer Science*, pages 1–34. Springer.

Goguen, J. and Diaconescu, R. (1995). An introduction to category-based equational logic. In Alagar, V. and Nivat, M., editors, *Algebraic Methodology and Software Technology*, volume 936 of *Lecture Notes in Computer Science*, pages 91–126. Springer.

Goguen, J. and Malcolm, G. (1997). A hidden agenda. Technical Report CS97-538, University of California at San Diego.

Goguen, J. and Meseguer, J. (1985). Completeness of many-sorted equational logic. *Houston Journal of Mathematics*, 11(3):307–334.

Goguen, J. and Meseguer, J. (1987). Models and equality for logical programming. In Ehrig, H., Levi, G., Kowalski, R., and Montanari, U., editors, *Proceedings, 1987 TAPSOFT*, pages 1–22. Springer. Lecture Notes in Computer Science, Volume 250.

Goguen, J., Winkler, T., Meseguer, J., Futatsugi, K., and Jouannaud, J.-P. Introducing OBJ. In Goguen, J., editor, *Algebraic Specification with OBJ: An Introduction with Case Studies*. Cambridge. To appear.

Jaffar, J. and Lassez, J.-L. (1987). Constraint logic programming. In *14th ACM Symposium on the Principles of Programming languages*, pages 111–119.

Lane, S. M. (1971). *Categories for the Working Mathematician*. Springer.

Meseguer, J. (1992). Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155.

Mosses, P. (1989). Unified algebras and institutions. In *Proceedings, Fourth Annual Conference on Logic in Computer Science*, pages 304–312. IEEE.

Stickel, M. (1985). Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1(4):333–355.

Tarlecki, A., Burstall, R., and Goguen, J. (1991). Some fundamental algebraic tools for the semantics of computation, part 3: Indexed categories. *Theoretical Computer Science*, 91:239–264. Also, Monograph PRG–77, August 1989, Programming Research Group, Oxford University.

Tarski, A. (1944). The semantic conception of truth. *Philos. Phenomenological Research*, 4:13–47.