An Abstract View on Modularity in Knowledge Representation

Yuliya Lierler

Miroslaw Truszczyński
University of Kentucky

An Abstract View on Modularity in Knowledge Representation

Yuliya Lierler
Department of Computer Science
University of Nebraska at Omaha
Omaha, NE 68182, USA
ylierler@unomaha.edu

Miroslaw Truszczynski
Department of Computer Science
University of Kentucky
Lexington, KY 40506, USA
mirek@cs.uky.edu

Abstract

Modularity is an essential aspect of knowledge representation theory and practice. It has received substantial attention. We introduce *model-based modular systems*, an abstract framework for modular knowledge representation formalisms, similar in scope to multi-context systems but employing a simpler information-flow mechanism. We establish the precise relationship between the two frameworks, showing that they can simulate each other. We demonstrate that recently introduced modular knowledge representation formalisms integrating logic programming with satisfiability and, more generally, with constraint satisfaction can be cast as modular systems in our sense. These results show that our formalism offers a simple unifying framework for studies of modularity in knowledge representation.

Introduction

Modularity is crucial in design, analysis and reasoning about complex systems. It has long been recognized as one of the key techniques in software development. Modularity has also played an important role in artificial intelligence and, in particular, in knowledge representation and reasoning. Formalisms and implemented systems supporting modular knowledge representation and exploiting modularity in efficient reasoning include logic programming modules (Oikarinen and Janhunen 2006), modular logic programs (Lierler and Truszczynski 2013), HEX-programs (Eiter et al. 2005), constraint answer set programming (Lierler 2014), and logics FO(ID) (Denecker and Ternovska 2008; de Cat et al. 2014) and SM(ASP) (Lierler and Truszczynski 2011). These formalisms address issues in (i) knowledge modeling such as principled modular knowledge base design, and (ii) reasoning where they exploit modular structure to improve the performance of solvers.

To identify and study general principles of modularity in knowledge representation, researchers proposed several general frameworks such as:

- *Abstract multi-context systems* (Brewka and Eiter 2007) that provide an abstract view on multi-context systems (McCarthy 1987; Giunchiglia 1993). These systems are constructed from *contexts*, each being a theory in some logic. An essential element of a multi-context system is the notion of a bridge rule used to model information flow between contexts. The formalism of Brewka and Eiter achieves its generality thanks to an abstract notion of a logic that eliminates syntactic and semantic particulars from consideration.
- *Mx-systems* (Tasharrofi and Ternovska 2011) that view a theory as a representation of a set of interpretations. Consequently, modules in the mx-system setting are simply sets of interpretations of some vocabulary. Modules have inputs and outputs. That allows them to be composed into larger modular systems. An mx-system may have several module components and a (possibly) complex structure that describes information flow between modules.
- *Abstract modular inference systems* (Lierler and Truszczynski 2014) that are designed to facilitate modeling *solvers* of multi-logic systems. Abstract modular inference systems represent theories in a logic in abstract terms: via inferences these theories support.

The focus of the first framework, abstract multi-context systems, is on knowledge modeling in multi-logic, modular environments. The focus of the latter two formalisms is on principles of reasoning in such environments.

In this work, we propose the notion of an *abstract model-based modular system*, a conceptually simpler alternative to nonmonotonic multi-context systems by Brewka and Eiter (2007). Modeling *contextual* information and the flow of information among *contexts* have long been among the central problems of knowledge representation (McCarthy 1987; Giunchiglia 1993). Applications in multi-agent systems provide a compelling motivation for formal frameworks to represent multi-context systems and study their properties. The contexts are commonly modeled by theories in some logics, and the information flow between contexts is modeled by *bridge rules*. To develop a general framework for studies of multi-context systems, Brewka and Eiter introduced an abstract representation of a logic and extended the concept of a bridge rule. Since their inception, multi-context systems have received substantial attention and inspired implementations of hybrid reasoning systems including DMCS (El-Din Bairakdar et al. 2010).

We base our abstract modular systems on a slightly modified notion of the abstract logic by Brewka and Eiter (2007).

Modules are just like Brewka and Eiter's contexts. The main and essential difference is in the way we model the information flow. We do away with bridge rules and instead rely on a simple idea of information sharing via variables (or atoms) that are common in modules. We show that our abstract framework is well suited for representing concrete modular knowledge systems studied in the literature such as modular logic programs (Lierler and Truszczynski 2013), logic SM(ASP) (Lierler and Truszczynski 2011), and constraint answer set programming (Lierler 2014). We also show that despite their simplicity, abstract modular systems can actually simulate multi-context systems of Brewka and Eiter. In fact, the two formalisms are of the same expressive power since, as we also show, the converse holds, too.

The primary focus of model-based systems is on modularity in knowledge modeling, just as in the case of abstract multi-context systems. However, the connection between model-based modular systems, on the one hand, and mx-systems by Tasharrofi and Ternovska (2011), logic SM(ASP) and constraint answer set programming, on the other, show that abstract modular systems also have a potential to offer insights into reasoning tools for hybrid modular knowledge representation formalisms.

We start by introducing model-based modular systems. We then illustrate how these systems capture modular logic programs, logic SM(ASP), and constraint answer set programming. Next, model-based systems are formally related to multi-context systems. The paper concludes with a discussion on the connection between model-based modular systems and mx-systems.

Model-based Modular Systems

A *language* is a set L of *formulas*. A *theory* is a subset of L. Thus the set of theories is closed under union and has the least and the greatest elements: $\emptyset$ and L. This definition ignores any syntactic details behind the concepts of a formula and a theory. A *vocabulary* is an infinite countable set of *atoms*. Subsets of a vocabulary σ represent (classical propositional) *interpretations* of σ. We write $Int(\sigma)$ for the family of all interpretations of a vocabulary σ.

Definition 1. *A* logic *is a triple* $\mathcal{L} = (L_{\mathcal{L}}, \sigma_{\mathcal{L}}, sem_{\mathcal{L}})$, *where*

1. $L_{\mathcal{L}}$ *is a language (the language of the logic* $\mathcal{L}$*)*
2. $\sigma_{\mathcal{L}}$ *is a vocabulary (the vocabulary of the logic* $\mathcal{L}$*)*
3. $sem_{\mathcal{L}} : 2^{L_{\mathcal{L}}} \rightarrow 2^{Int(\sigma_{\mathcal{L}})}$ *is a function assigning collections of interpretations to theories in* $L_{\mathcal{L}}$ *(the* semantics *of* $\mathcal{L}$*)*

If a logic $\mathcal{L}$ is clear from the context, we omit the subscript $\mathcal{L}$ from the notation of the language, the vocabulary and the semantics of the logic.

This definition of a logic is a slight specialization of the one used by Brewka and Eiter (2007). Namely, the semantics function sem maps theories into "belief sets" that are interpretations of some vocabulary σ rather than elements from some arbitrary set.

We are now ready to present the two central concepts of our paper.

Definition 2. *Let* $\mathcal{L} = (L_{\mathcal{L}}, \sigma_{\mathcal{L}}, sem_{\mathcal{L}})$ *be a logic. A theory of* $\mathcal{L}$*, that is, a subset of the language* $L_{\mathcal{L}}$ *is called a* (model-based) $\mathcal{L}$-module *(or a* module*, if the explicit reference to its logic is not necessary). An interpretation* $I \in Int(\sigma_{\mathcal{L}})$ *is a* model *of an* $\mathcal{L}$*-module* B *if* $I \in sem_{\mathcal{L}}(B)$.

For an interpretation I, by $I_{|\sigma}$ we denote an interpretation over vocabulary σ constructed from I by dropping all its members not in σ. For example, let σ_1 be a vocabulary such that $a \in \sigma_1$ and $b \notin \sigma_1$, then $\{a, b\}_{\sigma_1} = \{a\}$.

Definition 3. *A collection of modules, possibly in different logics and over different vocabularies is a* (model-based) abstract modular system. *For a modular system* $\mathcal{H}$*, the union of the vocabularies of the logics of modules in* $\mathcal{H}$ *forms the* vocabulary *of* $\mathcal{H}$*, denoted by* $\sigma_{\mathcal{H}}$*. An interpretation* $I \in Int(\sigma_{\mathcal{H}})$ *is a* model *of* $\mathcal{H}$ *if for every* $B \in \mathcal{H}$*, if* B *is an* $\mathcal{L}$*-module for some logic* $\mathcal{L}$*,* $I_{|\sigma_{\mathcal{L}}}$ *is a model of* B.

Modular Logic Programs and Logic SM(ASP) as Model-based Modular Systems

We start this section by briefly reviewing logic programs and two modular formalisms based on logic programming and propositional logic — modular logic programs (Lierler and Truszczynski 2013), and the logic SM(ASP) (Lierler and Truszczynski 2011). We then show that these formalisms can be seen as instantiations of our general concept of a modular system.

Literals over a vocabulary σ are expressions a and $\neg a$, where a ranges over σ. A *logic program* over σ is a set of *rules* of the form

$$a_0 \leftarrow a_1, \ldots, a_\ell,\ not\ a_{\ell+1}, \ldots,\ not\ a_m, \qquad (1)$$

where a_0 is an atom in σ or $\bot$ (empty), and each a_i, $1 \leq i \leq m$, is an atom in σ.

The expression a_0 is the *head* of the rule. The expression on the right hand side of the arrow is the *body*. We write $hd(\Pi)$ for the set of nonempty heads of rules in Π. We refer the reader to the paper by Lifschitz, Tang and Turner (1999) for details on the definition of an *answer set*.

Definition 4 (Lierler and Truszczynski 2011)**.** *A set* X *of atoms from a vocabulary* σ *is an* input answer set *of a logic program* Π *over* σ *if* X *is an answer set of the program* $\Pi \cup (X \setminus hd(\Pi))$.

Brewka and Eiter (2007) showed that their abstract notion of a logic captures default logic, propositional logic, and logic programs under the answer set semantics. For example, the logic $\mathcal{L} = (L, \sigma, sem)$, where

1. L is the set of propositional formulas over σ,
2. $sem(F)$, for a theory $F \subseteq L$, is the set of propositional models of F over σ,

captures propositional logic. We call this logic $\mathcal{L}$ the *pl-logic* and modules in the pl-logic, *pl-modules*.

Similarly, abstract logics of Brewka and Eiter subsume the formalism of logic programs under the input answer set semantics. Indeed, let us consider a logic $\mathcal{L} = (L, \sigma, sem)$, where

1. L is the set of logic program rules over σ,

2. $sem(\Pi)$, for a program $\Pi \subseteq L$, is the set of input answer sets of Π over σ,

We call this logic the *ilp-logic* and modules in this logic, *ilp-modules*.

Definition 5 (Lierler and Truszczynski 2013). *A* modular logic program *over a vocabulary σ is a set of logic programs over σ. A set $X \subseteq \sigma$ is an* answer set *of a modular logic program $\mathcal{P}$ if X is an input answer set of every program $\Pi \in \mathcal{P}$.*

The following result is a direct consequence of definitions.

Proposition 1. *Let $\mathcal{P} = \{\Pi_1, \ldots, \Pi_n\}$ be a modular logic program, where each Π_i, $1 \leq i \leq n$, is a program over a vocabulary σ_i. An interpretation $X \subseteq \sigma = \bigcup_{i=1}^{n} \sigma_i$ is an answer set of the modular logic program $\mathcal{P}$ if and only if X is a model of an abstract modular system $(\Pi_1, \ldots, \Pi_n)$, where each Π_i, $1 \leq i \leq n$, is viewed as an ilp-module.*

It follows that abstract modular systems over the ilp-logic capture modular logic programs. Since modular logic programs subsume the formalism of lp-modules by Oikarinen and Janhunen (2006), the same is true also for lp-programs.

Another class of modular knowledge representation formalisms combine logic programs with propositional theories. For example, *satisfiability modulo ASP* or SM(ASP) (Lierler and Truszczynski 2011), a formalism that is closely related to the logic PC(ID) (Mariën et al. 2008), combines a single propositional theory with a single logic program.

Definition 6 (Lierler and Truszczynski 2011). Theories *of SM(ASP) are pairs $[F, \Pi]$, where F is a set of propositional clauses over a vocabulary σ_{pl} and Π is a logic program over a vocabulary σ_{lp}. For an SM(ASP) theory $[F, \Pi]$, a set X of atoms over $\sigma_{pl} \cup \sigma_{lp}$ is a* model *of $[F, \Pi]$ if X is a model of F and an input answer set of* Π.

As before, directly from this definition it follows that SM(ASP) theories can be viewed as abstract modular systems combining a pl-module with an ilp-module.

Proposition 2. *Let $[F, \Pi]$ be an SM(ASP) theory, where F is a collection of clauses over a vocabulary σ_{pl} and Π is a logic program over a vocabulary σ_{lp}. An interpretation $X \subseteq \sigma_{pl} \cup \sigma_{lp}$ is a model of $[F, \Pi]$ if and only if X is a model of an abstract modular system (F, Π), where we treat F as a pl-module over σ_{pl} and Π as an ilp-module over σ_{lp}.*

Constraint Answer Set Programs as Modular Systems

Constraint answer set programming (CASP) (Lierler 2014) is a promising research direction that integrates answer set programming with constraint processing. Its potential stems from (i) its support to model constraints directly, and (ii) the availability of fast CASP solvers including CLINGCON (Gebser, Ostrowski, and Schaub 2009), and EZCSP (Balduccini 2009). In this section we show how constraint answer set programming can be cast in terms of model-based modular systems. We start by reviewing the basic concepts of CASP.

Let V be a set of variables and D a set of values for variables in V, or the *domain* for V. A *constraint* over V and D is a pair $\langle t, \mathbb{R} \rangle$, where t is a tuple of some (possibly all) variables from V and $\mathbb{R}$ is a relation on D of the same arity as t. A collection of constraints over V and D is a *constraint satisfaction problem* (CSP) over V and D. An *evaluation* of V is a function assigning to every variable in V a value from D. An evaluation ν *satisfies* a constraint $\langle (x_1, \ldots, x_n), \mathbb{R} \rangle$ (or is a *solution* of this constraint) if $(\nu(x_1), \ldots, \nu(x_n)) \in \mathbb{R}$. An evaluation *satisfies* (or is a *solution* to) a constraint satisfaction problem if it satisfies every constraint of the problem.

Let $c = \langle t, \mathbb{R} \rangle$ be a constraint and D the domain of its variables. Let k denote the arity of t. The constraint $\overline{c} = \langle t, D^k \setminus \mathbb{R} \rangle$ is the *complement* (or *dual*) of c. Clearly, an evaluation of variables in t satisfies c if and only if it does not satisfy $\overline{c}$.

Let σ_r and σ_c be two disjoint propositional vocabularies. We refer to their elements as *regular* and *constraint* atoms. Also, let $\mathcal{C}$ be a class of constraints. A *constraint answer set program* (CAS program) in $\mathcal{C}$ and over the vocabulary $\sigma = \sigma_r \cup \sigma_c$ is a triple $\langle \Pi, \mathcal{B}, \gamma \rangle$, where Π is a logic program over the vocabulary σ such that $hd(\Pi) \cap \sigma_c = \emptyset$, $\mathcal{B}$ is a set of constraints such that $\mathcal{B} \subseteq \mathcal{C}$, and γ is a function from the set $At(\Pi) \cap \sigma_c$ of constraint atoms of Π to the set $\mathcal{B}$ of constraints.

For a CAS program $P = \langle \Pi, \mathcal{B}, \gamma \rangle$ over the vocabulary $\sigma = \sigma_r \cup \sigma_c$ (understood as above), a set $X \subseteq \sigma$ is an *answer set*[1] of P if

- $X \cap \sigma_r \subseteq hd(\Pi)$ and $X \cap \sigma_c \subseteq At(\Pi)$
- X is an input answer set of Π, and
- the following CSP has a solution

$$\{\gamma(a) \colon a \in X \cap \sigma_c\} \cup \{\overline{\gamma(a)} \colon a \in (At(\Pi) \cap \sigma_c) \setminus X\}.$$

We will now introduce model-based modular systems designed to represent CAS programs in a class $\mathcal{C}$ of constraints.

We start by defining a *csp-logic* determined by $\mathcal{C}$. Theories of that csp-logic are meant to model constraint satisfaction problems constructed from constraints in $\mathcal{C}$. By $\sigma_{\mathcal{C}}$ we denote the vocabulary formed by the set of (fresh) names of the constraints in $\mathcal{C}$ (we write c^n for the name of the constraint $c \in \mathcal{C}$ and so, $\sigma_{\mathcal{C}} = \{c^n \colon c \in \mathcal{C}\}$). For a subset $\mathcal{B}$ of $\mathcal{C}$, we define $sem_{\mathcal{C}}(\mathcal{B})$ to contain an interpretation I of $\sigma_{\mathcal{C}}$ precisely when the following CSP has a solution

$$\{c \colon c \in \mathcal{B} \text{ and } c^n \in I\} \cup \{\overline{c} \colon c \in \mathcal{B} \text{ and } c^n \notin I\}.$$

We call the logic $(\mathcal{C}, \sigma_{\mathcal{C}}, sem_{\mathcal{C}})$ a *csp-logic* determined by $\mathcal{C}$ and modules in this logic, *csp-modules over* $\mathcal{C}$.

Next, we define mapping modules. Let $P = \langle \Pi, \mathcal{B}, \gamma \rangle$ be a CAS program in a class $\mathcal{C}$ of constraints and over a vocabulary $\sigma_r \cup \sigma_c$. A *mapping* module with respect to P is a pl-module over the vocabulary $\sigma_c \cup \sigma_{\mathcal{C}}$ that is denoted by M_γ and defined by

$$M_\gamma = \{a \leftrightarrow (\gamma(a))^n \colon a \in At(\Pi) \cap \sigma_c\}.$$

[1]The definition of answer sets for CAS programs as proposed by Lierler (2014) is different. One can show that the definition we present here is equivalent to the original one.

Finally, for a logic program Π over the vocabulary $\sigma_r \cup \sigma_c$ we define

$$\Pi^c = \Pi \cup \{\bot \leftarrow a\colon a \in \sigma_r \setminus hd(\Pi)\} \\ \cup \{\bot \leftarrow a\colon a \in \sigma_c \setminus At(\Pi)\}.$$

Let $\mathcal{C}$ be a class of constraints and $P = \langle \Pi, \mathcal{B}, \gamma \rangle$ a CAS program in $\mathcal{C}$ over a signature $\sigma_r \cup \sigma_c$ of regular and constraint atoms. We define a modular system P^m by setting $P^m = (\Pi^c, \mathcal{B}, M_\gamma)$, where Π^c is an ilp-module over the signature $\sigma_r \cup \sigma_c \cup \sigma_{\mathcal{C}}$, $\mathcal{B} \subseteq \mathcal{C}$ is a csp-module over $\mathcal{C}$, and M_γ is a mapping module with respect to P. The following theorem shows that a CAS program P can be viewed as a modular system P^m:

Theorem 3. *Let $\mathcal{C}$ be a class of constraints and $P = \langle \Pi, \mathcal{B}, \gamma \rangle$ a CAS program in $\mathcal{C}$ over a signature $\sigma_r \cup \sigma_c$. A set $X \subseteq \sigma_r \cup \sigma_c$ is an answer set for P if and only if $X \cup \{(\gamma(a))^n\colon a \in X \cap \sigma_c\}$ is a model of the modular system P^m.*

Proof. Let us set $X^n_\gamma = \{(\gamma(a))^n\colon a \in X \cap \sigma_c\}$.

Left-to-right: Let X be an answer set of P. By the definition of an answer set, (i) $X \cap \sigma_r \subseteq hd(\Pi)$ and $X \cap \sigma_c \subseteq At(\Pi)$, (ii) X is an input answer set of Π, and (iii) the CSP

$$\{\gamma(a)\colon a \in X \cap \sigma_c\} \cup \{\overline{\gamma(a)}\colon a \in (At(\Pi) \cap \sigma_c) \setminus X\} \quad (2)$$

has a solution. From (ii), it follows that X is an answer set of $\Pi \cup (X \setminus hd(\Pi))$. From (i), it follows that X satisfies all constraints in $\{\bot \leftarrow a\colon a \in \sigma_r \setminus hd(\Pi)\} \cup \{\bot \leftarrow a\colon a \in \sigma_c \setminus At(\Pi)\}$. Thus, X is an answer set of

$$\Pi \cup (X \setminus hd(\Pi)) \cup \{\bot \leftarrow a\colon a \in \sigma_r \setminus hd(\Pi)\} \\ \cup \{\bot \leftarrow a\colon a \in \sigma_c \setminus At(\Pi)\}.$$

Since $hd(\Pi) = hd(\Pi^c)$, X is an answer set of $\Pi^c \cup (X \setminus hd(\Pi^c))$. Thus, X is an input answer set of Π^c and, consequently, a model of the ilp-module Π^c. Since atoms $(\gamma(a))^n$ are not in $\sigma_r \cup \sigma_c$, $X \cup X^n_\gamma$ is a model of Π^c, too.

It remains to show that $X \cup X^n_\gamma$ is a model of $\mathcal{B}$ and M_γ. To illustrate the former we have to show that the CSP

$$\{c\colon c \in \mathcal{B} \text{ and } c^n \in X^n_\gamma\} \cup \{\bar{c}\colon c \in \mathcal{B} \text{ and } c^n \notin X^n_\gamma\} \quad (3)$$

has a solution. By the definition of CAS programs, γ is a function from the set $At(\Pi) \cap \sigma_c$ to the set $\mathcal{B}$. From the definition of X^n_γ, it follows that CSP (3) coincides with CSP (2). By (iii), CSP (2) has a solution. The fact that $X \cup X^n_\gamma$ is a model of M_γ follows immediately from the definition of X^n_γ.

Right-to-left: Let $X \cup X^n_\gamma$ be a model of $(\Pi^c, \mathcal{B}, M_\gamma)$. It follows that $X \cup X^n_\gamma$ is an input answer set of Π^c. Reasoning as before, we obtain that X is an input answer set of Π^c. Consequently, X is an answer set of the program $\Pi \cup (X \setminus hd(\Pi)$ and it satisfies all the constraints. The former implies that X is an input set of Π. The latter implies the inclusions $X \cap \sigma_r \subseteq hd(\Pi)$ and $X \cap \sigma_c \subseteq At(\Pi)$.

Since $X \cup X^n_\gamma$ is a model of $(\Pi^c, \mathcal{B}, M_\gamma)$, it also follows that $X \cup X^n_\gamma$ is a model of $\mathcal{B}$, and $X \cup X^n_\gamma$ is a model of M_γ. These two properties imply that the CSP (2) has a solution using the same arguments as above. □

Multi-context Systems

In this section we show that modular systems can express multi-context systems introduced by Brewka and Eiter (2007). Let us first review the notion of a multi-context system. Let $\mathcal{L}_1, \ldots, \mathcal{L}_n$ be logics, say $\mathcal{L}_i = (L_i, \sigma_i, sem_i)$, $1 \le i \le n$, and let us assume that the vocabularies σ_i of these logics are *pairwise disjoint*. A *bridge rule* over the logics $\mathcal{L}_1, \ldots, \mathcal{L}_n$ is an expression

$$b \leftarrow p_1, \ldots, p_k, not\ p_{k+1}, \ldots, not\ p_m, \quad (4)$$

where $b \in L_i$ for some logic $\mathcal{L}_i$, $1 \le i \le n$, and each p_j, $1 \le j \le m$, is an element of the vocabulary σ_k of some logic $\mathcal{L}_k$ such that $k \neq i$. We call b the *head* and the expression to the right of '$\leftarrow$' the body of the bridge rule (4). We denote the head of a bridge rule r by $hd(r)$ and its body by $bd(r)$. If the head of a bridge rule is an element of L_i, we say that the bridge rule is *into* $\mathcal{L}_i$. For an interpretation I of σ, we say that a rule r is *applicable* in I if I satisfies the body of the rule r identified with a propositional formula $p_1 \wedge \ldots \wedge p_k \wedge \neg p_{k+1} \wedge \ldots \wedge \neg p_m$, written $I \models bd(r)$.

We are now ready to define multi-context systems and their semantics. A *multi-context system* (*MCS*) over logics $\mathcal{L}_1, \ldots, \mathcal{L}_n$ (with pairwise disjoint vocabularies) is an $(n+1)$-tuple $\mathcal{M} = (M_1, \ldots, M_n, R)$, where for each i, $1 \le i \le n$, M_i is an $\mathcal{L}_i$-module, and R is a set of bridge rules over $\mathcal{L}_1, \ldots, \mathcal{L}_n$. In the sequel, by R_i we will denote the set of all bridge rules in R that are into the logic $\mathcal{L}_i$. An interpretation I of σ is an *equilibrium* model of $\mathcal{M}$ if for every i, $1 \le i \le n$,

$$I_{|\sigma_i} \in sem_i(M_i \cup \{hd(r)\colon r \in R_i,\ r \text{ applicable in } I\}),$$

where for every i, $1 \le i \le n$.

Brewka and Eiter (2007) considered bridge rules that could have theories – sets of elements from L_i – as their heads. We can simulate such a rule $B \leftarrow \varphi$, where $B \subseteq L_i$ and φ is a conjunction of literals over σ, by a *set* of bridge rules $\{b \leftarrow \varphi\colon b \in B\}$. Thus, the restricted form of bridge rules that we consider does not entail any loss of generality.

The main idea behind casting an arbitrary multi-context system $(M_1, \ldots, M_n, R)$ as a modular system is to express each module M_i together with all bridge rules in R_i as a single module in an appropriately defined logic.

Let $\mathcal{L}_i = (L_i, \sigma_i, sem_i)$, $1 \le i \le n$, be logics with pairwise disjoint vocabularies, and let $\sigma = \sigma_1 \cup \cdots \cup \sigma_n$. We define logics $\mathcal{K}_i = (K_i, \sigma, \overline{sem}_i)$, $1 \le i \le n$, as follows. We set K_i to consist of all expressions of the form $b \leftarrow \varphi$, where $b \in L_i$ and φ is a conjunction of literals over σ. If $\varphi = \top$ (empty conjunction), we simplify the notation to just b. In this way, we can see L_i as a sublanguage of K_i. In the same time, expressions $b \leftarrow \varphi$ allow us to model bridge rules. Having defined the language K_i, $1 \le i \le n$, we specify the function $\overline{sem}_i$. Namely, for a theory $T \subseteq K_i$, we define $\overline{sem}_i(T)$ to contain an interpretation $I \in Int(\sigma)$ if and only if $I_{|\sigma_i} \in sem_i(\{hd(t)\colon t \in T, I \models bd(t)\})$.

Let $\mathcal{H} = (M_1, \ldots, M_n, R)$ be a multi-context system over logics $\mathcal{L}_1, \ldots, \mathcal{L}_n$ with pairwise disjoint vocabularies. We define $M'_i = M_i \cup R_i$. By the comments above, we can see M'_i as a subset of K_i. Thus, M'_i is a $\mathcal{K}_i$-module and

$(M'_1, \ldots, M'_n)$ is a modular system over logics $\mathcal{K}_1, \ldots, \mathcal{K}_n$. We denote this system by $ms(\mathcal{H})$.

Theorem 4. *Let $\mathcal{H} = (M_1, \ldots, M_n, R)$ be a multi-context system over logics $\mathcal{L}_1, \ldots, \mathcal{L}_n$ with pairwise disjoint vocabularies σ_i. An interpretation $I \in Int(\sigma)$, where $\sigma = \sigma_1 \cup \ldots \cup \sigma_n$, is an equilibrium model of $\mathcal{H}$ if and only if I is a model of the modular system $ms(\mathcal{H})$.*

Proof. Let I be an interpretation of σ. By the definition, I is an equilibrium model of $\mathcal{H}$ if and only if for every i, $1 \leq i \leq n$,

$$I_{|\sigma_i} \in sem_i(M_i \cup \{hd(r) \colon r \in R_i,\ r \text{ applicable in } I\}).$$

Since $r \in R_i$ is applicable in I if and only if $I \models bd(r)$,

$$\begin{aligned} &M_i \cup \{hd(r) \colon r \in R_i,\ r \text{ applicable in } I\} \\ &\quad = M_i \cup \{hd(r) \colon r \in R_i,\ I \models bd(r)\}. \end{aligned}$$

Also by the definition, I is a model of $ms(\mathcal{H})$ if and only if for every i, $1 \leq i \leq n$, $I \in \overline{sem}_i(M'_i)$, that is,

$$I_{|\sigma_i} \in sem_i(\{hd(r) \colon r \in M'_i,\ I \models bd(r)\}.$$

Since $M'_i = M_i \cup R_i$ and all elements in M_i have empty body,

$$\begin{aligned} &\{hd(r) \colon r \in M'_i, I \models bd(r)\} \\ &\quad = M_i \cup \{hd(r) \colon r \in R_i,\ I \models bd(r)\}. \end{aligned}$$

Thus, the assertion follows. □

Modular Systems as Multi-context Systems

Up to now we focused on demonstrating the expressive power of model-based modular systems. Specifically, we proved that they capture several other modular frameworks studied in the literature. In this section we note that multi-context systems are as general as modular systems: every modular system can be cast as a multi-context system. The idea is to simulate the atom-sharing model of communication between modules employed in modular systems by means of bridge rules in multi-context systems.

We start by noting that each modular system $\mathcal{M} = (M_1, \ldots, M_n)$ over some logics $\mathcal{L}_i = (L_i, \sigma_i, sem_i)$, $1 \leq i \leq n$, can be regarded as a modular system in logics $\mathcal{L}'_i = (L_i, \sigma, sem'_i)$, where

- $\sigma = \sigma_1 \cup \ldots \cup \sigma_n$ and
- for each i and each $B \subseteq L_i$, $sem'_i(B)$ contains an interpretation $I \in Int(\sigma)$ if and only if $I_{|\sigma_i} \in sem_i(B)$.

Informally, the logics $\mathcal{L}'_i$ are extensions of the logics $\mathcal{L}_i$ to a larger, common, vocabulary so that new symbols in the vocabulary do not affect the meaning of the theory. The following simple result is a direct consequence of the definitions.

Proposition 5. *Let $\mathcal{M} = (M_1, \ldots, M_n)$ be a modular system over logics $\mathcal{L}_i = (L_i, \sigma_i, sem_i)$, $1 \leq i \leq n$. An interpretation $X \in Int(\sigma)$ is a model of $\mathcal{M}$ if and only if I is a model of $\mathcal{M}$ viewed as a modular system over logics $\mathcal{L}'_i = (L_i, \sigma, sem'_i)$, $1 \leq i \leq n$, where sem'_i is defined as above.*

From now on, without loss of generality, we consider modular systems over logics that have the same vocabulary.

Thus, let $\mathcal{M} = (M_1, \ldots, M_n)$ be a modular system over logics $\mathcal{L}_i = (L_i, \sigma, sem_i)$, $1 \leq i \leq n$. We now construct a multi-context system $mcs(\mathcal{M})$, over different but closely related logics, that captures the semantics of $\mathcal{M}$.

For $i = 1, \ldots, n$ and for every $a \in \sigma$, we introduce a fresh symbol a_i so that $a_i \notin L_1 \cup \ldots \cup L_n$. By θ_i we denote a vocabulary $\{a_i \colon a \in \sigma\}$, whereas by θ we denote a vocabulary $\theta_1 \cup \ldots \cup \theta_n$. For every i, $1 \leq i \leq n$, we set $L^e_i = L_i \cup \theta_i$. Since $L_i \cap \theta_i = \emptyset$, there is never any ambiguity between elements of the two sets. For every $B \subseteq L^e_i$, by $\widehat{sem}_i(B)$ we denote the set that contains an interpretation $J_i \in Int(\theta_i)$ precisely when the following two conditions hold:

1. $\{a \in \sigma \colon a_i \in J_i\} \in sem_i(B \cap L)$.
2. for every $a_i \in \theta_i$, condition $a_i \in B$ holds if and only if condition $a_i \in J_i$ holds.

Logic $\mathcal{N}_i$ is a triple $(L^e_i, \theta_i, \widehat{sem}_i)$. It is clear that every $\mathcal{L}_i$-module B is also an $\mathcal{N}_i$-module and its meaning in $\mathcal{N}_i$ is the same as in $\mathcal{L}_i$ (modulo the correspondence between the elements of the vocabularies θ_i and σ). This logic is also capable to represent bridge rules for simulating the atom-sharing communication model of modular systems.

We now represent modular systems over the logics $\mathcal{L}_i$ by multi-context systems over the logics $\mathcal{N}_i$. For a modular system $\mathcal{M} = (M_1, \ldots, M_n)$ over logics $\mathcal{L}_i = (L_i, \sigma, sem_i)$, we define the multi-context system $mcs(\mathcal{M}) = (M_1, \ldots, M_n, R)$, where every M_i is a module over the logic $\mathcal{N}_i$, $1 \leq i \leq n$, and R is a set of bridge rules $R = \{a_i \leftarrow a_j \colon a \in \sigma,\ i \neq j\}$. These are indeed correctly formed bridge rules, as expressions a_i showing in the heads belong to the language L^e_i of the logic $\mathcal{N}_i$. We note that R depends only on n and on the common vocabulary σ of the logics $\mathcal{L}_1, \ldots, \mathcal{L}_n$, and not on any particular choice of modules in a modular system over these logics.

Theorem 6. *Let $\mathcal{M} = (M_1, \ldots, M_n)$ be a modular system over logics $\mathcal{L}_i = (L_i, \sigma, sem_i)$. An interpretation $I \in Int(\sigma)$ is a model of $\mathcal{M}$ if and only if $\bigcup_{i=1}^n \{a_i \colon a \in I\}$ is an equilibrium model of the multi-context system $mcs(\mathcal{M})$. Moreover, every equilibrium model of $mcs(\mathcal{M})$ is of such form.*

Proof. For $I \in Int(\sigma)$, we define $I_i = \{a_i \colon a \in I\}$, $1 \leq i \leq n$, and $\tilde{I} = \bigcup_{i=1}^n I_i$. Clearly, $I_i \in Int(\theta_i)$ and $\tilde{I} \in Int(\theta)$.
Left-to-Right: The set R_i of all bridge rules in R that are into a module M_i is given by $R_i = \{a_i \leftarrow a_j \colon a \in \sigma, j \neq i\}$. Therefore,

$$\{hd(r) \colon r \in R_i,\ r \text{ is applicable in } \tilde{I}\} = \{a_i \colon a \in I\}.$$

Let $I \in Int(\sigma)$ be a model of $\mathcal{M}$. Since $\tilde{I}_{|\theta_i} = I_i$, to prove that $\tilde{I}$ is an equilibrium model of $mcs(\mathcal{M})$, we need to show that for every i, $1 \leq i \leq n$,

$$I_i \in \widehat{sem}_i(M_i \cup \{a_i \colon a \in I\}).$$

This amounts to showing that

$$\{a \in \sigma \colon a_i \in I_i\} \in sem_i(M_i) \qquad (5)$$

(the condition (2) of the definition of $\widehat{sem}_i$ is evident). However, since I is a model of $\mathcal{M}$, I is a model of M_i in the logic $\mathcal{L}_i$. Consequently, condition (5) holds.

Right-to-left: Let $J \in Int(\theta)$ be an equilibrium model of $mcs(\mathcal{M})$. By the definition, for every i, $1 \leq i \leq n$,

$$J_{|\theta_i} = \widehat{sem}_i(M_i \cup \{hd(r)\colon r \in R_i,\ r \text{ is applicable in } J\}).$$

By the condition (1) of the definition of $\widehat{sem}_i$, we have $\{a \in \sigma\colon a_i \in J_{|\theta_i}\} \in sem_i(M_i)$ Moreover, let $j \neq i$ and let $a_j \in J$. The bridge rule $a_i \leftarrow a_j$ is applicable in J and so, we also have that $a_i \in \{hd(r)\colon r \in R_i,\ r \text{ is applicable in } J\}$. By the condition (2) of the definition of $\widehat{sem}_i$, we have $a_i \in J_{\theta_i}$. Hence, for every i, j such that $i \neq j$, if $a_j \in J$ then $a_i \in J$ and, by symmetry, if $a_i \in J$ then $a_j \in J$. This observation implies that J is of the form $\tilde{I}$ for some interpretation I of σ (this I can be explicitly defined as, for instance, $I = \{a \in \sigma\colon a_1 \in J\}$). Consequently, $J_{|\theta_i} = I_i$ and so, $I_i \in \widehat{sem}_i(M_i)$. By the definition of $\widehat{sem}_i$, $I \in sem_i(M_i)$. Thus, I is a model of the modular system $\mathcal{M}$.

□

Mx-systems

Tasharrofi and Ternovska (2011; 2013) introduced *modular systems for model expansion task*, or *mx-systems* for short, as a unifying framework for combining logics (languages) and systems together. Their work was inspired by module-based framework due to Järvisalo et al. (2009). Tasharrofi and Ternovska defined mx-systems as certain *compositions* of *mx-modules*, studied properties of composition operators for *mx-modules*, and established the expressive power of mx-systems. In this section, we relate mx-systems to the abstract model-based modular systems.

To do so, we cast mx-systems in terms of our definition of logic. This, essentially means that we consider only a propositional fragment of the formalism of mx-systems, originally set in terms of interpretations of first-order signatures. Since generalizations of modular systems to the case when the semantics of modules is given in terms of first-order interpretations are straightforward, our discussion can be extended to the full formalism of mx-systems.

Definition 7. *Let $\mathcal{L} = (L_\mathcal{L}, \sigma_\mathcal{L}, sem_\mathcal{L})$ be a logic, let $\mathcal{B}$ be an $\mathcal{L}$-module, and ψ be a vocabulary such that $\psi \subseteq \sigma_\mathcal{L}$. A pair $(\mathcal{B}, \psi)$ is an* mx-module *over logic $\mathcal{L}$. An interpretation $I \in Int(\sigma_\mathcal{L})$ is a* model *of an mx-module $(\mathcal{B}, \psi)$ if I is a model of $\mathcal{B}$. Vocabularies ψ and $\sigma_\mathcal{L} \setminus \psi$ are called an* input *and* output *vocabularies respectively.*

Clearly, mx-modules and model-based modules are almost identical. The only difference is in the ability to explicitly specify input vocabulary in the case of mx-modules.

Modular systems proposed by Tasharrofi and Ternovska (2011) are restricted to those that can be built from *composable* and *independent* mx-modules. Mx-modules $\mathcal{M}_1$ and $\mathcal{M}_2$ are *composable* if output vocabularies of $\mathcal{M}_1$ and $\mathcal{M}_2$ are disjoint. An mx-module $\mathcal{M}_2$ is *independent* from an mx-module $\mathcal{M}_1$ if the input vocabulary of $\mathcal{M}_2$ is disjoint from the output vocabulary of $\mathcal{M}_1$. Tasharrofi and Ternovska proposed several operators constructing mx-systems from smaller ones, with mx-modules being elementary building blocks. They include *sequential composition, projection, union,* and *feedback*. Here we focus only on mx-systems resulting from applying the operator $\triangleright$ of sequential composition.

Definition 8. *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be mx-modules in logics over vocabularies σ_1 and σ_2, and let $\mathcal{M}_1$ and $\mathcal{M}_2$ be* composable *and $\mathcal{M}_2$ be* independent *from $\mathcal{M}_1$. The expression $\mathcal{M}_1 \triangleright \mathcal{M}_2$ is an mx-system (constructed by sequential composition). An interpretation I over $\sigma_{\mathcal{M}_1} \cup \sigma_{\mathcal{M}_2}$ is a* model *of $\mathcal{M}_1 \triangleright \mathcal{M}_2$ if $I|\sigma_1$ is a model of $\mathcal{M}_1$ and $I|\sigma_2$ is a model of $\mathcal{M}_2$.*

The relation between mx-modular systems (obtained by sequential composition) and model-based modular systems is evident.

Proposition 7. *Let $\mathcal{M}_1 = (\mathcal{B}_1, \psi_1)$ and $\mathcal{M}_2 = (\mathcal{B}_2, \psi_2)$ be mx-modules over logics $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, and let $\mathcal{M}_1 \triangleright \mathcal{M}_2$ be defined. An interpretation I over $\sigma_1 \cup \sigma_2$, where σ_i is the vocabulary of the logic $\mathcal{L}_i$, $i = 1, 2$, is a* model *of $\mathcal{M}_1 \triangleright \mathcal{M}_2$ if and only if I is a model of model-based modular system $(\mathcal{B}_1, \mathcal{B}_2)$ over $\mathcal{L}_1$ and $\mathcal{L}_2$.*

The proof is a direct consequence of the definitions of the semantics of the respective types of modular systems.

Restricting our discussion to systems built of two modules is inessential. Our discussion easily extends to allow multiple modules. But, it is important to note that the definition of sequential compositionality requires that modules be composable and independent. Model-based modular systems make no such restrictions and so, generalize mx-modular systems based on sequential composition method.

Conclusions

Modularity is an important aspect of knowledge representation formalisms and applications. Accordingly, there has been significant interest in abstract principles behind modularity in knowledge representation, with multi-context systems and mx-systems being prime examples of frameworks proposed as a setting for such studies. In this paper, we introduced model-based modular systems as yet another abstract framework. Its key distinguishing aspect is the simplicity of the information-sharing model it employs: modules share information through common elements of their vocabularies.

We demonstrated that proposed modular systems can be seen as abstract representations of modular logic programs, SM(ASP) theories, and CAS programs. On a more general note, we showed that modular systems, despite their simplicity, are as expressive as multi-context systems. We also identified a class of mx-systems that can be viewed directly as modular systems, again simplifying the former by eliminating assumptions of composability and independence.[2] The results of this paper provide strong evidence that the simple abstract framework we proposed has a significant unifying, simplifying, and explanatory potential in the studies of modularity in knowledge representation.

[2]It is of interest to investigate whether mx-systems based on other composition operators can be related in a natural way to modular systems.

References

Balduccini, M. 2009. Representing constraint satisfaction problems in answer set programming. in: Proceedings of ICLP Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP), https://www.mat.unical.it/ASPOCP09/.

Brewka, G., and Eiter, T. 2007. Equilibria in heterogeneous nonmonotonic multi-context systems. In *Proceedings of National Conference on Artificial Intelligence, AAAI 2007*, 385–390.

de Cat, B.; Bogaerts, B.; Bruynooghe, M.; and Denecker, M. 2014. Predicate logic as a modelling language: The IDP system. *CoRR* abs/1401.6312.

Denecker, M., and Ternovska, E. 2008. A logic for nonmonotone inductive definitions. *ACM Transactions on Computational Logic* 9(2).

Eiter, T.; Ianni, G.; Schindlauer, R.; and Tompits, H. 2005. A uniform integration of higher-order reasoning and external evaluations in answer set programming. In *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI 2005*, 90–96.

El-Din Bairakdar, S.; Dao-Tran, M.; Eiter, T.; Fink, M.; and Krennwallner, T. 2010. The DMCS solver for distributed nonmonotonic multi-context systems. In *Proceedings of the 12th European Conference on Logics in Artificial Intelligence, JELIA 2010*, LNCS 6341, 352–355. Springer.

Gebser, M.; Ostrowski, M.; and Schaub, T. 2009. Constraint answer set solving. In *Proceedings of 25th International Conference on Logic Programming, ICLP 2009*, LNCS 5649, 235–249. Springer.

Giunchiglia, F. 1993. Contextual reasoning. *Epistemologia* XVI:345–364.

Järvisalo, M.; Oikarinen, E.; Janhunen, T.; and Niemelä, I. 2009. A module-based framework for multi-language constraint modeling. In *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2009*, LNCS 5753, 155–168. Springer.

Lierler, Y., and Truszczynski, M. 2011. Transition systems for model generators — a unifying approach. *Theory and Practice of Logic Programming* 11(4-5):629–646. (Special Issue, Proceedings of the 27th International Conference on Logic Programming, ICLP 2011).

Lierler, Y., and Truszczynski, M. 2013. Modular answer set solving. In *Late-Breaking Developments in the Field of Artificial Intelligence*, volume WS-13-17 of *AAAI Workshops*. AAAI.

Lierler, Y., and Truszczynski, M. 2014. Abstract modular inference systems and solvers. In *Proceedings of the 16th International Symposium on Practical Aspects of Declarative Languages, PADL 2014*, LNCS 8324, 49–64. Springer.

Lierler, Y. 2014. Relating constraint answer set programming languages and algorithms. *Artificial Intelligence* 207C:1–22.

Lifschitz, V.; Tang, L. R.; and Turner, H. 1999. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence* 25:369–389.

Mariën, M.; Wittocx, J.; Denecker, M.; and Bruynooghe, M. 2008. SAT(ID): Satisfiability of propositional logic extended with inductive definitions. In *Proceedings of the 11th International Conference on Theory and Applications of Satisfiability Testing, SAT 2008*, volume 4996 of *LNCS*, 211–224. Springer.

McCarthy, J. 1987. Generality in Artificial Intelligence. *Communications of the ACM* 30(12):1030–1035.

Oikarinen, E., and Janhunen, T. 2006. Modular equivalence for normal logic programs. In *Proceedings of the 17th European Conference on Artificial Intelligence, ECAI 2006*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, 412–416. IOS Press.

Tasharrofi, S., and Ternovska, E. 2011. A semantic account for modularity in multi-language modelling of search problems. In *Proceedings of the 8th international Symposium on Frontiers of Combining Systems, FroCoS 2011*, LNCS 6989, 259–274. Springer.

Tasharrofi, S. 2013. *Arithmetic and Modularity in Declarative Languages for Knowledge Representation*. Ph.D. Dissertation, Simon Fraser University.